2021

Reactive Planning With Legged Robots In Unknown Environments

Vasileios Vasilopoulos
University of Pennsylvania

Reactive Planning With Legged Robots In Unknown Environments

Abstract

Unlike the problem of safe task and motion planning in a completely known environment, the setting where the obstacles in a robot's workspace are not initially known and are incrementally revealed online has so far received little theoretical interest, with existing algorithms usually demanding constant deliberative replanning in the presence of unanticipated conditions. Moreover, even though recent advances show that legged platforms are becoming better at traversing rough terrains and environments, legged robots are still mostly used as locomotion research platforms, with applications restricted to domains where interaction with the environment is usually not needed and actively avoided.

In order to accomplish challenging tasks with such highly dynamic robots in unexplored environments, this research suggests with formal arguments and empirical demonstration the effectiveness of a hierarchical control structure, that we believe is the first provably correct deliberative/reactive planner to engage an unmodified general purpose mobile manipulator in physical rearrangements of its environment. To this end, we develop the mobile manipulation maneuvers to accomplish each task at hand, successfully anchor the useful kinematic unicycle template to control our legged platforms, and integrate perceptual feedback with low-level control to coordinate each robot's movement.

At the same time, this research builds toward a useful abstraction for task planning in unknown environments, and provides an avenue for incorporating partial prior knowledge within a deterministic framework well suited to existing vector field planning methods, by exploiting recent developments in semantic SLAM and object pose and triangular mesh extraction using convolutional neural net architectures. Under specific sufficient conditions, formal results guarantee collision avoidance and convergence to designated (fixed or slowly moving) targets, for both a single robot and a robot gripping and manipulating objects, in previously unexplored workspaces cluttered with non-convex obstacles. We encourage the application of our methods by providing accompanying software with open-source implementations of our algorithms.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Mechanical Engineering & Applied Mechanics

First Advisor

Daniel E. Koditschek

Keywords

Collision avoidance, Formal methods, Legged robots, Reactive planning

Subject Categories

Mechanical Engineering | Robotics

REACTIVE PLANNING WITH LEGGED ROBOTS IN UNKNOWN ENVIRONMENTS

Vasileios Vasilopoulos

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2021

Supervisor of Dissertation

Daniel E. Koditschek
Alfred Fitler Moore Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Jennifer R. Lukes
Professor of Mechanical Engineering and Applied Mechanics

Dissertation Committee

George J. Pappas, UPS Foundation Professor of Electrical and Systems Engineering
Daniel E. Koditschek, Alfred Fitler Moore Professor of Electrical and Systems Engineering
Kostas Daniilidis, Ruth Yalom Stone Professor of Computer and Information Science
Mark Yim, Asa Whitney Professor of Mechanical Engineering and Applied Mechanics
Nicholas Roy, Professor of Aeronautics and Astronautics (MIT)

REACTIVE PLANNING WITH LEGGED ROBOTS

IN UNKNOWN ENVIRONMENTS

Dedicated to Ioanna, for always being there

ACKNOWLEDGMENTS

I would like to acknowledge the many sponsors that made this research possible. The present thesis has been supported by AFRL grant FA865015D1845 (subcontract 669737-1), AFOSR grant FA9550-19-1-0265 (Assured Autonomy in Contested Environments), the ARL/GDRS RCTA project, Coop. Agreement #W911NF-10-2-0016, and ONR grant #N00014-16-1-2817, a Vannevar Bush Fellowship held by my advisor, Dan Koditschek, sponsored by the Basic Research Office of the Assistant Secretary of Defense for Research and Engineering.

First and foremost, I would like to thank my advisor, Dan Koditschek. Each and every conversation with Dan has been fascinating and deeply inspiring; he has played a unique role in teaching me how to think and communicate with the world. I really cannot describe with words how privileged I feel to have conducted my PhD with one of the legends in the field, and what an amazing experience it is to collaborate with Dan on a daily basis. He has been like a father to me over the past few years and has heavily influenced my perspective on several issues, not necessarily restricted to the field of Robotics.

I also feel very grateful to have collaborated with Nick Roy, George Pappas and Kostas Daniilidis. Their unique expertise and advice, as well as their willingness to bring me in touch with their amazing students and post-docs made the burden of my PhD a little easier to bear, and helped me get a more holistic view of the field. They are all amazing researchers, with a unique way to communicate their ideas, and I deeply value the exposure to their ambitious research agendas. I would also like to thank Mark Yim for participating in my committee and providing useful comments.

During my PhD, I had the privilege to meet and work with many different people from other labs (some within and others not affiliated with Penn), that made this journey more exciting and enjoyable. I would like to thank my consistent collaborators, Will Vega-Brown, George Pavlakos, Sean Bowman and Yiannis Kantaros, for always supporting our common goals with their dedication and hard work, and for providing new research avenues by coming up with fascinating ideas. Even though all of them come from different research backgrounds, they have always been curious about my research and I have been trying to learn as much

as I can from them.

Kod*lab has been my second family during this journey and specials thanks belong to all of its members (past and present) for creating and maintaining a healthy, friendly and supportive work environment. I would particularly like to thank Omur Arslan, Avik De and Paul Reverdy for introducing me to the lab's work and helping me make my first research steps. I also feel very lucky to have collaborated with Turner Topping, whose brilliance and unique talent at making everything magically work never ceases to amaze me, and Diego Caporale, who always reminds me that the best engineers simply do careful work. Apart from research, the conversations with Turner and Diego about sports and politics have been my favorite part of the day and I have dearly missed them since the onset of the pandemic. Diedra Krieger was also always there to help with many administrative challenges and to provide logistical support for physical experiments with robots. Last but not least, Wei-Hsi Chen has been like a brother to me all these years; we entered the graduate program at the same time, we have taken many (maybe too many!) courses together and we have shared all the ups and downs of the PhD. He was always there to provide useful critique on my research, help me with his expertise on mechanical design, or even talk about an interesting show or movie to cheer me up.

This section would not be complete without acknowledging support from old and (many) new friends I made in Philadelphia. Special thanks belong to Spiros and Vaso for introducing me to the city, sharing many moments together, and being our close quarantine buddies during these past few difficult months. Our long conversations with George and Nikos (with or without beer) have also kept me sane over these years; I know that their friendship will endure even if we end up in different parts of the world. Christoforos has also been a great friend and an amazing resource; I would always be grateful for guiding me through the academic field since my applications to PhD programs. I would also like to thank Iosif for his loyal friendship and support, even though we are several thousands of miles apart; I will never forget that he mentored me during my undergraduate years and introduced me to this amazing world of research.

I would not be here without the love of my future wife, Ioanna, who has always been there for me throughout these years. Sharing a relationship over Skype with a 7-hour time difference for the first four years of my PhD was not easy, but I never felt alone in this journey; her valuable advice and emotional support managed to push me through the finish line. This thesis is dedicated to her and the lovely years lying ahead. Finally, I would like to thank my parents, Thanasis and Zacharoula, my brother, Panagiotis, and my grandfather, Vassilis, for their endless support, their unconditional love and their willingness to always fuel my dreams.

ABSTRACT

REACTIVE PLANNING WITH LEGGED ROBOTS

IN UNKNOWN ENVIRONMENTS

Vasileios Vasilopoulos

Daniel E. Koditschek

Unlike the problem of safe task and motion planning in a completely known environment, the setting where the obstacles in a robot's workspace are not initially known and are incrementally revealed online has so far received little theoretical interest, with existing algorithms usually demanding constant deliberative replanning in the presence of unanticipated conditions. Moreover, even though recent advances show that legged platforms are becoming better at traversing rough terrains and environments, legged robots are still mostly used as locomotion research platforms, with applications restricted to domains where interaction with the environment is usually not needed and actively avoided.

In order to accomplish challenging tasks with such highly dynamic robots in unexplored environments, this research suggests with formal arguments and empirical demonstration the effectiveness of a hierarchical control structure, that we believe is the first provably correct deliberative/reactive planner to engage an unmodified general purpose mobile manipulator in physical rearrangements of its environment. To this end, we develop the mobile manipulation maneuvers to accomplish each task at hand, successfully anchor the useful kinematic unicycle template to control our legged platforms, and integrate perceptual feedback with low-level control to coordinate each robot's movement.

At the same time, this research builds toward a useful abstraction for task planning in unknown environments, and provides an avenue for incorporating partial prior knowledge within a deterministic framework well suited to existing vector field planning methods, by exploiting recent developments in semantic SLAM and object pose and triangular mesh extraction using convolutional neural net architectures. Under specific sufficient conditions, formal results guarantee collision avoidance and convergence to designated (fixed or slowly moving) targets, for both a single robot and a robot gripping and manipulating objects, in

previously unexplored workspaces cluttered with non-convex obstacles. We encourage the application of our methods by providing accompanying software with open-source implementations of our algorithms.

Contents

List of Tables

List of Figures

Part I

Introduction, Related Work & Preliminaries

Chapter 1

Introduction

1.1 Motivation

Task and Motion Planning is a quite popular subject in the field of Robotics, as robotic platforms usually need to simultaneously reason about both the task sequence that achieves a desired end goal (e.g., determining the sequence of objects to be manipulated in an assembly task, or the sequence of locations to be visited in a patrolling scenario) and a motion plan that realizes a specific sub-task (e.g., determining the trajectories the robot's joints need to follow in order to grab an object, or the path that reaches a specific location). Existing methods [89, 181] can find a particular (often optimal [208]) solution to a task at hand, but require good prior knowledge [132], and do not generalize well in the presence of unanticipated conditions. Similarly, recent developments in Deep Reinforcement Learning [172] have yielded impressive results [141, 186], but are tied to a specific platform for which an abundance of data is needed. Overall, existing algorithms from the task planning literature are either *task-specific*, *environment-specific* or *platform-specific*, and are typically not accompanied by any formal proofs of correctness.

At the same time, even though recent advances in the field of legged robotics [26, 81, 101, 157, 217], including several demonstrations from companies [3, 29, 66, 195], show that legged machines are becoming better at traversing rough terrains and environments, legged robots are still mostly used as locomotion research platforms [186], and their limited commercial

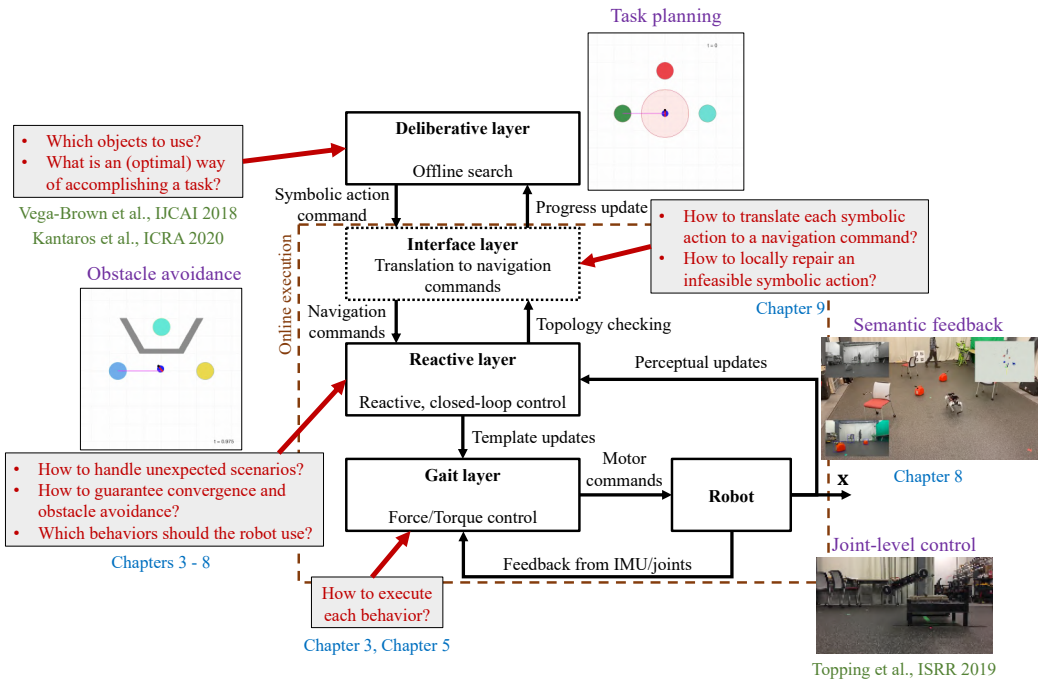

Figure 1.1: The proposed hierarchical control structure. In the deliberative layer, an offline high-level planner outputs a sequence of symbolic actions, that are executed online using a reactive controller that incorporates perception to account for unanticipated obstacles and issues abstract velocity and gripper commands. The communication between the deliberative and the reactive layer is facilitated by an interface layer that translates each symbolic action to appropriate navigation commands and locally repairs any infeasible actions. The low-level gait layer uses the commands from the reactive layer to call out appropriately parameterized joint-level feedback controllers for the robotic platform.

applications are restricted to inspection [4], security, and "last-meter" delivery [5], where interaction with the environment is not needed and rather avoided. Given the inherent ability of legged robots to use their limbs as general-purpose manipulators, this research seeks to demonstrate ways of accomplishing *tasks* with legged robots that require interaction with their à-priori unexplored surroundings, such as rearrangement planning [80], or navigation among movable obstacles [171] to escape a dangerous situation or help trapped people in search-and-rescue missions.

To this end, the present thesis proposes a modular, and task and platform independent architecture (inherently unavailable in end-to-end deep learning schemes), with formal correctness conclusions based on some underlying assumptions about the environment[1]. In

[1]In practice, these assumptions represent sufficient conditions for our formal correctness results. However, as demonstrated in the numerical results of Sections 4.6 and 9.5, and the experimental results of Sections 7.8 and 8.5, our reactive planning algorithms anecdotally work in way out of scope environments.

this architecture, an offline *deliberative layer* for task planning works closely with an online *reactive layer*, that uses exteroception and handles environment uncertainties. This interaction is facilitated by an *interface layer*, that translates each provided symbolic action to an appropriate navigation command (e.g., a path to be followed or a point to be reached) and locally repairs an infeasible action by rearranging the surrounding workspace when the underlying topological assumptions are violated. Finally, a platform-specific *gait layer*, comprised of a set of simple dynamical primitives, realizes the commands from the reactive layer in a way that is meaningful for the robot.

Each of these independent layers comes with provable guarantees of either probabilistic optimality [210] or completeness [92] (for the deliberative layer), collision avoidance and convergence (for the reactive layer), and low-level performance, expressed as "symbols" of energy landscapes composed either in parallel [48, 191] or sequentially [35, 126, 191] (for the gait layer), offering the chance of generalization across multiple mobile manipulators (legged or wheeled). The hope is that the adaptation, formalization and then coordination of these layers to communicate with each other in a meaningful way would generate a more powerful hierarchical structure, shown in Fig. 1.1, and would allow any robot in general (and any legged robot in particular) to interact with its environment and react to sudden changes in a predictable manner. Moreover, since the abstract commands from the reactive layer are given in a dynamically appropriate form (i.e., velocity or acceleration), the overall approach becomes better suited to the highly dynamic capabilities of legged robots compared to traditional AI planners, which provide position (e.g., paths) or merely symbolic commands and need to be accompanied by a separate kinodynamic planner to realize such commands.

Building toward these goals, Table 1.1 provides a summary of the (next outlined) thesis' contributions, along with a list of specific accompanying sections and publications.

1.2 Overview of Contributions

After a brief presentation of the empirical anchoring [61] of the unicycle template on the dynamic quadrupedal Minitaur robot and the reactive planning algorithm for unknown convex

Conceptual Content	Thesis Section	Published Literature
Preliminaries (Part I)		
Empirical anchoring of the kinematic unicycle template to control legged robots for motion planning tasks	3.1	[201]
Extension of motion planning algorithm with guarantees of collision avoidance and target convergence, to workspaces cluttered with well-separated but completely unknown convex obstacles [6, 7], using range-only target localization	3.2, 3.3	[201]
Mobile Manipulation in Partially Known Environments (Part II)		
First provably correct deliberative/reactive planner to engage an unmodified general purpose mobile manipulator in physical rearrangements of its environment	4.2, 4.3, 5.2.1, 5.2.2	[202, 203]
Development of steady-state and transitional maneuvers to accomplish tasks with Minitaur, and integration of perception with low-level feedback to control the robot's limbs	5.2.3	[202]
Development of reactive motion planning algorithm for workspaces cluttered with well-separated but completely unknown non-convex obstacles that obey specific "length-scale" geometric assumptions	5.3	[202]
Reactive Navigation in Unfamiliar Semantic Environments (Part III)		
Development of motion planning algorithms with simultaneous guarantees of collision avoidance and convergence to the designated goal, employing tools from the hybrid dynamical systems literature [87], in workspaces cluttered with: (i) well-separated unknown convex, and well-separated "familiar" star-shaped obstacles (ii) well-separated unknown convex, and "familiar" polygonal obstacles, with fixed or slowly moving targets	(i) 6.3, (ii) 7.4, 8.3	(i) [200], (ii) [204, 205]
Integration of developed motion planning algorithms with state-of-the-art perception and semantic mapping techniques on the Turtlebot, Minitaur and Spirit robots	7.7, 8.5	[204, 205]
Reactive Semantic Planning for Mobile Manipulation (Part IV)		
First planning and control architecture to provide a formal interface between an abstract temporal logic engine and a physically grounded mobile manipulation vector field planner	9.3	[206]
Description of conditions under which the symbolic controller is complete, and development of a new heuristic vector field controller for greedy physical rearrangement of the workspace when these conditions are violated	9.2, 9.4	[206]

Table 1.1: Thesis Contributions, mapped to specific sections and accompanying publications.

environments [6, 7] which will serve as the "backbone" of the thesis' main results in Part I (Chapter 3), Part II (Chapters 4 - 5) addresses a very specific instance of the Warehouseman's Problem [80] as a challenging setting in which to advance the formal integration of deliberative and reactive modes of robot assembly planning and control. We posit a planar disk-shaped robot with velocity controlled unicycle kinematics placed in an indoor environment with known floor-plan, cluttered with obstacles of unknown number and placement. The robot's task is to bring a collection of known disk-shaped objects from their initial placement to their prescribed destination by approaching, attaching and then pushing it into place, making sure to avoid any collisions with the known walls, other objects and unanticipated obstacles along the way. We show that this is the first provably correct deliberative/reactive planner to engage an unmodified general purpose mobile manipulator in physical rearrangements of its environment, by switching between a path following phase, where the robot follows a nominal path provided by an external deliberative planner, and a wall following phase with specific formal properties, where the robot avoids previously unanticipated obstacles in the environment. We also develop the steady-state and transitional maneuvers to accomplish such tasks with Minitaur, and integrate perceptual feedback with low-level limb control to coordinate the robot's movement.

Motivated by the need for robust reactive controllers that enhance the capabilities of deliberative task planners by handling unanticipated conditions during execution time, Part III of the thesis (Chapters 6 - 8) considers the navigation problem in a 2D workspace cluttered with unknown convex obstacles, along with "familiar" non-convex obstacles that belong to classes of known geometries, but whose number and placement are à-priori unknown. We assume a limited-range onboard sensor and a catalogue of known obstacles, along with a "mapping oracle" for their online identification and localization in the physical workspace. This framework allows the robot to explore the geometry and topology of its workspace in real time as it navigates toward its goal, by recognizing and incorporating in its stored semantic map "familiar" obstacles, whose number and placement are otherwise unknown, awaiting discovery at execution time. Based on the aforementioned description, we propose

a representation of the environment taking the form of a "multi-layer" collection of topological spaces whose realtime interaction can be exploited to integrate the geometrically naive sensor driven methods of [7] (briefly presented in Chapter 3) with the offline geometrically sensitive methods of [165], and show that our framework guarantees both obstacle avoidance and convergence to fixed or slowly moving targets, by relying on tools from the hybrid dynamical systems literature [87].

Finally, seeking to combine the mobile manipulation capabilities introduced in Part II with the reactive planning architectures from Part III, Part IV of this thesis revisits the Warehouseman's problem, abandons the path following phase of offline-computed paths, characterized by computationally expensive, offline deliberative search, and replaces the probabilistically optimal deliberative layer with a formal interface between an abstract temporal logic engine and a mobile manipulation vector field planner for the rearrangement of movable objects in semantically unexplored environments. We describe the conditions under which our architecture is complete, and introduce a new heuristic vector field controller for greedy rearrangement of the physical environment when these conditions are violated.

1.3 Contributions Mapped to Thesis Organization

In Chapter 3, our main advance is to use the emerging understanding of Minitaur's bounding [48] and walking trot gaits to improve its horizontal plane behavior to the point of exhibiting the dynamics of a horizontal plane unicycle, which we can then adopt as the *navigation template* [61] assumed by the reactive navigation algorithms [6, 7] it must execute. A second contribution is to realize this algorithm in a GPS-denied environment by recourse to a body-frame, range-only target localization scheme. More specifically, the robot is assumed to possess only an RF sensor providing range measurements from the desired goal. In addition to the LIDAR signals used to avoid the unknown obstacles, our algorithm uses only this one-dimensional information to extract the (two-dimensional) position of the goal, and the reformulation of the navigation algorithm in [6] in the robot's body frame allows for successful homing while guaranteeing obstacle avoidance along the way.

In Chapter 4, we present a provably correct architecture for planning and executing a successful solution to the Warehouseman's problem [80] by decomposition into an offline "deliberative" planning module and an online "reactive" execution module. The deliberative planner [210], adapted from the probabilistically complete (and optimal) algorithm of [128], is assigned the job of finding an assembly plan, while the reactive planner accepts each next step of that planned sequence, and uses online (LIDAR-style) sensory measurements to avoid the unanticipated obstacles (as well as the known walls and objects) by switching between following the deliberative planner's specified path or instead following a sensed wall. The wall following algorithm is guaranteed to maintain the robot distance from the wall within some specified bounds, while making progress along the wall boundary. After imposing specific constraints on how tightly packed the unknown obstacles and the known objects' initial and final configurations can be, we prove that the hybrid control scheme generated by this reactive planner must succeed in achieving any specified step of the deliberative sequence with no collisions along the way. Moreover, the reactive module serves as a useful tool to abstract away the geometric details of the environments and relieve the computational burden of the deliberative layer, by handling unanticipated obstacles online, during execution time. In turn, this significantly reduces the overall planning time, by letting the deliberative layer focus just on the high-level task planning problem solution — taking the form of a sequence of robot traversals, grasps, pushes and releases that would rearrange the environment as specified — were the known objects (along with the walls of the floor plan) the only obstacles to be dealt with.

In Chapter 5, we recruit the Minitaur quadruped [65] as a legged "mobipulator" [130] — a mobile robot that uses only its native, general purpose mechanical appendages to effect work on itself and the surrounding environment — in order to re-arrange according to a user's command the location of objects in a known environment that is sparsely obstructed by unanticipated, immovable obstacles of unknown general placement and shape. The integration of deliberative and reactive layers as described above guarantees that a unicycle capable of pushing or releasing such objects at will must always accomplish its task so long

as the unanticipated objects are all convex and sufficiently sparsely placed relative to the known floor plan. Seeking to bring a greater degree of realism to that framework, we also relax the geometric restriction to convex obstacles and prove that the idealized unicycle will still succeed even when confronted with non-convex unanticipated objects at run-time, so long as they are "moderately curved" and "sufficiently sparse". We relax the mechanical assumption of an idealized gripper by adding an entirely new "gait layer" that translates the erstwhile unicycle's velocity and gripper commands into a Minitaur joint-level architecture taking the form that we conjecture meets the requirements of a simple hybrid dynamical manipulation and self-manipulation system[2] [87].

In Chapter 6, we adapt the construction of [164] to generate a realtime smooth change of coordinates (a *diffeomorphism*) of the mapped space of the environment into a (locally) topologically equivalent but geometrically more favorable model space, relative to which the sensor-based reactive methods of [7] can be directly applied. We prove that the conjugate vector field defined by appropriately transforming the reactive model space back through this diffeomorphism induces a vector field on the robot's physical configuration space that inherits the same formal guarantees of obstacle avoidance and convergence.

Since the robot's knowledge about the geometry and topology of its workspace at execution time is constantly updated, in Chapter 7 we extend the formal construction of our navigation framework by adopting a hybrid dynamical systems description and show that the resulting hybrid system both inherits the consistency properties outlined in [87] and safely drives the robot to the goal without violating given command limits. In both Chapters 6 and 7, we extend the construction to the case of a differential drive robot, by pulling back the extended field over planar rigid transformations introduced for this purpose in [7] through a suitable polar coordinate transformation of the tangent lift of our original planar

[2] Although this would insure at least that the hybrid system is guaranteed to be live and non-blocking [87], the formal relationships of the legged dynamics to the abstracted unicycle reference remain to be examined. Later work [191] presents empirical evidence that the delicate grasping tasks for mounting and dismounting objects, comprising the key pedipulation competences required for robust success of this approach beyond mere mobility, can be specified and executed by recourse to further abstraction that anchors a lexicon of low degree of freedom closed loop dynamical templates [61], in the high degree of freedom Minitaur robot [65], whose systematic parallel and sequential compositions [48] yield the full range of necessary grasping behaviors in a rational, robust, highly repeatable and reliable manner.

diffeomorphism and demonstrate, once again, that the physical differential drive robot inherits the same obstacle avoidance and convergence properties as those guaranteed for the geometrically simple model robot [7].

We believe that this is the first *doubly-reactive controller* (i.e., a navigation framework wherein not only the robot's trajectory but also the control vector field that generates it are computed online at execution time) that can handle arbitrary polygonal shapes in real time without the need for specific separation assumptions between the familiar obstacles, by combining perception and object recognition for the familiar obstacles with local range measurements (e.g., LIDAR) for the unknown obstacles, to yield provably correct navigation in geometrically complicated environments. Furthermore, unlike RRT-based [119] or PRM-based [97] algorithms, and similarly to other vector-field based approaches, our framework is capable of solving the overall "kinodynamic" problem online, instead of executing separate trajectory and motion planning, for both a fully actuated particle and a differential drive robot. Finally, by coupling the semantic SLAM framework of [30] and the object detection pipeline of [148] with our reactive planning architecture, we are able to localize against isolated semantic cues while navigating, instead of localizing against entire scenes [71] or visual geometric features [77]. Therefore, by training just on data from the objects the robot is expected to encounter, we introduce modularity and robustness in our approach, while simultaneously performing online planning that does not rely on specific features of a deep network architecture (e.g., number or type of layers), [19, 71, 113].

In Chapter 8, we introduce a new change of coordinates, replacing the (potentially combinatorially growing) triangulation on the fly from Chapter 7 with a fixed convex decomposition [68] for each catalogued obstacle and revisit the prior hybrid dynamics convergence result to once again guarantee obstacle free geometric convergence. These new formal advances streamline the reactive computation, enabling robust online and onboard implementation (perceptual updates at 4Hz; reactive planning updates at 30Hz), affording tight realtime integration of the Semantic SLAM engine [30], that integrates observations and semantic labels over time. Second, we incorporate a separate deep neural net that captures a wire

mesh representation of encountered humans [105], enabling our reactive module to track and respond in realtime to semantically labeled human motions and gestures in unexplored environments. In turn, realtime semantics combined with human recognition capability motivates the proof of new rigorous guarantees for the robots to track suitably non-adversarial moving targets, while maintaining collision avoidance guarantees. We suggest the utility of the proposed architecture with a numerical study including comparisons with a state-of-the-art dynamic replanning algorithm [143], and physical implementation on both a wheeled and legged platform in highly varied environments (cluttered outdoor and indoor spaces including sunlight-flooded floors as well as featureless hallways). Targets are robustly followed up to speeds amenable to the perceptual pipeline's tracking rate. Importantly, the semantic capabilities of our pipeline are exploited to introduce more complex task logic (e.g., track a given target unless encountering a specific human gesture). This motivates the integration of this reactive planning architecture in the overall hierarchical control architecture for mobile manipulation tasks, shown in Fig. 1.1, which is the main focus of Chapter 9.

In Chapter 9, we combine the reactive planning algorithm of Chapter 8 with the mobile manipulation capabilities of Chapter 4, to introduce the first planning and control architecture that provides a formal interface between an abstract temporal logic engine and a physically grounded mobile manipulation vector field planner for the rearrangement of movable objects in partially known workspaces cluttered with unknown obstacles. We provide conditions under which the temporal logic controller is complete, while exploiting the formal results presented in Chapter 8 to guarantee safe physical achievement of the symbolic controller's sub-tasks when they are feasible, and introduce a new heuristic vector field controller for greedy physical rearrangement of the workspace when they are not. We provide a variety of simulation examples that illustrate the efficacy of the proposed algorithm for accomplishing complex manipulation tasks in unknown environments.

Chapter 2

Overview of Related Work

2.1 Mobile Manipulation

Mobile manipulation has been heavily investigated in the field of Robotics, with the hope that mobile autonomous platforms interacting with and manipulating their surroundings could assist in a variety of applications, such as search and rescue missions, planetary exploration and home healthcare. As a particular example, existing literature has focused on the problem of navigation among movable obstacles (NAMO) [184], where the robot needs to grasp and move obstacles in order to connect disconnected components of the configuration space and reach its goal, with more recent extensions focusing on efficient heuristics for manipulation planning in unknown environments, using either modified versions of the A* algorithm [72] or Monte Carlo simulations [120].

Mobile manipulation was also a central theme of the DARPA Robotics Challenge (DRC) [45], with several robots engaging in complex manipulation tasks while navigating challenging terrains. A variety of robot designs and control approaches were presented at the contest, including the quasistatic quadruped RoboSimian [75], the humanoid Valkyrie [156] or DRC-HUBO [218], a humanoid with wheels on its knees, allowing the robot to simply drive on flat terrain and use its legs and limbs for more complex tasks. Although the DRC was a useful test for autonomous robots in the physical world, the (sometimes spectacular) robot failures at the contest [82] demonstrated the need for robust behaviors and assured autonomy [15],

which is the main focus of this thesis.

2.1.1 Task and Motion Planning for Mobile Manipulation Tasks

Task and motion planning for complex manipulation tasks, such as rearrangement planning of multiple objects, has recently received increasing attention [63, 112, 208]. However, existing algorithms are typically combinatorially hard and do not scale well, while they also focus mostly on *known* environments [74, 181]. As a result, such methods cannot be applied to scenarios where the environment is initially unknown or needs to be reconfigured to accomplish the assigned mission and, therefore, online replanning may be required, resulting in limited applicability. Instead, this work proposes an architecture (Fig. 1.1) for addressing complex mobile manipulation task planning problems, which can handle unanticipated conditions in unknown environments.

Planning the rearrangement of movable objects has long been known to be algorithmically hard (e.g., PSPACE hardness was established in [80]) and a lively contemporary literature [51, 211] continues to explore conditions under which the additional complexity of planning the grasps results in a deterministically undecidable problem. While that interface has been understood to be crucial for decades [38], the literature on reactive approaches to this problem has been far more sparse. For example, past work on reactive rearrangement using vector field planners such as navigation functions [165] assumes either that each object is actuated [95, 215] or that there are no other obstacles in the environment [10, 31, 94].

On the contrary, when considering more complicated workspaces, most approaches focus either on sampling-based methods that empirically work well [198], motivated by the typically high dimensional configuration spaces arising from combined task and motion planning [63, 112], or learning a symbolic language on the fly [107]. However, such methods require constant deliberative replanning in the presence of unanticipated conditions, come with no guarantee of task completion under partial prior knowledge, and their search time grows exponentially with the number of movable pieces [209].

Other approaches focus on the use of reactive temporal logic planning algorithms, that can account for environmental uncertainty in terms of incomplete environment models [1,

69, 70, 111, 117, 124, 125, 127]. Particularly, [69, 70] model the environment as a transition system which is partially known. Then, discrete controllers are designed by applying graph search methods on a product automaton. As the environment, i.e., the transition system, is updated, the product automaton is locally updated as well, and new paths are re-designed by applying graph search approaches on the revised automaton. A conceptually similar approach is proposed in [117, 127] as well. The works in [124, 125] propose methods to locally patch paths, as the transition system (modeling the environment) changes so that GR(1) (General Reactivity of Rank 1) specifications [149] are satisfied. Reactive to LTL specifications planning algorithms are also proposed in [1, 111], allowing the robot to react to the environment by using the task specification to capture this reactivity. Correctness of these algorithms is guaranteed if the robot operates in an environment that satisfies the assumptions that were explicitly modeled in the task specification. All these works rely on discrete abstractions of the robot dynamics [21, 151] while active interaction with the environment to satisfy the logic specification is neglected.

It should be noted that the problem of using a higher-level planner to inform subgoals of a lower-level planner for mobile manipulation tasks, as outlined in Fig. 1.1 and described in Chapters 4, 5 and 9, has been examined previously, and we build on prior work in hybrid systems and task planning. However, most work has focused on ad hoc abstractions that perform well empirically. For example, Wolfe et al. [216] use a task hierarchy to guide the search for a low-level plan by expanding high-level plans in a best-first way. This approach guarantees hierarchical optimality: it will generate the best plan which can be represented in a given task hierarchy. Ensuring optimality has always been difficult to achieve due to computational complexity. Berenson et al. [22] and Konidaris et al. [108] use specific formulations of hierarchy without guaranteeing optimality. Kaelbling and Lozano-Perez [89] avoid the computational cost by committing to decisions at a high level of abstraction, before a full low-level plan is available. Vega-Brown and Roy [208] provided a further step towards tractable planning with complex kinematic constraints, but no dynamically appropriate approach exists for the complex legged robot dynamics considered in Chapter 5.

2.1.2 Mobile Manipulation with Legged Robots

Specifically focusing on manipulation with legged robots, a large, longstanding [110] and still very active [52, 159] literature concerns the design and control of legged robots equipped with additional arms (and, not infrequently, wheeled legs [173]) for purposes of mobile manipulation, typically focusing on quasi-statically formed grasps and movements. The smaller but similarly longstanding [129] literature on dynamical pedipulation appears to focus even in recent years on impulsive interaction with the movable objects [40] including, seemingly most close to our work, a recent simulation study on repeated, constrained, impulsive pushes for controlled ball rolling [185]. In contrast, we seek dynamically formed force closure grasps for purposes of pushed or dragged rearrangement, as shown in Chapter 5.

2.2 Reactive and Sensor-Based Planning

Even as legged [83, 86, 217] and aerial [2, 62, 135, 187] robots engage increasingly realistic, unstructured environments, intuition suggests that prior experience ought to yield deterministic navigation guarantees, postponing statistical predictions of performance to estimated [192], learned [76] or simulated [96] characterizations of truly bewilderingly dense or moving environments. Similarly, sampling-based methods, motivated by the typically high dimensional configuration spaces arising from combined task and motion planning [63], can achieve asymptotic optimality [210], but no guarantee of convergence (or task completion) under partial prior knowledge or limited sampling. Moreover, their probabilistic completeness guarantees can be slow to be realized in practice, especially when confronting settings with narrow passages [138], as we later report in Chapter 8.

It should be also noted that, unlike the problem of safe navigation in a completely known environment, the setting where the obstacles are not initially known and are incrementally revealed online has so far received little theoretical interest. Some few notable exceptions include considerations of optimality in unknown spaces [84], online modifications to temporal logic specifications [117] or deep learning algorithms [18] that assure safety against obstacles, or the use of trajectory optimization along with offline computed reachable sets [109] for

online policy adaptations. However, none of these advances has achieved simultaneous guarantees of obstacle avoidance and convergence. In contrast, our compositional use of semantically tagged, learned-object recognizers, outlined in Chapters 6-8, affords systematic re-use across many different environments and achieves formal deterministic guarantees as well — at least up to their (admittedly still far from formally justifiable) idealization as perfect realtime perceptual oracles — even when faced with moving targets.

2.2.1 Reactive Navigation

Heretofore, deterministically safe, convergent reactive methods have required substantial prior knowledge of a static environment, whether encoded using navigation functions [59, 123, 165], harmonic potential functions [42, 213] or pre-computed sequences of "funnels" [126]. In contrast, sensor-driven planners in this general tradition [27, 28, 33, 60, 86, 102, 146, 180, 197, 199] have guaranteed collision avoidance but have offered no assurance of convergence to a designated goal.

Recent advances in the theory of sensor-based navigation [6–8] relying on the properties of metric projections on convex sets [115] (and other parallel approaches [13, 37, 83, 147]) add the key feature of guaranteed convergence to a designated goal, by trading away prior knowledge for the presumption of simplicity: unknown obstacles can be successfully negotiated in real time without losing global convergence guarantees if they are "round" (i.e., very strongly convex in a sense made precise in [9]).

However, this presumption, along with the additional requirement for enough separation between the obstacles in the workspace, limit the domain of application for such methods to geometrically simple environments and might prohibit successful navigation in complicated, unstructured environments with non-convex geometry. Hence, other reactive approaches either seek to appropriately modify the input reference signal to account for unanticipated (potentially non-convex) disturbances [162], or rely on stochastic frameworks that are empirically shown to improve performance with non-convex obstacles [160], with no guarantees of convergence.

2.2.2 Realtime Perception

In Chapters 7 and 8, we address these shortcomings by appeal to an agent's memory evoked by execution-time perceptual cues. Recent advances in semantic SLAM [14, 30] and object pose extraction using convolutional neural net architectures [93, 106, 148] now provide an avenue for systematically composing partial prior knowledge about the robot's workspace within a deterministic framework well suited to the vector field planning methods reviewed above.

Contrasting recent work has recruited end-to-end learning to achieve obstacle avoiding reactions within metric [71] or topological [169] representations of familiar semantic environments, or supplemented such deep-learned representations with reference paths [113], or optimally generated waypoint sequences [19] that guide the robot to its destination. Although such approaches cannot guarantee safe convergence to the robot's destination, they promote the importance of landmark-based navigation, already highlighted by parallel work in biology [85]. However, characteristically, the input to such architectures is raw visual data thereby generating egocentric reactions that are hostage to the experience of one particular environment.

More modular data driven methods that separate the recruitment of learned visual representation to support learned control policies achieve greater generalization [178], but even carefully modularized approaches that handcraft the interaction of learned topological plans with learned reactive motor control in a physically informed framework [134] cannot bake into their architectures properties that afford the guaranteed policies of convergence and obstacle avoidance outlined in Chapters 6 - 8.

2.2.3 Topologically Informed Navigation

Work on the topology of motion planning [57, 58] has overtaken earlier investigation of reactive (i.e., vector field) navigation planners [163, 164] to the point that, comparatively, only preliminary results on their intrinsic limitations have been reported [20]. It seems clear that our success in achieving such strong results for a broad class of partially known

environments is due to the simplicity of the problem class (punctured two dimensional manifolds have the homotopy type of a bouquet of circles), but we are not in a position to opine firmly on the likely limitations of this approach in higher dimensional settings.

Recently, several contributions have focused on either finding invariants for homology classes to facilitate optimal path search in known environments [23], exploiting data to enforce topological constraints [150], or conceptualizing sensor measurements related to the shape of an object in a topologically meaningful way using persistent homology [136]. In contrast, we extract geometric and topological information about the robot's workspace at execution time in order to construct a map between a geometrically complicated mapped space and a (topologically equivalent but geometrically simple) model space that can be used for planning purposes. To this end, we employ methods from the field of computational geometry for (online constructed) implicit description of geometric shape using R-functions [176], convex decomposition [98] and logic operations with polygons [41, 53, 54], as discussed in Chapters 6 - 8.

Chapter 3

Preliminaries on Reactive Navigation with Legged Robots

In this Chapter, we demonstrate a fully sensor-based reactive homing behavior on a physical quadrupedal robot, using onboard sensors, in simple (convex obstacle-cluttered) unknown, GPS-denied environments. Its implementation is enabled by our empirical success in controlling the legged machine to approximate the (abstract) unicycle mechanics assumed by the navigation algorithm, and our proposed method of range-only target localization using particle filters. Both the empirical unicycle anchoring and the reactive control principles, originally presented in [6] and later extended in [7] and [9], will serve as "building blocks" for the mobile manipulation algorithms presented in Chapters 4 - 5, as well as the reactive navigation algorithms for unexplored semantic environments presented in Chapters 6 - 8.

The Chapter is organized as follows. Section 3.1 gives a description of the Minitaur robot and the control strategy that empirically anchors a kinematic unicycle on Minitaur while it is executing a bounding or a walking trot gait. Section 3.2 summarizes the ideas behind both the locally sensed and the sensor-based motion planning strategy. Section 3.3 describes the proposed body-frame, range-only target localization algorithm that allows for successful homing. Section 3.4 continues with a description of our experimental setup. Section 3.5 begins by demonstrating the effectiveness and robustness of the doubly reactive motion

planning scheme in different experimental environments (Section 3.5.1) using the bounding gait and only offboard sensing, and continues with more experiments using the full sensor-based version of the algorithm and the target localization scheme (Section 3.5.2). Finally, Section 3.6 discusses a summary of our results.

3.1 Empirical Unicycle Anchoring on the Minitaur Robot

This Section describes the experimental platform mainly used for our physical experiments, the Minitaur quadruped [101], focusing on the empirical anchoring [61] of a first order unicycle model [12] in its bounding and walking gait. This anchoring becomes essential as it provides an "interface" between the more abstract differential drive model in the horizontal plane (assumed in the development of our reactive controllers) and the physical platform.

3.1.1 Minitaur Hardware

Minitaur (Fig. 3.1 [65, 101]) is a 6kg direct drive quadruped that has already demonstrated a variety of interesting behaviors, including a 48 cm vertical leap [101], bounding at a continuum of speeds up to 2 m/s, pronking, trotting, etc. [48]. From the already developed palette of behaviors, we mainly use the "bounding" [48] and "walking trot" gaits [49] (for moving with a desired fore-aft and angular velocity), and "standing" (employed before the beginning and after the end of any motion for safely starting and terminating experiments) behaviors.

The bounding gait can achieve higher speeds, but it induces a strong body pitching motion which makes the application of onboard, sensor-based navigation techniques quite hard. For this reason, we use the bounding gait in the context of a navigation algorithm using only local but "bird's eye" information about the surrounding obstacles from the motion capture arena, and the walking trot gait for fully sensor-based experiments with a LIDAR (for obstacle avoidance - see Section 3.2, and Chapters 5, 7 and 8), a range RF sensor (for target localization in the body frame - see Section 3.3), or an onboard stereo camera (for robot localization and object recognition - see Chapters 7 and 8).

As shown in Fig. 3.1, Minitaur consists of a symmetric body with four 2DOF (Degree-of-

Figure 3.1: The Ghost Minitaur [65] experimental platform.

Freedom) legs. Each leg consists of 2 symmetric RR chains closing at the toe, and is actuated by 2 direct-drive, brushless DC motors (T-Motor U8) mounted at the hip. The forward and inverse kinematics of this 5-bar mechanism[1], which allows for augmented available workspace for each leg, are presented in [100]. More details about the physical parameters of Minitaur can be found in [101]. Control and phase commutation of the motors is handled by a (custom) controller board, and the leg actions in order to generate a desired behavior are synchronized by a (custom) "mainboard" equipped with an ARM microcontroller, which is pre-programmed beforehand. A LiPo battery provides power to the system.

3.1.2 Bounding Gait as a Kinematic Unicycle

Bounding is a virtual bipedal gait, wherein the front pair and rear pair of legs are phase-locked to each other, and the steady state stepping pattern is an alternation of front and rear stance periods, typically with substantial aerial phases in between. Minitaur's bounding is implemented using compositional principles [47] yielding a controller which requires few parameters, and exerts no feedback phase coordination between the front and rear hips [48].

The bounding controller exposes two commands: horizontal plane translational speed

[1]The fifth bar between the 2 motors is considered to have zero length.

v_c and yaw rate ω_c. Heretofore, these parameters have been set by a human operator, but in this work, for the first time, we supply these parameters from a higher-level controller. It is worth noting here that BigDog was able to generate such control commands autonomously for following a leader [217], but here we focus on the autonomous navigation problem. Though we don't make any formal claims of anchoring [61], we present an empirical characterization of bounding Minitaur as a kinematic unicycle, and use this working model as a trial *navigation template* for our legged platform. Our ultimate goal is to abstract away the complicated bounding dynamics of Minitaur and allow the robot to be controlled by a high-level motion planner as a differential drive robot.

However, bounding Minitaur is very much a dynamic system, and requires a non-trivial amount of time to accelerate between different speeds and yaw rates. In fact, the stride rate (3Hz) limits the control authority available, since the body cannot be actuated in flight. We hypothesize that a dynamic unicycle model [140, 145] with limits on acceleration [83] would be the most appropriate horizontal template for Minitaur, but here, we instead smooth the inputs with an auto-regressive filter to reduce the magnitude of the acceleration.

Given as inputs a desired speed $v_d \in \mathbb{R}$, and yaw rate $\omega_d \in \mathbb{R}$, let v_c and ω_c be the commands sent to Minitaur. Then, we set

$$\dot{v}_c = -\sigma_v(v_c - v_d), \qquad \dot{\omega}_c = -\sigma_\omega(\omega_c - \omega_d), \tag{3.1}$$

for some $\sigma_v, \sigma_\omega \in \mathbb{R}_+$. Note that smaller σ_* results in a smoother output, and vice versa.

For the empirical characterization of our strategy, we send Minitaur time-varying signals, and plot its response. Fig. 3.2 shows the time trajectories of the observed speed and yaw (measured by the motion capture system described in Section 3.4) for a commanded sinusoidal signal of a fixed frequency. The 3Hz cutoff filter removes periodicities caused by Minitaur's 3Hz stride rate.

Minitaur's response to smooth commands is very accurate in yaw, and more lagged in speed. We believe that this is due to the very small σ_v that had to be used in (3.1) in order to limit acceleration, since lower speeds are necessitated in the case of vector fields with high

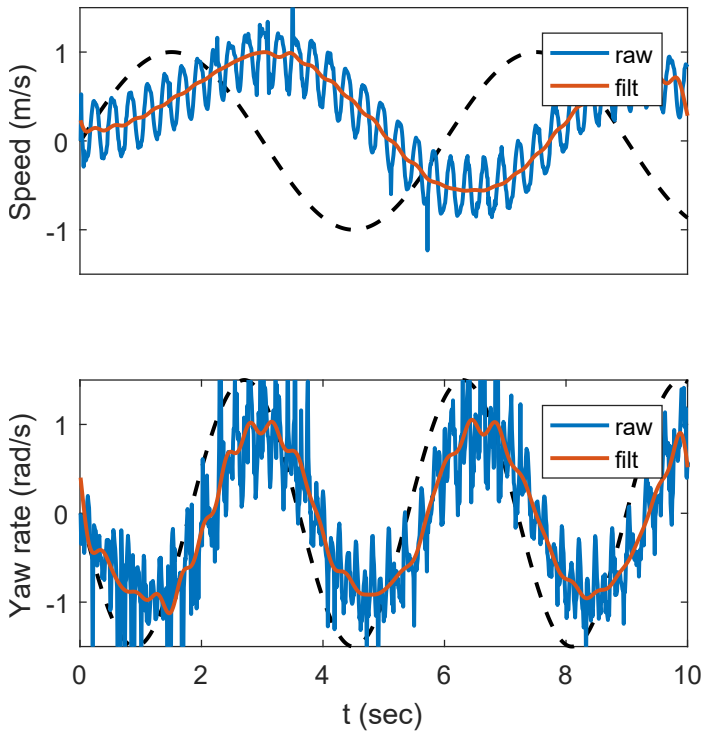

Figure 3.2: Frequency domain characterization of Minitaur's bounding response to smooth input signals v_d, ω_d (3.1): raw speed v and yaw response ω (blue), with a 3Hz cutoff filter (red), and the reference signals v_d, ω_d (black dashed).

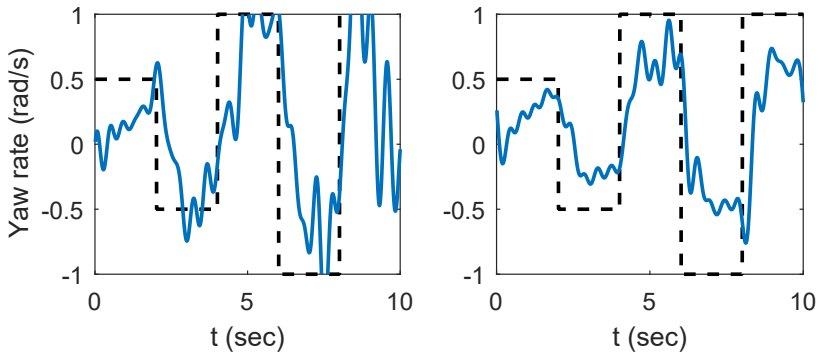

Figure 3.3: Minitaur's response to step signals in yaw rate, ω (blue), and the reference signal, ω_d (dashed black). For these trials, $v_d = 0$.

curvature. The robot's response to speed and yaw commands in typical experiment runs is shown in Fig. 3.14.

Response to Step Signals

Due to step signals having unbounded acceleration demands, Minitaur's performance in response to them is not as good as to smooth signals (especially in the heavily filtered speed command). In Fig. 3.3 we plot Minitaur's response to step signals in yaw (two trials), and in Fig. 3.4 we plot the response to speed signals (four trials). In the latter, we also include the robot's proprioceptive speed estimate (using the leg kinematics and joint velocities), which resemble the motion capture measurements closely. This confirms that the laggy speed tracking is not due to the robot's onboard speed estimate, but rather due to the heavy-handed smoothing (3.1) required to limit acceleration. For the experiments in these figures, the other input is set to zero.

3.1.3 Walking Trot Gait as a Kinematic Unicycle

The walking trot gait is also a virtual bipedal gait, wherein the diagonal pairs of legs are phase-locked to each other and the steady state stepping pattern is an alternation of diagonal stance periods with rapid flight phases in between. Although the formal analysis of this gait is still work in progress, we attempt an empirical characterization of the walking Minitaur as a kinematic unicycle, in an effort to use this model as the *navigation template* for sensor-based navigation and range-only target localization, since its negligible pitching motion

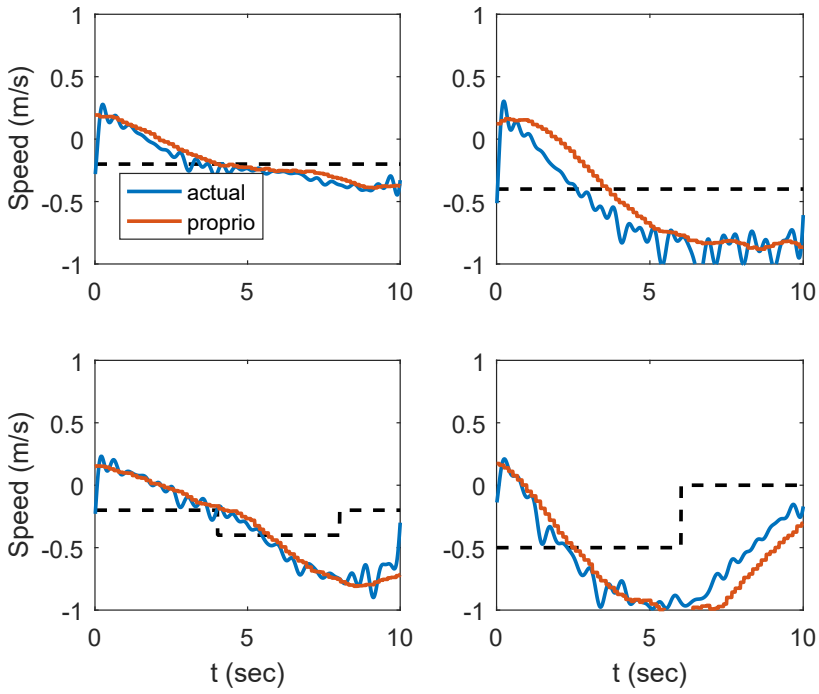

Figure 3.4: Minitaur's response to step signals in fore-aft speed v, given by the motion capture system (blue), and its proprioceptive speed estimate (red). The reference signal v_d is shown dashed black. For these trials, $\omega_d = 0$.

allows for the straightforward use of onboard sensors.

Similarly to the bounding gait, the walking trot controller exposes two commands: horizontal plane translational speed v_c and yaw rate ω_c, set by a higher-level controller. For the generation of smooth commands v_c and ω_c from the desired inputs v_d and ω_d we employ a first-order filter similar to (3.1) with lower gains σ_v, σ_ω, since we noticed that rapid changes in the inputs v_d, ω_d resulted in easier loss of traction and more falls compared to the bounding gait. We suspect that this occurs due to the more complicated stance kinematics of walking that make turning harder, but further investigation is currently underway.

As in the bounding gait trials, for the empirical characterization of our strategy, we send Minitaur time-varying signals and plot its response. Fig 3.5 shows the time trajectories of the observed speed and yaw (measured by a Vicon motion capture system [212]) for a commanded sinusoidal signal of a fixed frequency. We use a 3Hz cutoff filter to remove periodicities and numerical noise from the differentiation of the position signals.

Similarly to the bounding gait trials, we observe lagged response in speed and better

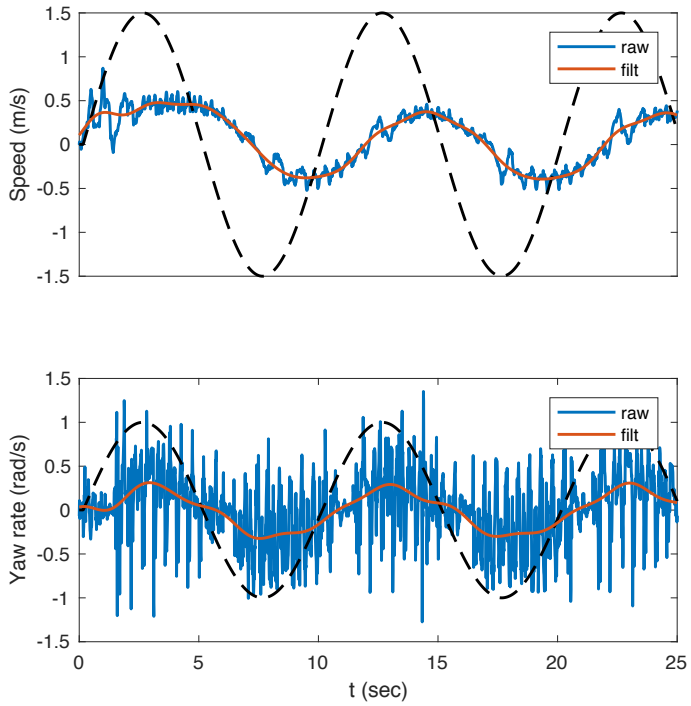

Figure 3.5: Frequency domain characterization of Minitaur's walking trot response to smooth input signals v_d, ω_d (3.1): time-domain plots of raw speed v and yaw response ω (blue), with a 3Hz cutoff filter (red), and the reference signals v_d, ω_d (black dashed).

frequency tracking in yaw. Small magnitudes in both speed and yaw can be attributed to the low gains σ_v, σ_ω we used, as well as physical limitations of the gait, which was developed for easier navigation over rough, uncluttered terrain rather than high-speed, energetically efficient motion.

Response to Step Signals

Finally, in Fig. 3.6 we show Minitaur's response to random step inputs in both fore-aft speed and yaw rate, supplied at the same time. Again, due to step signals having unbounded acceleration demands, Minitaur's performance is not as good as in the smooth case (especially in the heavily filtered speed command).

3.2 Reactive Navigation in Unknown Convex Environments

In this Section, we give an overview of the reactive navigation schemes that guarantee almost global navigation in convex workspaces using only local knowledge of the environment. We

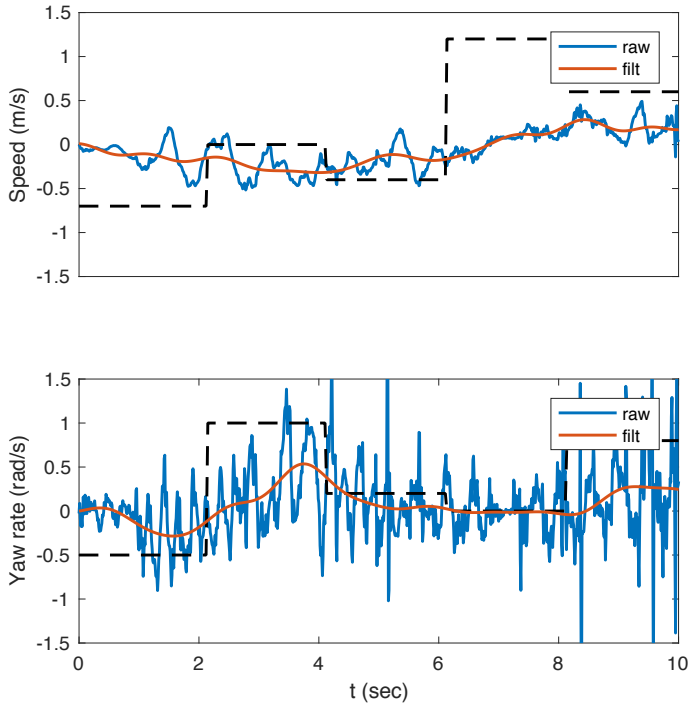

Figure 3.6: Minitaur's walking trot response to step signals in both fore-aft speed v (top), and yaw rate ω (bottom).

find it important to distinguish between the reactive navigation algorithm using local but "bird's eye" information, implemented on top of the bounding gait, and the fully sensor-based reactive navigation algorithm, implemented on top of the walking trot gait along with a LIDAR and a RF sensor, as described below and in later Chapters of the thesis.

In every case, it is assumed that the robot's motion is described by unicycle kinematics

$$\dot{\mathbf{x}} = v \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}, \tag{3.2}$$

$$\dot{\psi} = \omega, \tag{3.3}$$

3.2.1 Reactive Navigation Using Local but "Bird's Eye" Information

We use the algorithm in [6] as an example of a high-level strategy, capable of solving the navigation problem for a differential drive robot, in order to test the limits of the kinematic unicycle navigation template for bounding legged robots (as empirically validated in

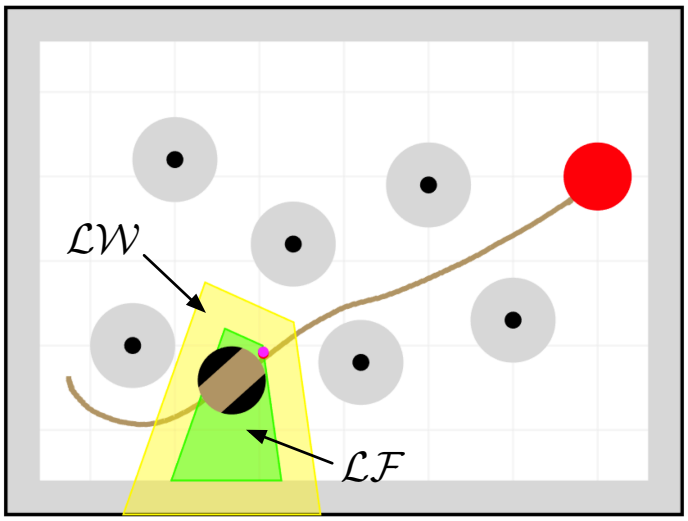

Figure 3.7: Reactive navigation with local but "bird's eye" information: A depiction of the "local workspace" $\mathcal{LW}$ (yellow polygon) and "local freespace" $\mathcal{LF}$ (green polygon) concepts that illustrates the local nature of the control strategy [6]. The goal position is shown as a solid red disk, and the local goal as a dot on one edge of the local freespace. The dark disks correspond to the physical obstacles, while the grey regions delimits the free space (for the robot's centroid) boundary. The trajectory corresponds to an experimental trial also shown in Fig. 3.12.

Section 3.1.2) in a real-world setting.

In brief, its construction utilizes power diagrams—generalized Voronoi diagrams with additive weights [16]—to identify a local workspace $\mathcal{LW}$ and a collision-free local freespace $\mathcal{LF} \subset \mathcal{LW}$ of a disk-shaped robot in a sphere world, and continuous motion towards the closest point in the robot's local safe neighborhood to a designated goal location is proven to asymptotically drive almost all robot configurations to the destination location with no collisions along the way, as in the example shown in Fig. 3.7.

Namely, the local workspace $\mathcal{LW}(\mathbf{x})$ for a robot with radius r at position $\mathbf{x}$, navigating a convex workspace $\mathcal{W}$ cluttered with N disk-shaped obstacles centered at $\mathbf{x}_i$ with radius r_i for $i \in \{1, \ldots, N\}$, is defined as the Voronoi cell

$$\mathcal{LW}(\mathbf{x}) := \{\mathbf{q} \in \mathcal{W} \,|\, ||\mathbf{q} - \mathbf{x}||^2 - r^2 \leq ||\mathbf{q} - \mathbf{x}_i||^2 - r_i^2, \forall i\} \tag{3.4}$$

In turn, in order to determine a collision-free neighborhood of the robot, the local freespace $\mathcal{LF}(\mathbf{x})$ is defined by eroding $\mathcal{LW}(\mathbf{x})$, removing the volume swept along its boundary $\partial\mathcal{LW}(\mathbf{x})$

by the robot radius r, as

$$\mathcal{LF}(\mathbf{x}) := \left\{ \mathbf{q} \in \mathcal{W} \,|\, \overline{\mathsf{B}\left(\mathbf{q}, r\right)} \subseteq \mathcal{LW}(\mathbf{x}) \right\} \tag{3.5}$$

with $\overline{\mathsf{B}\left(\mathbf{q}, r\right)}$ denoting the closure of the ball centered at $\mathbf{q}$ with radius r. For fully actuated particles with first-order dynamics defined as $\dot{\mathbf{x}} = \mathbf{u}$ navigating toward a goal $\mathbf{x}^*$, the control law $\mathbf{u} : \mathbb{R}^2 \to T\mathbb{R}^2$ can then simply be defined as

$$\mathbf{u}(\mathbf{x}) := -k \left(\mathbf{x} - \Pi_{\mathcal{LF}(\mathbf{x})}(\mathbf{x}^*) \right) \tag{3.6}$$

with $\Pi_A : \mathbb{R}^2 \to A$ denoting the projection function onto a convex subset $A \subseteq \mathbb{R}^2$, i.e.,

$$\Pi_A(\mathbf{q}) := \underset{\mathbf{a} \in A}{\operatorname{argmin}} \|\mathbf{a} - \mathbf{q}\| \tag{3.7}$$

It is also shown in [6] that this construction can be further adapted to a nonholonomically constrained "unicycle" robot, whose model is given in (3.2)-(3.3), while maintaining the stability and collision avoidance properties. In this case, the control inputs are given as

$$v = -k \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}^\top \left(\mathbf{x} - \Pi_{\mathcal{LF}_v(\mathbf{x})}(\mathbf{x}^*) \right) \tag{3.8}$$

$$\omega = k \operatorname{atan} \left(\frac{\begin{bmatrix} -\sin\psi \\ \cos\psi \end{bmatrix}^\top \left(\mathbf{x} - \frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*) + \Pi_{\mathcal{LF}(\mathbf{x})}(\mathbf{x}^*)}{2} \right)}{\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}^\top \left(\mathbf{x} - \frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*) + \Pi_{\mathcal{LF}(\mathbf{x})}(\mathbf{x}^*)}{2} \right)} \right) \tag{3.9}$$

with

$$\mathcal{LF}_v(\mathbf{x}) := \mathcal{LF}(\mathbf{x}) \cap H_\parallel \tag{3.10}$$

$$\mathcal{LF}_\omega(\mathbf{x}) := \mathcal{LF}(\mathbf{x}) \cap H_G \tag{3.11}$$

and $H_{\|}$ and H_G the lines defined as

$$H_{\|} = \left\{ \mathbf{z} \in \mathcal{W} \,\middle|\, \begin{bmatrix} -\sin\psi \\ \cos\psi \end{bmatrix}^{\top} (\mathbf{z} - \mathbf{x}) = 0 \right\} \tag{3.12}$$

$$H_G = \{\alpha\mathbf{x} + (1-\alpha)\mathbf{x}^* \in \mathcal{W} \,|\, \alpha \in \mathbb{R}\} \tag{3.13}$$

3.2.2 Sensor-Based Reactive Navigation

The algorithm in [6] was extended in [7], by replacing the Voronoi power diagrams with separating hyperplanes to account for a broader than spheres class of convex bodies and to accommodate a realistic 2D LIDAR sensor model for obstacle detection. As shown in Fig. 3.8, the algorithm relies again on the construction of a local workspace $\mathcal{LW}$ and a collision-free local freespace $\mathcal{LF} \subset \mathcal{LW}$ and continuous motion towards the closest point in the local freespace brings the robot to a designated goal location. However, as shown in [7] and in Fig. 3.8, the construction of these cells is now based on the intersection of the (local) LIDAR footprint with appropriately defined hyperplanes, one for each local minimum observed within this footprint.

More specifically, the LIDAR sensor is modeled by a polar curve $\rho_{\mathbf{x}} : (-\pi, \pi] \to [0, R]$ as follows

$$\rho_{\mathbf{x}}(\theta) := \min \begin{pmatrix} R \\ \min\{\|\mathbf{p} - \mathbf{x}\| \,|\, \mathbf{p} \in \partial\mathcal{W}, \operatorname{atan2}(\mathbf{p} - \mathbf{x}) = \theta\} \\ \min\limits_{i} \{\|\mathbf{p} - \mathbf{x}\| \,|\, \mathbf{p} \in O_i, \operatorname{atan2}(\mathbf{p} - \mathbf{x}) = \theta\} \end{pmatrix} \tag{3.14}$$

where R denotes the LIDAR sensor range, and O_i the i-th obstacle. Assuming that $\rho_i : (\theta_{l_i}, \theta_{u_i}) \to [0, R]$ is a convex curve segment of the LIDAR scan $\rho_{\mathbf{x}}$ at a location $\mathbf{x} \in \mathcal{W}$, then the associated "line-of-sight obstacle" [7] is defined as the open epigraph of ρ_i with its pole located at $\mathbf{x}$ as $L_i := \{\mathbf{x}\} \oplus \{(\rho\cos\theta, \rho\sin\theta) | \theta \in (\theta_{l_i}, \theta_{u_i}), \rho > \rho_i(\theta)\}$. Assuming, then, the availability of a sensor model $\mathcal{L}_R(\mathbf{x}) := \{L_1, L_2, \ldots, L_t\}$ that returns the list of convex line-of-sight obstacles[2] detected by the LIDAR scanner at location $\mathbf{x}$, the local workspace is

[2]Here t denotes the number of detected obstacles and changes as a function of robot location.

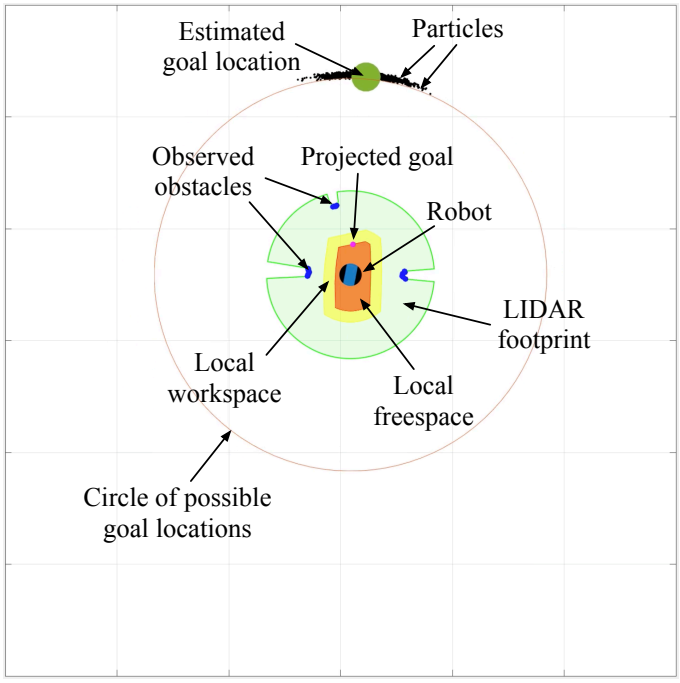

Figure 3.8: Sensor-Based Reactive Navigation: A depiction of the "local workspace" $\mathcal{LW}$ (yellow polygon) and "local freespace" $\mathcal{LF}$ (orange polygon) constructed from a LIDAR footprint (green) [7]. The estimated goal position (dark green dot) is calculated using range-only information and a particle filter. Notice how the particles spread on the circle with radius equal to the current range measurement. The local goal is computed from the projection of the estimated goal position onto $\mathcal{LF}$.

defined as [7]

$$\mathcal{LW}(\mathbf{x}) := \left\{ \mathbf{q} \in L_{ft}(\mathbf{x}) \cap \overline{\mathsf{B}\left(\mathbf{x}, \tfrac{r+R}{2}\right)} \;\middle|\; \left|\left| \mathbf{q} - \mathbf{x} + r \frac{\mathbf{x} - \Pi_{\overline{L}_i}(\mathbf{x})}{||\mathbf{x} - \Pi_{\overline{L}_i}(\mathbf{x})||} \right|\right| \leq ||\mathbf{q} - \Pi_{\overline{L}_i}(\mathbf{x})||, \forall i \right\} \tag{3.15}$$

where $L_{ft}(\mathbf{x})$ denoting the LIDAR sensory footprint at $\mathbf{x}$, given by the hypograph of the LIDAR scan $\rho_{\mathbf{x}}$ at $\mathbf{x}$ (see (3.14)), defined as $L_{ft}(\mathbf{x}) := \{\mathbf{x}\} \oplus \{(\rho\cos\theta, \rho\sin\theta) | \theta \in (-\pi, \pi], 0 \leq \rho \leq \rho_{\mathbf{x}}(\theta)\}$. Given the local workspace $\mathcal{LW}(\mathbf{x})$, the local freespace $\mathcal{LF}(\mathbf{x})$ is defined as in (3.5).

3.3 Body Frame Target Localization

In the sensor-based framework of Section 3.2.2, the problem of homing on a beacon using range-only measurements can become quite challenging (see e.g [196]). In the absence of global information, both the target localization and the navigation control strategy must

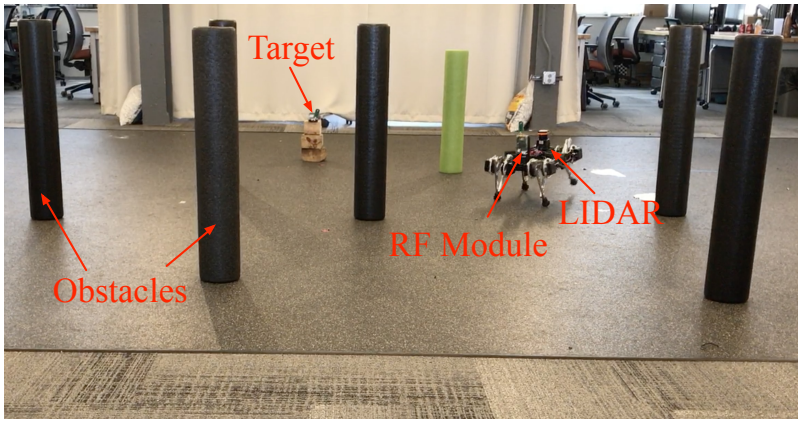

Figure 3.9: Minitaur navigating through an artificial forest towards a target.

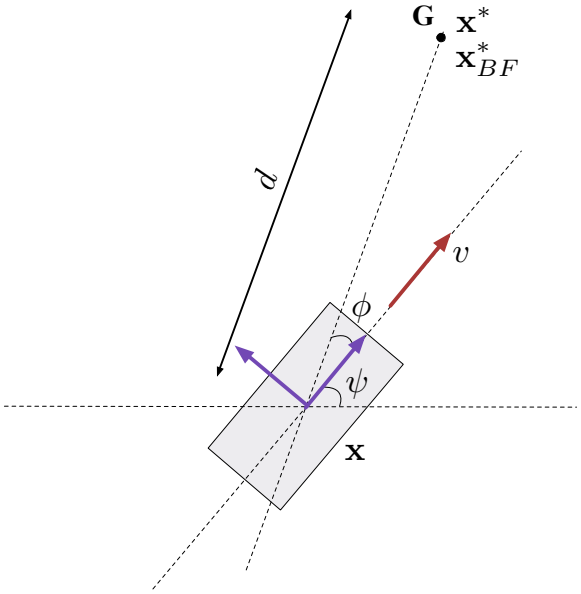

Figure 3.10: Range-only target localization in the robot's body frame (purple).

be appropriately modified for the robot's body frame [46, 88]. Thankfully, as we will see next, the algorithm in Section 3.2.2 can be reformulated in the robot's body frame, provided successful target localization. To facilitate our analysis, we refer to Fig. 3.10.

Localization Model

We assume that the robot is located at $\mathbf{x}$ with an orientation ψ, which are both unknown. The goal is to navigate to point G, whose position in the global frame $\mathbf{x}^* := (x^*, y^*) \in \mathbb{R}^2$ is also unknown to the robot. The robot can only measure (with some accuracy) its distance $d := ||\mathbf{x}^* - \mathbf{x}||$ from G. Let $\mathbf{x}^*_{BF} := (x^*_{BF}, y^*_{BF}) \in \mathbb{R}^2$ denote the target position in the

robot's body frame.

Lemma 3.1. *For the unicycle dynamics described in* (3.2)-(3.3)*, if* $\mathbf{x}^* = const.$*, then*

$$\dot{x}^*_{BF} = -v + \omega\, y^*_{BF} \tag{3.16}$$

$$\dot{y}^*_{BF} = -\omega\, x^*_{BF} \tag{3.17}$$

Proof. Included in Appendix C.1. □

Measurement and Estimation

We use the localization model laid out in (3.16)-(3.17) to perform state estimation for $\mathbf{x}^*_{BF}$ using a particle filter [189] implemented in the ParticleFilter class of the MATLAB Robotics toolbox [43]. We assume that the only measurement provided for the propagation of the particle filter is the distance of the robot to the target $d = \sqrt{(x^*_{BF})^2 + (y^*_{BF})^2}$ and use a measurement model of the form

$$y(t) = d(t) + \epsilon(t) \tag{3.18}$$

with $\epsilon(t)$ representing the measurement noise. We note here that various statistical distributions have been considered for $\epsilon(t)$ in the RF literature, but, consistent with other work [116], a Gaussian distribution with mean zero and a specified standard deviation according to the range sensor's characteristics was determined to be sufficient for our purposes.

By supplying an initial estimate for $\mathbf{x}^*_{BF}$, an initial estimate covariance Σ_0, suitable process noise estimates for the proprioceptive linear speed and yaw rate provided by the robot, a suitable measurement noise standard deviation and a proper number of particles (please refer to Section 3.4 for more details), the particle filter provides an estimate of the goal location $\mathbf{x}^*_{BF}$, which is constantly updated and gets better as the robot moves. Some tuning on the number of particles is required to balance between the needs for fast filter updates and the achievement of good convergence properties. It must be noted that the problem of beacon homing using RF sensors is worthy of independent study due to issues related to multipath interference etc., which go beyond the scope of this work.

Body Frame Navigation Algorithm

With the localization algorithm supplying an estimate of $\mathbf{x}^*_{BF} = (x^*_{BF}, y^*_{BF})$ at every control iteration already in place, we construct the homing behavior by writing the control law in (3.8) - (3.9) in the body frame and setting the inputs

$$v_d = k\, \bar{x}^*_{v,BF} \tag{3.19}$$

$$\omega_d = k \operatorname{atan}(\bar{y}^*_{\omega,BF} / \bar{x}^*_{\omega,BF}) \tag{3.20}$$

with

$$\bar{\mathbf{x}}^*_{v,BF} = (\bar{x}^*_{v,BF}, 0) := \left(\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}^\top \Pi_{\mathcal{LF}_v(\mathbf{x})}(\mathbf{x}^*_{BF}), 0 \right)$$

$$\bar{\mathbf{x}}^*_{\omega,BF} = (\bar{x}^*_{\omega,BF}, \bar{y}^*_{\omega,BF}) :=$$

$$\left(\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}^\top \left(\frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*_{BF}) + \Pi_{\mathcal{LF}(\mathbf{x})}(\mathbf{x}^*_{BF})}{2} \right), \right.$$

$$\left. \begin{bmatrix} -\sin\psi \\ \cos\psi \end{bmatrix}^\top \left(\frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*_{BF}) + \Pi_{\mathcal{LF}(\mathbf{x})}(\mathbf{x}^*_{BF})}{2} \right) \right)$$

the linear and angular projected goals respectively, and the local freespace $\mathcal{LF}$ as described in (3.5), computed from the local workspace $\mathcal{LW}$ in (3.15) and demonstrated in Fig. 3.8. In this way, we have constructed a minimalistic sensory-driven approach to the homing problem, that uses a LIDAR for obstacle avoidance and an RF sensor, providing only one-dimensional information (range), for the target location.

3.4 Experimental Setup

Here, we detail the ROS networked environment, in which Minitaur operates, that generates its high level ("unicycle-like") control inputs by implementing the reactive navigation algorithm summarized in either Section 3.2.1 (bounding) or Section 3.2.2 (walking). As shown

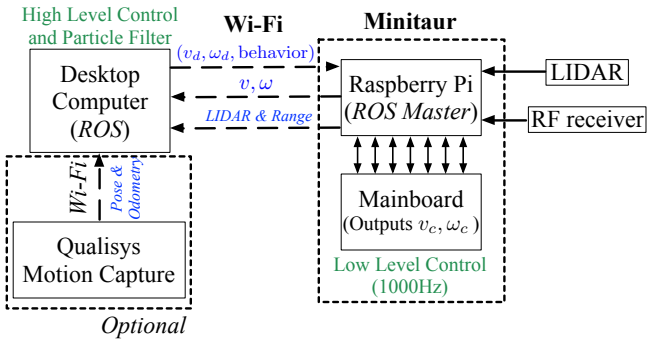

Figure 3.11: A schematic demonstrating the system structure of the experimental setup. Minitaur's Raspberry Pi, the central element of this configuration, acts as the ROS Master and forwards any LIDAR and range readings. The external computer runs the high level controller which gives the desired linear and angular velocities v_d, ω_d, while Minitaur's mainboard runs the low level controller by calculating the actual commands v_c, ω_c using (3.1), and provides proprioceptive speed and yaw rate feedback v, ω, forwarded to the desktop computer by the Raspberry Pi.

in Fig. 3.11, this environment consists of a computer implementing the high-level controller and of Minitaur's ROS infrastructure, exchanging messages over a Wi-Fi network.

In order to provide a hardware abstraction commensurate with the behavioral abstraction of Section 3.1.2, Minitaur's computational subsystem is enhanced with a Raspberry Pi Model 3, which is able to both run ROS and connect to a Wi-Fi access point. A custom ROS node on the Raspberry Pi receives (v_d, ω_d) and the desired mode of operation (bounding, walking, standing) as ROS messages (from the desktop computer) and forwards them to the Minitaur mainboard (microcontroller implementing the functionalities shown in (3.1) to produce the actual commands v_c, ω_c) at 100Hz over a 115.2 Kbps USART connection. The Raspberry Pi acts as the ROS Master that resolves networking for the rest of the ROS nodes: a dedicated ROS node is activated as soon as the system boots and automatically subscribes to the (v_d, ω_d) ROS topics, as well as an additional one capable of defining the desired behavior.

Bounding-Specific Infrastructure Components

In the case of bounding and in the absence of any onboard sensor, the odometry information consisting of the linear speed v and the yaw rate ω is extracted from a Qualisys Motion Capture System [154] (QMCS) at 100 Hz, using a set of motion capture cameras positioned around a 20m $\times$ 6m arena. The desktop computer receives the online data from QMCS

using the ROS package `mocap_qualisys` [114] and outputs specific desired linear and angular velocity values (v_d, ω_d) for Minitaur as described in Section 3.2.1. The high level control loop runs at approximately 100Hz, which is more than enough for the robot to recover if any obstacle is detected, and the low-level (bounding or walking trot) controller runs at 1KHz.

Walking-Specific Infrastructure Components

For the walking trot experiments, the setup is enhanced with two Pulson P-440 RF modules [190] (one beacon for the goal and one receiver for the robot), along with a Hokuyo UTM-30LX LIDAR [79].

Since the fully sensor-based navigation approach described in Sections 3.2.2 and 3.3 is used, a second ROS node reads the proprioceptive odometry feedback[3] from the mainboard and forwards it to the desktop computer for use in the particle filter propagation [189]. Also, a third node, adapted from the ROS library in [177], is responsible of sending the range measurements from the RF sensor to the desktop computer. A final ROS node, taken from [166], forwards the LIDAR measurements to the desktop computer.

The desktop computer is responsible for running the high-level control algorithm outlined in Section 3.2.2, along with the particle filter propagation for target localization, as described in Section 3.3. For the particle filter, we use a process noise of 0.2m/s for the linear speed and 0.4rad/s for the angular speed. Also, a range measurement noise (standard deviation) of 10cm is used, consistent with the Pulson P-440 RF module datasheet. We use 2000 particles, systematic resampling and an effective particle ratio of 0.8.

As we show in Section 3.5 and the accompanying video of [201][4], this infrastructure works robustly and without any discernible network-induced latency. The high level control loop here is slower and runs at approximately 50Hz, since several sensor readings have to be sent and processed, but this frequency is still more than enough for the robot to recover if any obstacle is detected.

[3] Here the forward speed v is estimated with the use of leg kinematics as shown in [100] and ω is provided by a VN-100 IMU [207].

[4] `https://youtu.be/kJM0fSxxL9k`

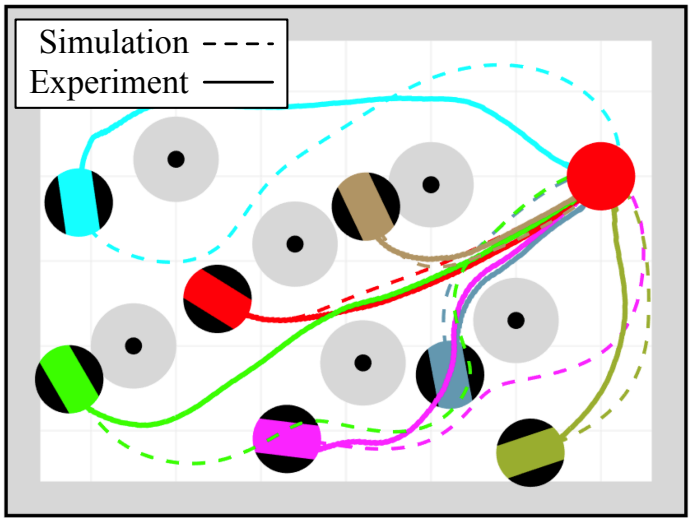

Figure 3.12: Trajectories extracted from simulations and bounding experiments in a small and dense with obstacles environment. The goal position (shown as a solid red disk) is fixed but initial robot configurations vary.

3.5 Experimental Results

We illustrate the qualitative features and performance of the navigation algorithm by presenting empirical results for both the bounding and the walking trot gait. Section 3.5.1 reports on experiments run using the bounding gait and local but "bird's eye" information as described in Section 3.2.1 in environments cluttered with disk-shaped obstacles, and Section 3.5.2 describes the results in a similar workspace with the walking gait, the use of the sensor-based navigation algorithm described in Section 3.2.2 and the range-only target localization scheme presented in Section 3.3. In all of our experiments, Minitaur is approximated as a disk-shaped robot with radius[5] 0.4 m, and a margin of 0.1 m is added to the robot's radius for safety reasons.

3.5.1 Bounding Experiments

Fig. 3.12 depicts our results in a small and obstacle-dense environment, and Fig. 3.13 depicts our results in a less dense with obstacles but larger arena. The obstacles have common radius $\rho = 0.1$ m and are randomly placed throughout the environment. The goal position is near

[5]Minitaur's length (hip-to-hip) is 0.4 m and an extra length of 0.4 m due to fore and hind leg extensions in the sagittal plane (typically about 0.2 m) has to be accounted for.

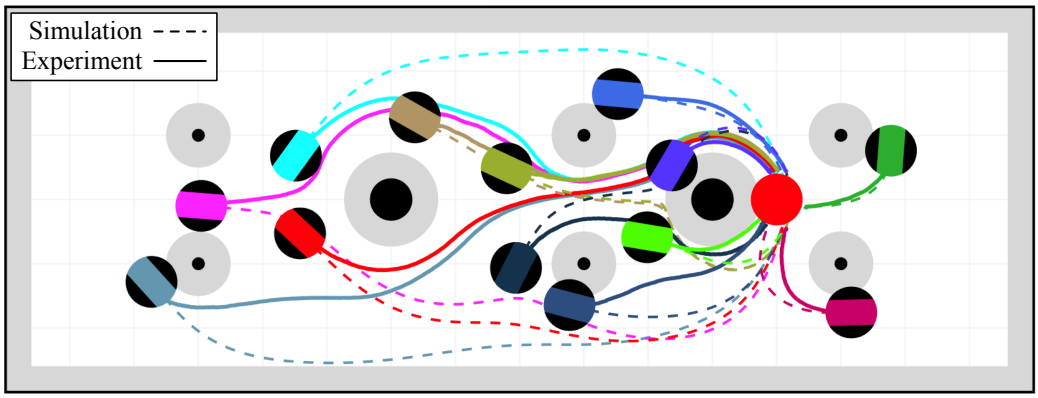

Figure 3.13: Trajectories extracted from simulations and bounding experiments in a large and less dense with obstacles environment. The goal position (shown as a solid red disk) is fixed but initial robot configurations vary.

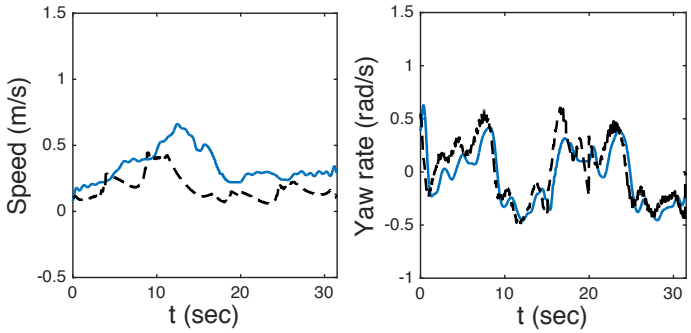

Figure 3.14: Minitaur's response (blue) to speed and yaw reference signals (black) during a bounding experimental trial.

the top right corner of the workspace behind several obstacles.

As it is evident from Fig. 3.12 and Fig. 3.13, Minitaur manages to converge to the desired location from a variety of initial configurations. In a total of over 50 trials, Minitaur reaches the goal and avoids all the obstacles each time. In both cases, we also overlay trajectories from MATLAB simulations of a differential-drive robot with the same initial conditions and similar control gains. The simulation and physical platform follow similar trajectories in most of the considered cases. It is important, though, that even when the trajectory is quite different, the robot always safely navigates to the goal location.

This is the most clear demonstration of the benefits provided by memoryless reactive planners; although slight perturbations and modeling errors[6] result in large perturbations

[6]Minitaur is only an imperfect kinematic unicycle, as discussed in Section 3.1.

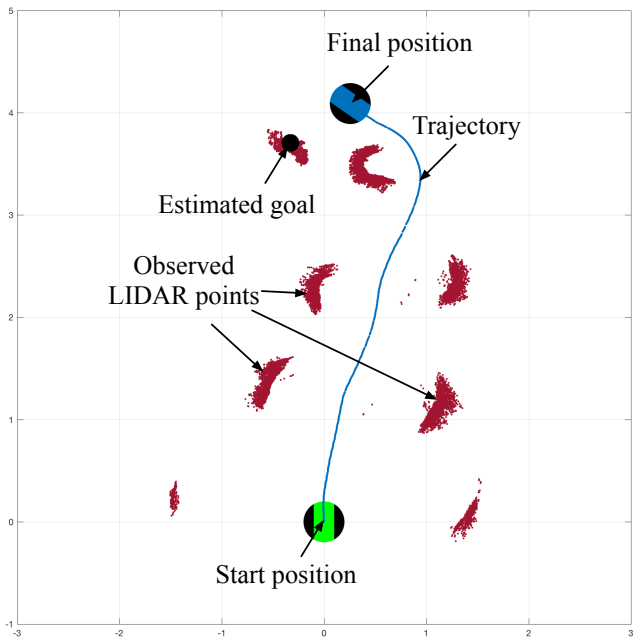

Figure 3.15: A suggestive path reconstructed from Minitaur's proprioceptive data in the environment shown in Fig. 3.9. The black dot corresponds to the (converged) estimated goal location at the end of the trial. The brown points consist the corresponding pointcloud of observed obstacle points; in the absence of ground-truth their exact location cannot be precisely determined.

to the final trajectory, the generated vector field guarantees global navigation to the goal without collisions. The navigation trajectory is neither known nor required a priori for guaranteed safety and task completion.

Finally, to illustrate Minitaur's bounding performance as a kinematic unicycle (Section 3.1.2), we plot in Fig. 3.14 Minitaur's response to the commanded fore-aft and yaw speeds during an experimental trial. Similarly to Section 3.1, we use a 3Hz cutoff filter to remove periodicities caused by Minitaur's bounding. As can be seen, the yaw response is quite good, but the speed tracking (while yawing) is less accurate. However, as demonstrated in Fig. 3.12, perfect speed tracking is not crucial for the reactive scheme presented here. The "zero yaw" trials of Fig. 3.4 reveal partly the reasons for this (the large amount of lag induced by smoothing in (3.1)). We also believe that the laggy speed response is partly responsible for the differences between the actual and simulated trajectories in our trials.

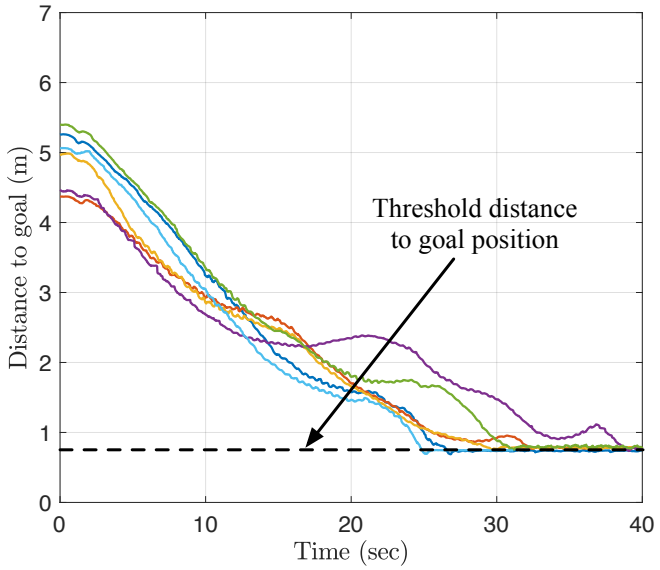

Figure 3.16: The distance to the goal position as a function of time for several initial conditions with the walking trot gait. In every case, the robot was commanded to stop as soon as it got within a distance of 0.8m from the target position.

3.5.2 Sensor-Based Walking Experiments

As mentioned in Section 3.1, the change from bounding to walking allows for the use of the fully sensor-based algorithm described in Sections 3.2.2 and 3.3. Fig. 3.15 depicts our results in a workspace cluttered with obstacles of common radius $\rho = 0.1$ m and randomly placed throughout the environment. Because of the lack of a portable ground-truth mechanism, the path shown in Fig. 3.15 was obtained by numerically integrating all the saved proprioceptive linear speed v and yaw rate ω estimates, and is thus suggestive but not exact. This also explains the non-convex shape of the observed "obstacles" in the workspace, reconstructed from the union of all the LIDAR readings. From this figure, it is evident that the robot managed to successfully localize the target, navigate there and stop within a predefined distance from it. In the absence of ground-truth, we plot in Fig. 3.16 the range measurements obtained by the RF sensor for several trials, showing convergence to the target. Finally, to illustrate Minitaur's walking performance as a kinematic unicycle (Section 3.1.3), we plot in Fig. 3.17 its response to commanded fore-aft and yaw speeds during an experimental trial.

It must be emphasized that we never experienced a failure to achieve the goal nor any

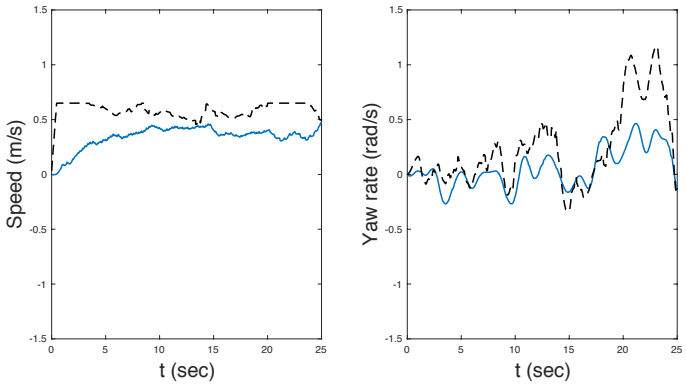

Figure 3.17: Minitaur's response (blue) to speed and yaw reference signals (black) during a walking trot experimental trial.

collisions along the way in any of the experiments conducted for this work. In the accompanying video[4], we demonstrate several of the numerous successful experimental trials for both bounding and walking, with the robot indefatigably seeking the goal.

3.6 Discussion

This Chapter demonstrates the empirical anchoring of a kinematic unicycle model on the dynamically complicated bounding and walking trot gaits of a quadrupedal robot and the robustness and efficiency of a sensor-based doubly reactive homing scheme, as an example of a high level motion planning strategy for legged robots. The realization of this algorithm in a GPS-denied environment is largely enabled by a proposed body-frame, range-only target localization algorithm which uses one-dimensional, range information to estimate the goal position in the body frame. The empirical results to date are very promising: the robot is driven to the desired goal location from any initial position and configuration in the workspace, while avoiding obstacles.

Part II

Mobile Manipulation in Partially Known Environments

Chapter 4

Reactive Symbolic Planning Using a Hierarchical Control Structure

This Chapter considers the problem of completing assemblies of passive objects in non-convex environments, cluttered with convex obstacles of unknown position, shape and size that satisfy a specific separation assumption. A differential drive robot equipped with a gripper and a LIDAR sensor, capable of perceiving its environment only locally, is used to position the passive objects in a desired configuration. The method combines the virtues of a deliberative planner generating high-level, symbolic commands, with the formal guarantees of convergence and obstacle avoidance of a reactive planner that requires little onboard computation and is used online. The validity of the proposed method is verified both with formal proofs and numerical simulations.

The Chapter is organized as follows. Section 4.1 describes the problem and summarizes our approach. Section 4.2 gives a brief outline of the high-level deliberative planner that generates the sequence of appropriate symbolic commands to accomplish the task at hand, without any information about the internal obstacles. Section 4.3 describes the fundamental idea of reactively switching between a path following and a wall following mode, for both a holonomic and a nonholonomic robot, while Section 4.4 extends our reactive ideas to the navigation problem of a nonholonomic robot grasping a passive object and using its sensor

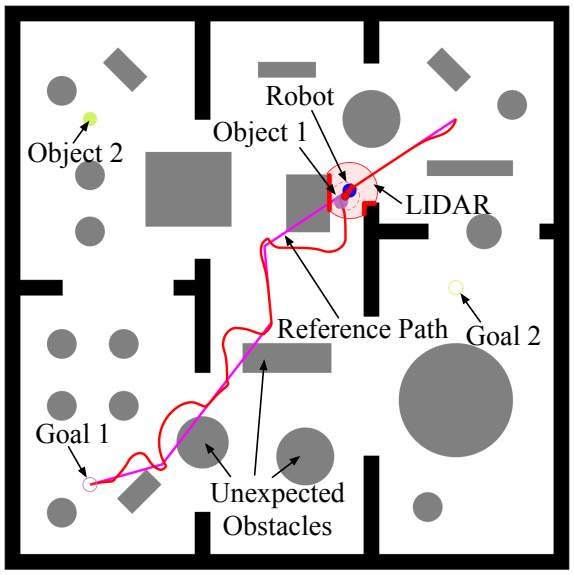

Figure 4.1: A depiction of an intermediate stage of an assembly process. The robot is tasked to move two objects from their start to their final configuration using a gripper and a LIDAR. The deliberative planner outputs a reference path (purple) which the reactive planner has to follow, while avoiding the unexpected obstacles (grey) in the (potentially) non-convex workspace. The resulting piecewise differentiable object trajectory for one object is shown in red.

to position it at a desired location. Section 4.5 combines the ideas from the previous two sections and describes the low-level, online implementation of the symbolic action command set. Finally, Section 4.6 presents illustrative numerical examples for the ideas presented.

4.1 Problem Formulation

We consider a first-order, nonholonomically-constrained, disk-shaped robot, centered at $\mathbf{x} \in \mathbb{R}^2$ with radius $r \in \mathbb{R}_{>0}$ and orientation $\psi \in S^1$, using a gripper to move circular objects in a closed, compact, not necessarily convex workspace $\mathcal{W} \subset \mathbb{R}^2$ as shown in Fig. 4.1, whose boundary $\partial\mathcal{W}$ is assumed to be known. The robot dynamics are described by

$$(\dot{\mathbf{x}}, \dot{\psi}) = \mathbf{B}(\psi)\mathbf{u}_{ku} \tag{4.1}$$

with $\mathbf{B}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$ the differential constraint matrix and $\mathbf{u}_{ku} = (v, \omega)$ the input vector[1] consisting of a linear and an angular command. The robot is assumed to possess a LIDAR, positioned at $\mathbf{x}$, with a 360° angular scanning range and a fixed sensing range $R \in \mathbb{R}_{>0}$ and is tasked with moving each of the $n \in \mathbb{N}$ movable disk-shaped objects, centered at $\mathbf{p} := (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n) \in \mathcal{W}^n$ with a vector of radii $(\rho_1, \rho_2, \ldots, \rho_n) \in (\mathbb{R}_{>0})^n$, from its initial configuration to a user-specified goal configuration $\mathbf{p}^* := (\mathbf{p}_1^*, \mathbf{p}_2^*, \ldots, \mathbf{p}_n^*) \in \mathcal{W}^n$. We assume that both the initial configuration and the target configuration of the objects are known. In addition to the known boundary of the workspace $\partial\mathcal{W}$, the workspace is cluttered by an unknown number of fixed, disjoint, convex obstacles of unknown position and size, denoted by $\mathcal{O} := (O_1, O_2, \ldots)$. To simplify the notation, also define $\mathcal{O}_w := \mathcal{O} \cup \partial\mathcal{W}$.

We adopt the following assumptions to guarantee that any robot-object pair can go around any obstacle in the workspace along any possible direction, introduced only to facilitate the proofs of our formal results, without being necessary for the existence of some solution to the problem.

Assumption 4.1 (Obstacle separation)**.** *The obstacles $\mathcal{O}$ in the workspace are separated from each other by clearance*[2] *of at least $d(O_i, O_j) > 2(r + \max_k \rho_k), i \neq j$, with k an index spanning the set of movable objects. They are also separated from the boundary of the (potentially non-convex) workspace $\mathcal{W}$ by at least $d(O_i, \partial\mathcal{W}) > 2(r + \max_k \rho_k)$ for all i.*

Assumption 4.1 means that there exists $\eta \in \mathbb{R}_{>0}$ such that

$$\eta = \min \left\{ \min_{\substack{i,j \\ i \neq j}} d(O_i, O_j), \min_i d(O_i, \partial\mathcal{W}) \right\} \tag{4.2}$$

and $\eta > 2(r + \max_k \rho_k)$.

Also, in order to ensure successful positioning of all the objects to their target configuration using reactive control schemes, it is convenient to impose a further constraint on how

[1] Throughout this work, we will use the ordered set notation $(*, *, \ldots)$ and the matrix notation $\begin{bmatrix} * & * & \ldots \end{bmatrix}^T$ for vectors interchangeably.

[2] Here the clearance between two sets A and B is defined as $d(A, B) := \inf\{\|\mathbf{a} - \mathbf{b}\| \mid \mathbf{a} \in A, \mathbf{b} \in B\}$

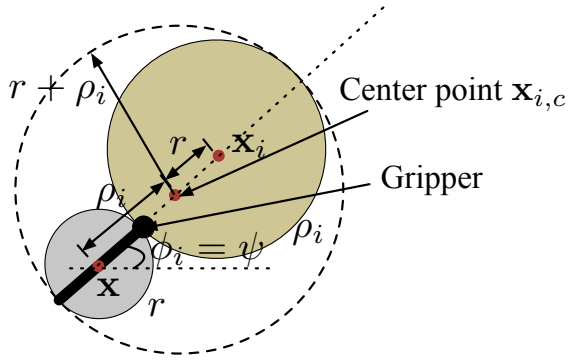

Figure 4.2: A depiction of a disk-shaped robot with radius r (grey) moving a disk-shaped object with radius ρ_i (yellow).

tightly packed the desired goal configuration can be.

Assumption 4.2 (Admissible object goals)**.** *For any object* $i \in \{1, \ldots, n\}$, $d(\mathbf{p}_i^*, \mathcal{O}_w) > \rho_i + 2r$.

The robot's gripper can either be engaged or disengaged; we will write $g = 1$ when the gripper is engaged and $g = 0$ when it is disengaged.

In order to accomplish the task of bringing every object to its designated goal position, we endow the deliberative planner with a set of three symbolic output action commands:

- MoveToObject$(i, \mathcal{P})$ instructing the robot to move and grasp the object i along the piecewise continuously differentiable path $\mathcal{P} : [0, 1] \to \mathcal{W}$ such that $\mathcal{P}(0) = \mathbf{x}$ and $\mathcal{P}(1) = \mathbf{p}_i$.

- PositionObject$(i, \mathcal{P})$ instructing the robot to push the (assumed already grasped) object i toward its designated goal position, $\mathbf{p}_i^*$, along the piecewise continuously differentiable path $\mathcal{P} : [0, 1] \to \mathcal{W}$ such that $\mathcal{P}(0) = \mathbf{p}_i$ and $\mathcal{P}(1) = \mathbf{p}_i^*$.

- Move$(\mathcal{P})$ instructing the robot to move along the piecewise continuously differentiable path $\mathcal{P} : [0, 1] \to \mathcal{W}$ such that $\mathcal{P}(0) = \mathbf{x}$.

This symbolic command set, comprising the interface between the deliberative and reactive components of our planner enforces the following problem decomposition into the complementary pair:

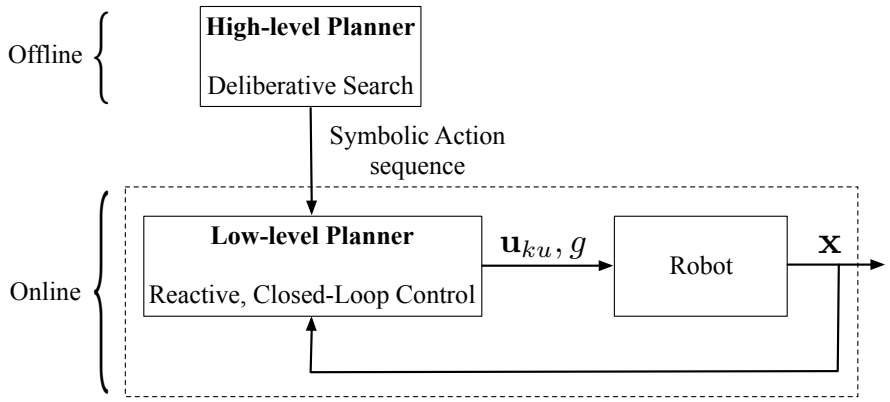

Figure 4.3: An outline of the control approach followed in order to position the objects. A high-level, deliberative planner outputs a sequence of symbolic actions that are realized and executed sequentially in low-level using a reactive controller. The architecture follows Fig. 1.1 without including the interface layer, since it is assumed that each provided symbolic action from the high-level planner is always feasible, and the (platform-specific) gait layer, since the presentation in this Chapter is limited to differential drive robots equipped with a gripper.

1. Find a *symbolic plan*, i.e., a sequence of symbolic actions whose successful implementation is guaranteed to complete the task.

2. Implement each of the symbolic actions using the appropriate commands $\mathbf{u}_{ku}$ according to the robot's equations of motion shown in (4.1), while avoiding the perceived unanticipated by the deliberative planner obstacles.

Fig. 4.3 depicts this problem decomposition and the associated interface between the deliberative and reactive components of our architecture.

4.2 Deliberative Planner

In order to obtain plans suitable for the reactive planner to track, we use a high-level planner that combines the factored orbital random geometric graph (FORGG) construction [208] with the approximate angelic A* (AAA*) search algorithm [210]. FORGG extends the asymptotic optimality guarantees of the PRM* algorithm to problems involving discontinuous differential constraints like contact and object manipulation. Searching this planning graph using conventional methods like A* is computationally expensive, due to the size of search space. To facilitate efficient search, we employ the angelic semantics developed by Marthi *et al.* [128] to encode bounds on the possible cost of sets of possible plans. AAA*

uses these bounds to guide the search, allowing large parts of the search space to be pruned away and accelerating the search for a near-optimal high-level plan.

With this construction, the deliberative planner is supplied with the initial position and size of the robot and the objects to be placed, along with any information assumed to be known (boundary of the workspace, walls, interior obstacles etc.) and outputs a series of symbolic action commands (MoveToObject, PositionObject, Move) each associated with a collision-free path $\mathcal{P}$ in order to accomplish the task at hand.

4.3 Reactive Planning for Single Robots

In this Chapter we describe the (low-level) reactive algorithms which guarantee collision avoidance and (almost) global convergence[3] to the plan provided by the (high-level) deliberative planner, described in Section 4.2. First, we focus on the navigation problem of a single (fully actuated or nonholonomically-constrained) robot, using tools from [6] and [7], and we will show in Section 4.4 how to extend these principles for the case of gripping contact.

4.3.1 Doubly-Reactive Planner for Holonomic Robots

First we consider a fully actuated disk-shaped robot centered at $\mathbf{x} \in \mathbb{R}^2$ with radius $r > 0$, moving in a closed-convex environment (denoted by $\mathcal{W} \subset \mathbb{R}^2$) towards a goal location $\mathbf{x}^* \in \mathbb{R}^2$. Although we use a differential drive robot for our assembly problem here, we find it useful to present the basic algorithm for fully actuated robots, especially since it will be used in Section 4.4. The robot dynamics are assumed to be described by

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}) \tag{4.3}$$

[3]It is well-known that the basin of a point attractor in a non-contractible space must exclude a set of measure zero [104].

with $\mathbf{u} \in \mathbb{R}^2$ the input. The sensory measurement of the LIDAR at $\mathbf{x} \in \mathcal{W}$ is modeled as in [7] by a polar curve $\rho_{\mathbf{x}} : (-\pi, \pi] \to [0, R]$ as follows[4] (see (3.14) in Chapter 3)

$$\rho_{\mathbf{x}}(\theta) := \min \begin{pmatrix} R \\ \min \{\|\mathbf{p} - \mathbf{x}\| \,|\, \mathbf{p} \in \partial\mathcal{W}, \text{atan2}(\mathbf{p} - \mathbf{x}) = \theta\} \\ \min_i \{\|\mathbf{p} - \mathbf{x}\| \,|\, \mathbf{p} \in O_i, \text{atan2}(\mathbf{p} - \mathbf{x}) = \theta\} \end{pmatrix} \tag{4.4}$$

We will also use the definitions of free space $\mathcal{F}$, line-of-sight local workspace $\mathcal{LW}_{\mathcal{L}}(\mathbf{x})$ and line-of-sight local free space $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$ at $\mathbf{x}$ from [7] (given in (3.15) and (3.5) in Chapter 3).

Under the preceding definitions, it is shown in [7] that the control law

$$\mathbf{u}(\mathbf{x}) = -k \left(\mathbf{x} - \Pi_{\mathcal{LF}_{\mathcal{L}}(\mathbf{x})}(\mathbf{x}^*)\right), k \in \mathbb{R} \tag{4.5}$$

asymptotically drives almost all configurations in $\mathcal{F}$ to the goal $\mathbf{x}^*$ while avoiding obstacles and not increasing the Euclidean distance to the goal along the way.

4.3.2 Reactive Path Following

For a fixed goal $\mathbf{x}^*$, the reactive control law in (4.5) guarantees convergence only for convex workspaces (punctured by obstacles).

Therefore, inspired by [8], we apply the idea from Section 4.3.1 to the problem of a robot following a navigation path $\mathcal{P} : [0, 1] \to \mathring{\mathcal{F}}$, that joins a pair of initial and final configurations $\mathbf{x}^0, \mathbf{x}^1 \in \mathring{\mathcal{F}}$ in a potentially non-convex workspace and lies in the interior of the free space, i.e., $\mathcal{P}(0) = \mathbf{x}^0, \mathcal{P}(1) = \mathbf{x}^1$ and $\mathcal{P}(\alpha) \in \mathring{\mathcal{F}}, \forall \alpha \in [0, 1]$.

As demonstrated in [8], the *projected-path goal* $\mathcal{P}(\alpha^*)$ with α^* determined as[5]

$$\alpha^* = \max\{\alpha \in [0, 1] \,|\, \mathcal{P}(\alpha) \in \mathsf{B}\left(\mathbf{x}, d(\mathbf{x}, \partial\mathcal{F})\right)\} \tag{4.6}$$

replaces $\mathbf{x}^*$ in (4.5) as the target goal position and is constantly updated as the agent moves along the path. Note that in the LIDAR-based setting presented here, the distance of the

[4] See [7] for a discussion on the choice of LIDAR range R to avoid obstacle occlusions.
[5] Here $\mathsf{B}(\mathbf{q}, t) := \{\mathbf{p} \in \mathcal{W} \,|\, ||\mathbf{p} - \mathbf{q}|| \leq t\}$, i.e., the ball of radius t centered at $\mathbf{q}$.

agent from the boundary of the free space $d(\mathbf{x}, \partial\mathcal{F})$ can easily be determined as

$$d(\mathbf{x}, \partial\mathcal{F}) = \min_{\theta} \rho_{\mathbf{x}}(\theta) - r \tag{4.7}$$

4.3.3 Reactive Wall Following

As described in Section 4.1 and shown in Fig. 4.1, the path $\mathcal{P}$ might not lie in the free space since the deliberative planner is only aware of the boundary of the workspace and not of the position or size of the internal obstacles. For this reason, we present here a novel control law for reactive wall following, inspired from the "bug algorithm" [39], that exhibits desired formal guarantees.

The wall following law is triggered by saving the current index α_s^* of the path $\mathcal{P}$ when the distance of the agent from the boundary of its free space, given in (4.7), drops below a small critical value ϵ, i.e., when $d(\mathbf{x}, \partial\mathcal{F}) < \epsilon$. This would imply that the robot enters a "danger zone" within the vicinity of an unexpected obstacle. The goal now would be to follow the boundary of that obstacle without losing it, in order to find the path again.

Therefore, the robot first needs to select a specific direction to consistently follow the boundary of the obstacle along that direction. Since our problem is planar, there are only two possible direction choices: clockwise (CW) or counterclockwise (CCW). Also, since the robot has only local information about the obstacle based on the current LIDAR readings, a greedy selection of the wall following direction is necessary.

Let $\theta_m \in (-\pi, \pi]$ be the LIDAR angle such that $\rho_{\mathbf{x}}(\theta_m) = \min_{\theta} \rho_{\mathbf{x}}(\theta)$ corresponds to the minimum distance from the blocking obstacle. Let $\mathbf{n}_w(\mathbf{x}) := -(\cos\theta_m, \sin\theta_m)$ denote the normal vector to the boundary of the obstacle at the point of minimum distance and $\mathbf{t}_w(\mathbf{x}) = \mathbf{J}\,\mathbf{n}_w(\mathbf{x})$, with $\mathbf{J} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, the corresponding tangent vector.

Our proposed method uses the inner product $\mathbf{t}_{w,0} \cdot \mathbf{t}_{\mathcal{P}}(\alpha_s^*)$, with $\mathbf{t}_{w,0}$ denoting the tangent vector to the boundary of the obstacle at the beginning of the wall following phase and $\mathbf{t}_{\mathcal{P}}(\alpha_s^*)$ the tangent vector of the path $\mathcal{P}$ at α_s^*. Then, the value of a variable a is set to 1 for CCW

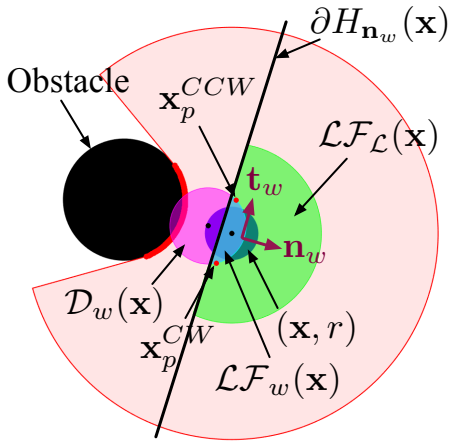

Figure 4.4: An example of computing the wall following local free space $\mathcal{LF}_w(\mathbf{x})$ (cyan) as the intersection of the local free space $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$ (green) and the offset disk $\mathcal{D}_w$ (magenta) for a robot with radius r positioned at $\mathbf{x}$, encountering an obstacle within its LIDAR footprint $L_{ft}(\mathbf{x})$ (red).

motion and to -1 for CW motion (fixed for all future time) according to

$$a = \begin{cases} 1, & \text{if} \quad \mathbf{t}_{w,0} \cdot \mathbf{t}_{\mathcal{P}}(\alpha_s^*) \geq 0 \\ -1, & \text{if} \quad \mathbf{t}_{w,0} \cdot \mathbf{t}_{\mathcal{P}}(\alpha_s^*) < 0 \end{cases} \tag{4.8}$$

since $\mathbf{t}_w(\mathbf{x})$ has counterclockwise direction around the obstacle by construction.

Define the *offset disk at* $\mathbf{x}$

$$\mathcal{D}_w(\mathbf{x}) := \{\mathbf{p} \in \mathcal{W} \,|\, ||\mathbf{p} - \mathbf{x}_{\text{offset}}(\mathbf{x})|| \leq \epsilon\} \tag{4.9}$$

with ϵ selected according to Assumption 4.1 to satisfy

$$0 < \epsilon < \frac{1}{2}\left[\eta - 2(r + \max_j \rho_j)\right] \tag{4.10}$$

with η given in (4.2) and

$$\mathbf{x}_{\text{offset}}(\mathbf{x}) := \mathbf{x} - (\rho_{\mathbf{x}}(\theta_m) - r)\,\mathbf{n}_w(\mathbf{x}) \tag{4.11}$$

Then define the *wall following local free space* $\mathcal{LF}_w(\mathbf{x})$ as *at* $\mathbf{x}$

$$\mathcal{LF}_w(\mathbf{x}) := \mathcal{LF}_{\mathcal{L}}(\mathbf{x}) \cap \mathcal{D}_w(\mathbf{x}) \tag{4.12}$$

Since $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$ is convex [7], $\mathcal{LF}_w(\mathbf{x})$ is convex as the intersection of convex sets.

The *wall following law* is then given as

$$\mathbf{u}(\mathbf{x}) = -k\left(\mathbf{x} - \mathbf{x}_p(\mathbf{x})\right) \tag{4.13}$$

with

$$\mathbf{x}_p(\mathbf{x}) := \mathbf{x}_{\text{offset}}(\mathbf{x}) + \frac{\epsilon}{2}\mathbf{n}_w(\mathbf{x}) + a\frac{\epsilon\sqrt{3}}{2}\mathbf{t}_w(\mathbf{x}) \tag{4.14}$$

Lemma 4.1. *If* $\mathbf{x} \in \mathcal{F}$ *and* $d(\mathbf{x}, \partial\mathcal{F}) < \epsilon$ *with* ϵ *chosen according to* (4.10)*:*

(i) The wall following free space $\mathcal{LF}_w(\mathbf{x})$ *contains* $\mathbf{x}$ *in its interior.*

(ii) $\mathcal{LF}_w(\mathbf{x}) = \mathcal{D}_w(\mathbf{x}) \cap H_{\mathbf{n}_w}(\mathbf{x})$ *with* $H_{\mathbf{n}_w}(\mathbf{x})$ *the half space*

$$H_{\mathbf{n}_w}(\mathbf{x}) = \{\mathbf{p} \in \mathcal{W} \,|\, (\mathbf{p} - \mathbf{x}_h(\mathbf{x})) \cdot \mathbf{n}_w \geq 0\}$$

and $\mathbf{x}_h(\mathbf{x}) = \mathbf{x} - \frac{1}{2}(\rho_{\mathbf{x}}(\theta_m) - r)$.

(iii) The point $\mathbf{x}_p(\mathbf{x})$ *lies on the boundary of* $\mathcal{LF}_w(\mathbf{x})$.

Proof. Included in Appendix C.2. □

Proposition 4.1. *With the choice of* ϵ *in* (4.10)*, the wall following law in* (4.13) *has the following properties:*

(i) It is piecewise continuously differentiable.

(ii) It generates a unique continuously differentiable flow, defined for all future time.

(iii) It has no stationary points.

(iv) The free space $\mathcal{F}$ *is positively invariant under its flow.*

(v) Moreover, the set $\left\{\mathbf{p} \in \mathcal{W} \,\middle|\, \frac{\epsilon}{2} < d(\mathbf{p}, \partial\mathcal{F}) < \epsilon\right\}$ *is positively invariant under its flow.*

Proof. Included in Appendix C.2. □

We find it useful to include the following definition

Definition 4.1. *The rate of progress along the boundary of the observed obstacle at* $\mathbf{x}$ *is defined as*

$$\sigma(\mathbf{x}) := \frac{\mathbf{u}(\mathbf{x}) \cdot \mathbf{t}_w(\mathbf{x})}{||\mathbf{u}(\mathbf{x})||} \tag{4.15}$$

By combining all these results, we arrive at the following Theorem.

Theorem 4.1. *With a selection of* ϵ *as in* (4.10)*, the wall following law in* (4.13) *has no stationary points, leaves the robot's free space* $\mathcal{F}$ *positively invariant under its unique continuously differentiable flow, and steers the robot along the boundary of a unique obstacle in* $\mathcal{O}$ *in a clockwise or counterclockwise fashion (according to the selection of a in* (4.8)*) with a nonzero rate of progress* σ*, while maintaining a distance of at most* $(r + \epsilon)$ *and no less than* $\left(r + \frac{\epsilon}{2}\right)$ *from it.*

In order to prove the theorem, we will make use of the following Proposition.

Proposition 4.2. *Let* $\mathbf{x}^t$ *denote the robot position at time* t*, with* $t = 0$ *corresponding to the beginning of the wall following phase. Suppose that the flow* $\mathbf{x}^t$ *is continuous,* $k = \arg\min_i d(\mathbf{x}^0, O_i)$ *with* $O_i \in \mathcal{O}$ *and* ϵ *satisfies* (4.10)*. Then* $d(\mathbf{x}^t, \partial\mathcal{F}) < \epsilon$ *implies* $k = \arg\min_i d(\mathbf{x}^t, O_i)$ *for all* $t > 0$*.*

Proof. Included in Appendix C.2. □

Proof of Theorem 4.1. Included in Appendix C.2. □

The robot exits the wall following mode and returns to the path following mode once it encounters the path again, i.e., when $\alpha^* = \max\{\alpha \in [0,1] \,|\, \mathcal{P}(\alpha) \in \mathsf{B}\left(\mathbf{x}, d(\mathbf{x}, \partial\mathcal{F})\right)\} > \alpha_s^*$. An immediate Corollary of Theorem 4.1, along with path continuity of $\mathcal{P}$ and Assumptions 4.1 and 4.2 is the following:

Corollary 4.1. *If the robot enters the wall following mode, it will exit it in finite time and return to the path following mode.*

Finally, since both the path following law [8] and the wall following law generate continuously differentiable flows, we find it useful to explicitly state the following result, in the sense of sequential composition [35].

Theorem 4.2. *In a workspace where Assumption 4.1 is satisfied, any composition of path following and wall following phases generates a unique piecewise continuously differentiable flow for* $\mathbf{x}$*, defined for all future time.*

4.3.4 Extension to Nonholonomic Robots

As shown in [7] and Section 3.2, the preceding results can easily be extended for the case of a differential-drive robot driving towards a goal $\mathbf{x}^*$, whose dynamics are given in (4.1). Here, we will use a slightly different control law since the robot possesses a gripper and must only move in the forward direction to grasp objects. The following inputs are used

$$v = \max\left\{-k\begin{bmatrix}\cos\psi\\ \sin\psi\end{bmatrix}^T\left(\mathbf{x} - \Pi_{\mathcal{LF}_v(\mathbf{x})}(\mathbf{x}^*)\right), 0\right\} \tag{4.16}$$

$$\omega = -k\,\mathrm{atan2}\left(\beta_2, \beta_1\right) \tag{4.17}$$

with

$$\beta_1 = \begin{bmatrix}\cos\psi\\ \sin\psi\end{bmatrix}^T\left(\mathbf{x} - \frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*) + \Pi_{\mathcal{LF}_\mathcal{L}(\mathbf{x})}(\mathbf{x}^*)}{2}\right) \tag{4.18}$$

$$\beta_2 = \begin{bmatrix}-\sin\psi\\ \cos\psi\end{bmatrix}^T\left(\mathbf{x} - \frac{\Pi_{\mathcal{LF}_\omega(\mathbf{x})}(\mathbf{x}^*) + \Pi_{\mathcal{LF}_\mathcal{L}(\mathbf{x})}(\mathbf{x}^*)}{2}\right) \tag{4.19}$$

in order to constrain the robot motion to forward only and align with the desired target respectively. Here $\mathcal{LF}_v(\mathbf{x}), \mathcal{LF}_\omega(\mathbf{x})$ are used as in [7] (see (3.10) and (3.11)).

Based on the preceding analysis, for a differential drive robot, we will use $\mathbf{x}^* = \mathcal{P}(\alpha^*)$

(with α^* shown in (4.6)) in the path following mode and $\mathbf{x}^* = \mathbf{x}_p(\mathbf{x})$ in the wall following mode. The following Theorem summarizes the qualitative properties of the wall following law for differential drive robots.

Theorem 4.3. *With a selection of ϵ as in* (4.10)*, the unicycle wall following law in* (4.16)*,* (4.17) *with $\mathbf{x}^* = \mathbf{x}_p(\mathbf{x})$ as in* (4.14) *leaves the robot's free space $\mathcal{F}$ positively invariant under its unique continuously differentiable flow, aligns the robot with $a\,\mathbf{t}_w(\mathbf{x})$ (according to the selection of a in* (4.8)*) in finite time and steers the robot along the boundary of a unique obstacle in $\mathcal{O}$ in a clockwise or counterclockwise fashion (depending on a) with a nonzero rate of progress σ afterwards, while maintaining a distance of at most $(r + \epsilon)$ from it.*

Proof sketch. Included in Appendix C.2. □

We summarize the proposed method for switching between a path following and a wall following phase and generating velocity commands for a differential drive robot following a reference path $\mathcal{P}$ in Algorithm 4.1, with the definition of an auxiliary symbolic action NavigateRobot$(\mathcal{P}, r, \epsilon, \delta)$.

4.4 Reactive Planning for Gripping Contact

In this Section, we describe a method for generating suitable motion commands online for two objects in contact, of which one is a differential drive robot and uses a gripper to push the other, passive object on the plane. Our method consists of generating "virtual" commands for different points of interest in the robot-object pair and translating them to "actual" commands for the robot using simple kinematic maps.

4.4.1 Gripping Contact Kinematics

Consider the robot gripping an object i, as shown in Fig. 4.2. We can find the position of the object center of mass $\mathbf{x}_i \in \mathcal{W}$ from the position of the robot center of mass $\mathbf{x}$ as

$$\mathbf{x}_i := \mathbf{x} + (\rho_i + r)\,\mathbf{e}_{\|} \tag{4.20}$$

Algorithm 4.1 Generating velocity commands for a nonholonomic robot with radius r following a reference path $\mathcal{P}$.

function NAVIGATEROBOT($\mathcal{P}, r, \epsilon, \delta$)
  `mode` $\leftarrow$ `PathFollowing` ▷ Initialize `mode`
  do
    $\mathbf{x} \leftarrow$ Read Robot State
    $\psi \leftarrow$ Read Robot Orientation
    $\rho_{\mathbf{x}} \leftarrow$ Read LIDAR
    $d \leftarrow \min_{\theta} \rho_{\mathbf{x}}(\theta) - r$
    $\alpha^* \leftarrow$ Find maximum path index ▷ (4.6)
    if `mode` = `PathFollowing` **then**
      $\mathcal{LF}_{\mathcal{L}}(\mathbf{x}) \leftarrow$ Find local free space ▷ (3.5)
      $\mathbf{x}^* \leftarrow \mathcal{P}(\alpha^*)$
      if $d < \epsilon$ **then**
        `mode` $\leftarrow$ `WallFollowing`
        $\alpha_s^* \leftarrow \alpha^*$
        $a \leftarrow$ Find wall following direction ▷ (4.8)
      end if
    else if `mode` = `WallFollowing` **then**
      $\theta_m \leftarrow \arg\min_{\theta} \rho_{\mathbf{x}}(\theta)$
      $\mathbf{n}_w \leftarrow -(\cos\theta_m, \sin\theta_m)$
      $\mathbf{t}_w \leftarrow (\sin\theta_m, -\cos\theta_m)$
      $\mathbf{x}_{\text{offset}} \leftarrow \mathbf{x} - (\rho_{\mathbf{x}}(\theta_m) - r)\,\mathbf{n}_w$
      $\mathbf{x}_p \leftarrow \mathbf{x}_{\text{offset}} + \frac{\epsilon}{2}\mathbf{n}_w + a\frac{\epsilon\sqrt{3}}{2}\mathbf{t}_w$
      $\mathbf{x}^* \leftarrow \mathbf{x}_p$
      if $\alpha^* > \alpha_s^*$ **then**
        `mode` $\leftarrow$ `PathFollowing`
      end if
    end if
    $v \leftarrow$ Find Linear Velocity command ▷ (4.16)
    $\omega \leftarrow$ Find Angular Velocity command ▷ (4.17)
    $\mathbf{u}_{ku} \leftarrow (v, \omega)$
    COMMAND $\mathbf{u}_{ku}$
  while $||\mathbf{x} - \mathcal{P}(1)|| > \delta$
  return
end function

where $\phi_i = \text{atan2}(\mathbf{x}_i - \mathbf{x})$ and $\mathbf{e}_{\parallel} = (\cos\phi_i, \sin\phi_i) \in \mathbb{R}^2$ is the unit vector along the line connecting the two bodies. Since, the velocity of the object center of mass will be $\dot{\mathbf{x}}_i = \dot{\mathbf{x}} + (\rho_i + r)\,\dot{\phi}_i\,\mathbf{e}_{\perp}$ with $\mathbf{e}_{\perp} = (-\sin\phi_i, \cos\phi_i) \perp \mathbf{e}_{\parallel}$, and since the robot has a grip on the object along its line of motion, so that the orientation of the robot ψ is always equal to the robot-object bearing angle ϕ_i, we can use (4.1) to write

$$\dot{\mathbf{x}}_i = \mathbf{T}_i\,\mathbf{u}_{ku} \tag{4.21}$$

with the Jacobian $\mathbf{T}_i$ given by

$$\mathbf{T}_i = \begin{bmatrix} \cos\psi & -(\rho_i + r)\sin\psi \\ \sin\psi & (\rho_i + r)\cos\psi \end{bmatrix} \tag{4.22}$$

and $\mathbf{u}_{ku} = (v, \omega)$ the input vector as defined above.

Similarly, consider the circumscribed circle enclosing the robot and the object with radius $(\rho_i + r)$, as shown in Fig. 4.2. Its *center point* is located at

$$\mathbf{x}_{i,c} = \mathbf{x} + \rho_i\,\mathbf{e}_{\parallel} \tag{4.23}$$

Following a similar procedure as above, we can show that

$$\dot{\mathbf{x}}_{i,c} = \mathbf{T}_{i,c}\,\mathbf{u}_{ku} \tag{4.24}$$

with the Jacobian $\mathbf{T}_{i,c}$ given by

$$\mathbf{T}_{i,c} = \begin{bmatrix} \cos\psi & -\rho_i\sin\psi \\ \sin\psi & \rho_i\cos\psi \end{bmatrix} \tag{4.25}$$

4.4.2 Generating Virtual Commands

For the planning process, the fact that both $\mathbf{T}_i$ and $\mathbf{T}_{i,c}$ are always non-singular implies that we can describe the robot-object pair as either a dynamical system of the form

$$\dot{\mathbf{x}}_i = \mathbf{u}_i(\mathbf{x}_i) \tag{4.26}$$

or a dynamical system of the form

$$\dot{\mathbf{x}}_{i,c} = \mathbf{u}_{i,c}(\mathbf{x}_{i,c}) \tag{4.27}$$

since we can always prescribe (virtual) arbitrary velocity commands $\mathbf{u}_i$ or $\mathbf{u}_{i,c}$ for either the object itself or for the center point and then translate them to (actual) inputs $\mathbf{u}_{ku}$ through (4.21) or (4.24) respectively ($\mathbf{u}_{ku} = \mathbf{T}_i^{-1}\,\mathbf{u}_i$ or $\mathbf{u}_{ku} = \mathbf{T}_{i,c}^{-1}\,\mathbf{u}_{i,c}$).

Since the circumscribed circle centered at $\mathbf{x}_{i,c}$ is the smallest circle enclosing both the robot and the object and since Assumption 4.1 guarantees only that $\eta > 2(r + \max_k \rho_k)$, we conclude that it is beneficial to consider the dynamical system (4.27) (and generate virtual commands for the center point $\mathbf{x}_{i,c}$) when following the path $\mathcal{P}$ that the high-level planner provides. However, this will eventually position $\mathbf{x}_{i,c}$ to $\mathbf{p}_i^*$, instead of the object $\mathbf{x}_i$ (which is desired). Therefore, once the center point is placed to $\mathbf{p}_i^*$, we have to switch to the system (4.26) and generate virtual commands for the object $\mathbf{x}_i$ to carefully position it to $\mathbf{p}_i^*$. Assumption 4.2 guarantees that this is always possible. We can think of generating commands $\mathbf{u}_i$ and $\mathbf{u}_{i,c}$ as a trade-off between careful object positioning and agility in avoiding obstacles respectively.

4.4.3 LIDAR Range Transformation

As described above, the robot-object pair is treated as a single holonomic agent with radius $\rho_i + r$ centered at $\mathbf{x}_{i,c}$ when following the reference path $\mathcal{P}$. However, we know that the LIDAR is positioned on the robot and its range measurements are given with respect to $\mathbf{x}$. Therefore, we need a mechanism for translating these measurements from $\mathbf{x}$ to $\mathbf{x}_{i,c}$. To this

end, we describe the observed points from the LIDAR using the function $\mathbf{x}_{\text{LIDAR}} : (-\pi, \pi] \rightarrow \mathcal{W}$

$$\mathbf{x}_{\text{LIDAR}}(\theta) = \mathbf{x} + \rho_{\mathbf{x}}(\theta)\,(\cos\theta, \sin\theta) \tag{4.28}$$

and find the equivalent ranges from $\mathbf{x}_{i,c}$ as

$$\rho_{\mathbf{x}_{i,c}}(\theta) = \min\{R - \rho_i, ||\mathbf{x}_{\text{LIDAR}}(\theta) - \mathbf{x}_{i,c}||\} \tag{4.29}$$

since $R - \rho_i$ is the minimum distance that can be observed from $\mathbf{x}_{i,c}$ when no obstacles are present and corresponds to the ray along the orientation ψ of the robot towards the object.

We summarize the proposed algorithm for switching between a path following and a wall following phase and generating velocity commands for a robot-object pair following a reference path $\mathcal{P}$ in Algorithm 4.2, with the definition of an auxiliary symbolic action NavigateRobotObject$(\mathcal{P}, r, \rho_i, \epsilon, \delta)$.

4.5 Low-Level Implementation of Symbolic Language

In this Section, we describe the low-level implementation and realization of the three symbolic actions introduced in Section 4.1, used to solve our assembly problem.

4.5.1 Action MoveToObject

The low-level implementation of this symbolic action is quite straightforward, since the robot just needs to follow the plan provided by the high-level planner and navigate to a specific object using the auxiliary action NavigateRobot. The only caveat is that the robot needs to be aligned with the object it needs to pick up in order to engage the gripper. Since, no continuous law can guarantee both position and orientation convergence for a nonholonomically-constrained, differential drive robot [34] and a discontinuous law needs to be introduced, we compute

$$\tilde{\alpha} := \min\{\alpha \in [0, 1] \,|\, \mathcal{P}(\alpha) \in \mathsf{B}\,(\mathbf{p}_i, \rho_i + r)\} \tag{4.30}$$

Algorithm 4.2 Generating velocity commands for a nonholonomic robot with radius r moving an object of radius ρ_i along a reference path $\mathcal{P}$.

function NAVIGATEROBOTOBJECT($\mathcal{P}, r, \rho_i, \epsilon, \delta$)
  `mode` $\leftarrow$ `PathFollowing` ▷ Initialize `mode`
  do
    $\mathbf{x} \leftarrow$ Read Robot State
    $\psi \leftarrow$ Read Robot Orientation
    $\rho_{\mathbf{x}} \leftarrow$ Read LIDAR
    $\mathbf{x}_{i,c} \leftarrow$ Find center of circumscribed circle ▷ (4.23)
    $\rho_{\mathbf{x}_{i,c}} \leftarrow$ Transform LIDAR readings ▷ (4.29)
    $d \leftarrow \min_{\theta} \rho_{\mathbf{x}_{i,c}}(\theta) - (r + \rho_i)$
    $\alpha^* \leftarrow$ Find maximum path index ▷ (4.6)
    if `mode` = `PathFollowing` **then**
      $\mathcal{LF}_{\mathcal{L}}(\mathbf{x}_{i,c}) \leftarrow$ Find local free space ▷ (3.5)
      $\mathbf{x}^*_{i,c} \leftarrow \Pi_{\mathcal{LF}_{\mathcal{L}}(\mathbf{x}_{i,c})}(\mathcal{P}(\alpha^*))$
      if $d < \epsilon$ **then**
        `mode` $\leftarrow$ `WallFollowing`
        $\alpha^*_s \leftarrow \alpha^*$
        $a \leftarrow$ Find wall following direction ▷ (4.8)
      end if
    else if `mode` = `WallFollowing` **then**
      $\theta_m \leftarrow \arg\min_{\theta} \rho_{\mathbf{x}_{i,c}}(\theta)$
      $\mathbf{n}_w \leftarrow -(\cos\theta_m, \sin\theta_m)$
      $\mathbf{t}_w \leftarrow (\sin\theta_m, -\cos\theta_m)$
      $\mathbf{x}_{\text{offset}} \leftarrow \mathbf{x}_{i,c} - \left(\rho_{\mathbf{x}_{i,c}}(\theta_m) - r - \rho_i\right)\mathbf{n}_w$
      $\mathbf{x}_p \leftarrow \mathbf{x}_{\text{offset}} + \frac{\epsilon}{2}\mathbf{n}_w + a\frac{\epsilon\sqrt{3}}{2}\mathbf{t}_w$
      $\mathbf{x}^*_{i,c} \leftarrow \mathbf{x}_p$
      if $\alpha^* > \alpha^*_s$ **then**
        `mode` $\leftarrow$ `PathFollowing`
      end if
    end if
    $\mathbf{u}_{i,c} \leftarrow -k(\mathbf{x}_{i,c} - \mathbf{x}^*_{i,c})$ ▷ Virtual commands
    $\mathbf{u}_{ku} \leftarrow \mathbf{T}^{-1}_{i,c}\,\mathbf{u}_{i,c}$ ▷ Actual commands
    COMMAND $\mathbf{u}_{ku}$
  while $||\mathbf{x}_{i,c} - \mathcal{P}(1)|| > r + \delta$
  return
end function

and "truncate" the path to $\mathcal{P}([0, \tilde{\alpha}])$. In this way, the robot will navigate to $\mathcal{P}(\tilde{\alpha})$ (within a δ tolerance) which satisfies $||\mathcal{P}(\tilde{\alpha}) - \mathbf{p}_i|| = \rho_i + r$ as desired. Then, in order to align the robot with the object, the linear command v is set to zero and the angular command is set to

$$\omega = -k(\phi_i - \psi) \tag{4.31}$$

until $\phi_i = \psi$. The low-level implementation is shown in Algorithm 4.3.

4.5.2 Action PositionObject

From the preceding analysis in Section 4.4, we can construct the PositionObject algorithm as shown in Algorithm 4.4. Since the task of NavigateRobotObject is to bring the object close enough to the destination in order to allow careful positioning (allowed by Assumption 4.2), a final positioning step is required. To this end, instead of generating virtual commands for the center of the circumscribed circle of the robot-object pair as shown in (4.27), we generate commands for the center of the object itself, as shown in (4.26), according to the following law

$$\mathbf{u}_i = -k(\mathbf{x}_i - \mathbf{p}_i^*) \tag{4.32}$$

These virtual commands are then translated to actual robot commands according to (4.21).

Algorithm 4.3 Robot navigation to object $\mathbf{p}_i$ along path $\mathcal{P}$

```
1:  function MoveToObject(i, P)
2:      ε ← Set Wall Following Tolerance                    ▷ ε < η
3:      δ ← Set Placement Tolerance
4:      α̃ ← min{α ∈ [0,1] | P(α) ∈ B(p_i, ρ_i + r)}
5:      NavigateRobot(P([0, α̃]), r, ε, δ)
6:      while |φ_i − ψ| > δ do
7:          u_ku ← (0, −k(φ_i − ψ))                         ▷ Align with object
8:          COMMAND   u_ku
9:      end while
10:     g ← 1                                               ▷ Engage gripper
11:     return
12: end function
```

Algorithm 4.4 Position object i to $\mathbf{p}_i^*$ along path $\mathcal{P}$

function PositionObject($i, \mathcal{P}$)
  $\epsilon \leftarrow$ Set Wall Following Tolerance ▷ $\epsilon < \eta$
  $\delta \leftarrow$ Set Placement Tolerance
  NavigateRobotObject($\mathcal{P}, r, \rho_i, \epsilon, \delta$)
  do
    $\mathbf{x} \leftarrow$ Read Robot State
    $\psi \leftarrow$ Read Robot Orientation
    $\mathbf{x}_i \leftarrow$ Find object position ▷ (4.20)
    $\mathbf{u}_i \leftarrow -k(\mathbf{x}_i - \mathbf{p}_i^*)$ ▷ Virtual commands
    $\mathbf{u}_{ku} \leftarrow \mathbf{T}_i^{-1}\,\mathbf{u}_i$ ▷ Actual commands
    COMMAND $\mathbf{u}_{ku}$
  while $||\mathbf{x}_i - \mathbf{p}_i^*|| > \delta$
  $g \leftarrow 0$ ▷ Disengage gripper
  return
end function

Algorithm 4.5 Free robot navigation along path $\mathcal{P}$

function Move($\mathcal{P}$)
  $\epsilon \leftarrow$ Set Wall Following Tolerance ▷ $\epsilon < \eta$
  $\delta \leftarrow$ Set Placement Tolerance
  NavigateRobot($\mathcal{P}, r, \epsilon, \delta$)
  return
end function

4.5.3 Action Move

This action is similar to MoveToObject, but there is no final orientation requirement. Its low-level implementation is shown in Algorithm 4.5.

Note here that the formal results accompanying both the path following phase [8] and the wall following phase (Theorems 4.1 and 4.3) along with Theorem 4.2 guarantee that every symbolic action command will be successfully executed.

4.6 Numerical Examples

In this Section, we provide numerical examples[6] of assembly processes in various environments using the symbolic action commands described above.

4.6.1 Environment Packed Circular Obstacles

First, we test our algorithm in a rectangular, 20x20m workspace, packed with circular obstacles, whose position and size are unknown to the deliberative planner. The minimum separation η between the obstacles is chosen to be only slightly above (e.g 5cm) the minimum allowed value prescribed by Assumption 4.1, in order to demonstrate the validity of our approach, deriving from the formal guarantees of Theorem 4.1. The goal is to place an object to a desired position, shown in Fig. 4.5. The deliberative planner outputs a plan comprising of two actions: $\textsc{MoveToObject}(1, \mathcal{P}_1) \rightarrow \textsc{PositionObject}(1, \mathcal{P}_2)$, whose sequential execution and the corresponding reference paths $\mathcal{P}_1, \mathcal{P}_2$ are depicted in Fig. 4.5.

4.6.2 Cluttered Environment with Walls

Here we demonstrate the execution of a more challenging task. The robot should position the two obstacles depicted in Fig. 4.6 to their predefined positions within a polygonal workspace with walls, whose locations are provided a-priori to the deliberative planner, and then return to a "nest" location. The workspace is packed with several convex, not-necessarily circular obstacles. As shown in Fig. 4.6, the deliberative planner outputs a high-level plan comprising of five actions: $\textsc{MoveToObject}(1, \mathcal{P}_1) \rightarrow \textsc{PositionObject}(1, \mathcal{P}_2) \rightarrow$

[6] All simulations were run in MATLAB using `ode45` and a gain $k = 2$.

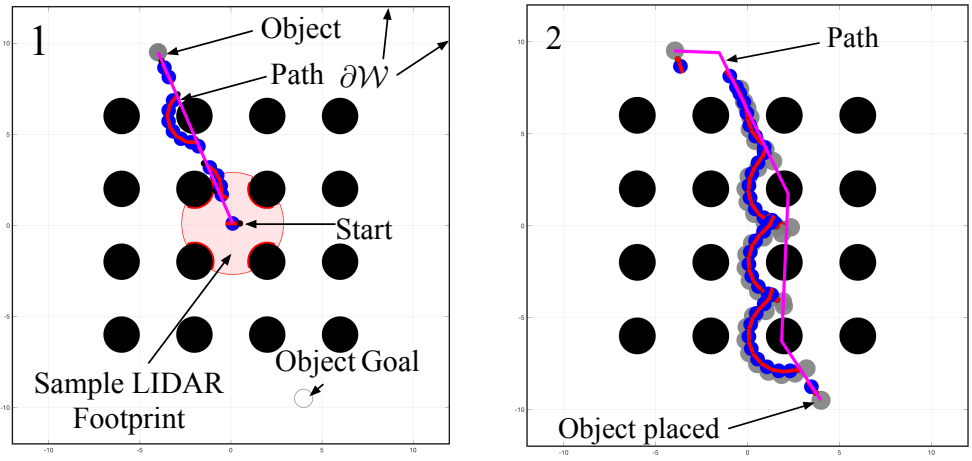

Figure 4.5: A depiction of a packed two stage assembly process with a fixed timestep, with the separation value just above the minimum allowed value.

$\textsc{MoveToObject}(2, \mathcal{P}_3) \rightarrow \textsc{PositionObject}(2, \mathcal{P}_4) \rightarrow \textsc{Move}(\mathcal{P}_5)$, which is successfully executed by the reactive planner. An example for an object trajectory during this execution is shown in Fig. 4.1. Notice that, in contrast with several reactive wall following schemes that require an estimate of the wall curvature, our scheme can easily handle obstacles with corners. It is also worth noting that the deliberative planner hit the maximum number of expansions allowed and had difficulties extracting a feasible plan when it was provided the exact position and size of every obstacle, due to the highly packed construction. This highlights another benefit of our approach; we can significantly reduce the computational load of high-level planners by tasking them only with the extraction of the action sequence required, and using the reactive planner for local obstacle avoidance and convergence online. This happens because the computational load of the reactive planner remains the same regardless of the number of obstacles.

Finally, it is worth noting that the proposed scheme is capable of executing a sequence of symbolic commands provided by the deliberative planner, even when Assumptions 4.1 or 4.2 or the obstacle convexity are not satisfied. In the accompanying video submission of [203][7], we provide examples of successful assemblies even in the absence of both obstacle convexity and enough separation.

[7] `https://youtu.be/_07_q-edjmM`

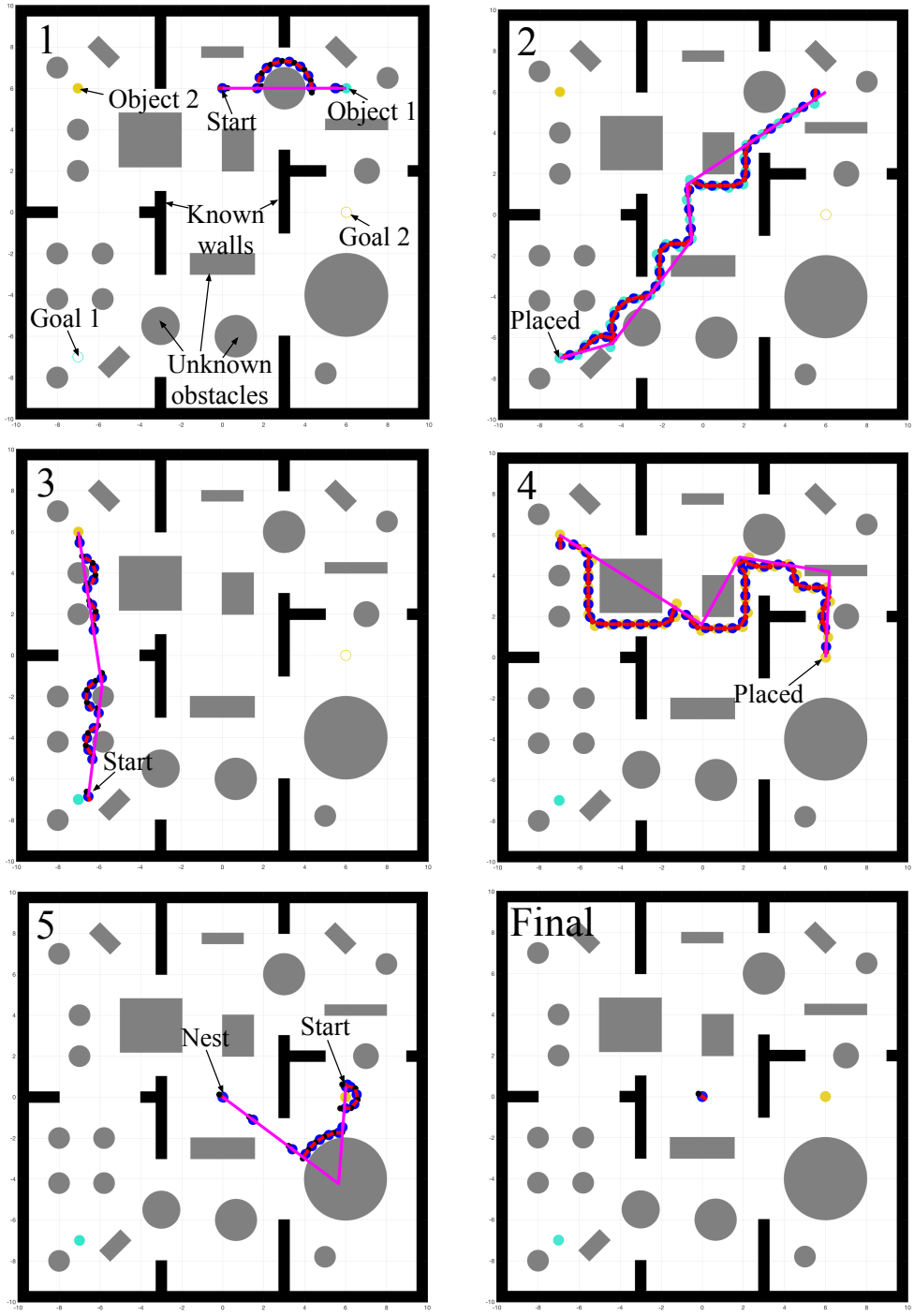

Figure 4.6: An illustration of the assembly process described in Section 4.6.2, with a fixed timestep. The walls and boundaries of the workspace, known to the deliberative planner, are shown in black and the unexpected obstacles handled by the reactive planner are shown in grey.

Chapter 5

Reactive Execution of Symbolic Rearrangement Plans with Minitaur

In this Chapter, we expand the architecture presented in Chapter 4 and demonstrate the physical rearrangement of wheeled stools in a moderately cluttered indoor environment, by a quadrupedal robot that autonomously achieves a user's desired configuration. The robot's behaviors are planned and executed by a three layer hierarchical architecture consisting of: an offline symbolic task and motion planner; a reactive layer that tracks the reference output of the deliberative layer and avoids unanticipated obstacles sensed online; and a gait layer that realizes the abstract unicycle commands from the reactive module through appropriately coordinated joint level torque feedback loops. This Chapter also extends prior formal results about the reactive layer to a broad class of non-convex obstacles. Our design is verified both by formal proofs as well as empirical demonstration of various assembly tasks.

The Chapter is organized as follows. Section 5.1 describes the problem and summarizes our approach. Section 5.2 describes each component of the hierarchical control structure (deliberative, reactive and gait controller) separately, and Section 5.3 presents our formal results on reactive wall following for non-convex obstacles. Finally, Section 5.4 begins with the description of our hardware infrastructure based on ROS and continues with the pre-

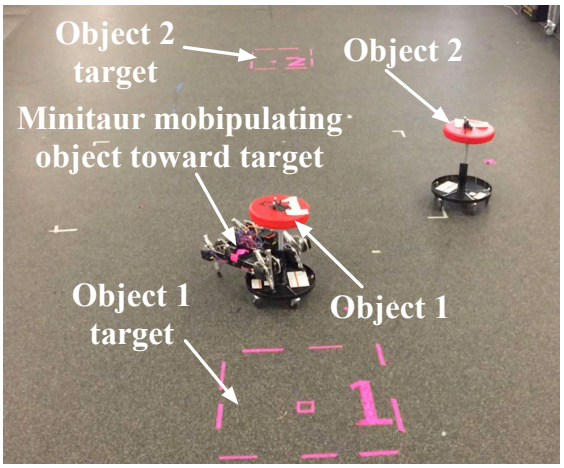

Figure 5.1: LIDAR-equipped Minitaur [65] mobipulating [130] two stools using gaits [49] called out by a deliberative/reactive motion planner (Chapter 4).

sentation of our empirical results for different classes of experiments.

5.1 Problem Formulation

As in Chapter 4, Minitaur is assumed to operate in a closed and compact workspace $\mathcal{W} \subset \mathbb{R}^2$ whose boundary $\partial\mathcal{W}$ is assumed to be known, and is tasked to move each of $n \in \mathbb{N}$ movable disk-shaped objects, centered at $\mathbf{p} := (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n) \in \mathcal{W}^n$ with a vector of radii $(r_1, r_2, \ldots, r_n) \in (\mathbb{R}_{>0})^n$, from their initial configuration to a user-specified goal configuration $\mathbf{p}^* := (\mathbf{p}_1^*, \mathbf{p}_2^*, \ldots, \mathbf{p}_n^*) \in \mathcal{W}^n$. For our hardware implementation, the movable objects are stools with five caster wheels. We assume that both the initial configuration and the target configuration of the objects are known. In addition to the known boundary of the workspace $\partial\mathcal{W}$, the workspace is cluttered by an unknown number of fixed, disjoint, potentially non-convex obstacles of unknown position and size, denoted by $\mathcal{O} := (O_1, O_2, \ldots)$. To simplify the notation, we also define $\mathcal{O}_w := \mathcal{O} \cup \partial\mathcal{W}$.

As discussed in Section 3.1, for (reactive) planning purposes, Minitaur is modeled as a first-order, nonholonomically-constrained, disk-shaped robot, centered at $\mathbf{x} \in \mathbb{R}^2$ with radius $r \in \mathbb{R}_{>0}$ and orientation $\psi \in S^1$. The model dynamics are described by

$$(\dot{\mathbf{x}}, \dot{\psi}) = \mathbf{B}(\psi)\mathbf{u}_{ku} \tag{5.1}$$

with $\mathbf{B}(\psi) := \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$ the differential constraint matrix and $\mathbf{u}_{ku} := (v, \omega)$ the input vector consisting of a linear and an angular command. Similarly to Chapter 4, we adopt Assumptions 4.1 and 4.2 to facilitate the proofs of our formal results, which are not necessary for the existence of some solution to the problem.

The robot is assumed to have access to its state[1] $(\mathbf{x}, \psi)$ and to possess a LIDAR for local obstacle avoidance, positioned at $\mathbf{x}$, with a 360° angular scanning range and a fixed sensing range $R \in \mathbb{R}_{>0}$. It is also assumed to use a gripper for moving objects, which can be engaged or disengaged; we will write $g = 1$ when the gripper is engaged and $g = 0$ when it is disengaged. Of course, Minitaur is only an imperfect unicycle (as discussed in Section 3.1) and does not actually possess a gripper; it has to successfully coordinate its limbs and walk while following a path, avoid an obstacle or lock an object in place and move it to a desired location. Hence, the reactive planner's commands $(\mathbf{u}_{ku}, g)$ must in turn be translated to appropriate low-level commands on the robot's joint level.

The aforementioned description imposes a hierarchical structure, as shown in Fig. 5.2. The deliberative planner is endowed with a symbolic command set comprised of three actions: MoveToObject$(i, \mathcal{P})$, PositionObject$(i, \mathcal{P})$ and Move$(\mathcal{P})$. Here i is the desired object and $\mathcal{P}$ is a piecewise continuously differentiable path $\mathcal{P} : [0, 1] \to \mathcal{W}$ connecting an initial and a final position, which can be seen as a "geometric suggester" in the sense of [89]. This command set suggests the following problem decomposition into the complementary sub-problems:

1. In the *deliberative layer*, find a *symbolic plan*, i.e., a sequence of symbolic actions whose successful implementation is guaranteed to complete the task.

2. In the *reactive layer*, implement each of the symbolic actions by finding appropriate commands $(\mathbf{u}_{ku}, g)$ according to the robot's equations of motion shown in (5.1), while avoiding the perceived obstacles (unanticipated by the deliberative planner) encoun-

[1]Since legged state estimation falls beyond the scope of this work, localization is performed using a Vicon motion capture system [212].

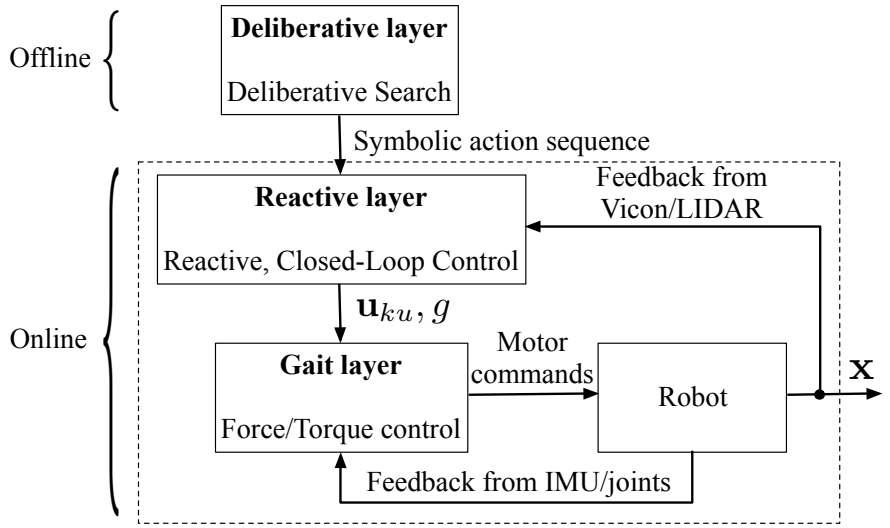

Figure 5.2: A coarse block diagram of the planning and control architecture, following Fig. 1.1 without including the interface layer, since it is assumed that each provided high-level action is always feasible. In the deliberative layer, a high-level planner [210] outputs a sequence of symbolic actions that are realized and executed sequentially using a reactive controller that issues unicycle velocity ($\mathbf{u}_{ku}$) (see Chapter 4), and abstract gripper (g) commands (see Section 5.2.2). The low-level gait layer uses the commands instructed by the reactive planner to call out appropriately parametrized joint-level feedback controllers (see [49] and Section 5.2.3) for Minitaur.

tered along the way.

3. In the *gait layer*, use a hybrid dynamical systems framework with simple guard conditions to choose between constituent gaits, providing a unicycle interface to the reactive layer, controllable by $(\mathbf{u}_{ku}, g)$, regardless of the state of the agent and objects.

5.2 System Architecture

In this Section, we describe the three-layer architecture used to accomplish the task at hand, shown in Fig. 5.2. After a description of the offline deliberative planner, we proceed with the features of the online reactive module and the new, low-level layer of control (the "gait" layer), used to achieve on Minitaur the commands instructed by the reactive layer.

5.2.1 Deliberative Layer

Similarly to Section 4.2, the deliberative layer finds a feasible path through the joint configuration space of the robot and anticipated environment. It takes as input a metric description

of the world, including object and obstacle geometry, and proceeds in two stages. First, it discretizes the environment by constructing a factored random geometric graph [208]. The factored graph is the product of $n + 1$ probabilistic roadmaps, one for each object and one for the robot. Each edge in the factored graph represents a feasible motion; these motions are either paths of the robot while the other objects do not move, or paths of the robot carrying a single object. Paths through this graph then represent continuous paths through configuration space.

This graph construction is asymptotically optimal; as the number of vertices in each factor increases, the cost of the best path through the graph approaches the cost of the optimal path. In addition, the factored representation allows us to quickly construct graphs with an exponential number of vertices. However, the number of graph vertices is exponential in n. We can search for a near-optimal path through the graph in a reasonable amount of time using the angelic hierarchical A* algorithm [128, 210]. This algorithm interleaves the search over high-level decision, like which objects to grasp and in which order, and over lower-level details, like where objects should be placed, by using a hierarchy of abstract operators, which are implicitly-defined sets of plans that achieve a specified effect. For example, the operator MoveToObject$(i, \cdot)$ represents any plan that eventually reaches object i.

We can derive bounds on the cost of any primitive plan contained in an abstract operator. For example, the cost of any plan in MoveToObject$(i, \cdot)$ starting from a position $\mathbf{x}$ is greater than the Euclidean distance from $\mathbf{x}$ to object i. If we find some path from $\mathbf{x}$ to object i, its cost is an *upper* bound on the cost of the best plan from $\mathbf{x}$ to object i. Using these bounds, we can estimate the cost of plans composed of sequences of abstract operators, allowing us to prune bad plans early and refine promising plans first. More importantly, these bounds allow us to prove that a symbolic plan is feasible before providing it to the reactive layer.

5.2.2 Reactive Layer

As shown in Fig. 5.2, the reactive layer is responsible for executing the symbolic action sequence, i.e., the output of the deliberative planner, using Algorithms 4.3, 4.4 and 4.5. As described in Section 4.3, we decompose the reactive behavior into two separate modes determined by the absence or presence of unanticipated obstacles:

Anticipated Environment

In the absence of unanticipated obstacles, the robot is in *path following mode*. Based on the results of [7, 8], this mode is responsible for steering the robot along a reference path $\mathcal{P}$ given by the deliberative planner. This is achieved by following the *projected-path goal* $\mathcal{P}(\alpha^*)$ with α^* determined as

$$\alpha^* := \max\{\alpha \in [0,1] \,|\, \mathcal{P}(\alpha) \in \mathsf{B}\left(\mathbf{x}, d(\mathbf{x}, \partial\mathcal{F})\right)\} \tag{5.2}$$

constantly updated as the agent moves along the path. Here $d(\mathbf{x}, \partial\mathcal{F})$ denotes the distance of the agent from the boundary of the free space $\mathcal{F}$, determined as

$$d(\mathbf{x}, \partial\mathcal{F}) = \min_{\theta} \rho_{\mathbf{x}}(\theta) - r \tag{5.3}$$

with $\rho_{\mathbf{x}}(\theta)$ denoting the polar curve describing the LIDAR measurements [7] (see (4.4)).

Unanticipated Obstacles

In the presence of unanticipated obstacles, i.e., when $d(\mathbf{x}, \partial\mathcal{F}) < \epsilon$ with ϵ a desired tolerance, the robot switches to *wall following mode*. In this mode, described in Section 4.3.3, the robot follows the *wall-following goal* $\mathbf{x}_p(\mathbf{x})$ defined as

$$\mathbf{x}_p(\mathbf{x}) := \mathbf{x}_{\text{offset}}(\mathbf{x}) + \frac{\epsilon}{2}\mathbf{n}_w(\mathbf{x}) + a\frac{\epsilon\sqrt{3}}{2}\mathbf{t}_w(\mathbf{x}) \tag{5.4}$$

with $\mathbf{x}_{\text{offset}}(\mathbf{x}) := \mathbf{x} - (\rho_{\mathbf{x}}(\theta_m) - r)\,\mathbf{n}_w(\mathbf{x})$ an offset point from the obstacle boundary, $\theta_m := \arg\min_{\theta} \rho_{\mathbf{x}}(\theta)$ the LIDAR angle corresponding to the minimum distance from the obstacle,

$\mathbf{n}_w(\mathbf{x}) := -(\cos\theta_m, \sin\theta_m)$ the normal vector to the boundary of the obstacle at the point of minimum distance and $\mathbf{t}_w(\mathbf{x}) := (\sin\theta_m, -\cos\theta_m)$ the corresponding tangent vector. Finally, $a \in \{-1, 1\}$ denotes the wall following direction (1 for CCW motion and -1 for CW motion). The robot exits the wall following mode and returns to the path following mode once it encounters the path again, i.e., when $\alpha^* = \max\{\alpha \in [0,1] \,|\, \mathcal{P}(\alpha) \in \mathsf{B}\,(\mathbf{x}, d(\mathbf{x}, \partial\mathcal{F})\} > \alpha_s^*$, with α_s^* the saved path index at the beginning of the wall following mode. As outlined in Theorem 4.1, the *wall following law*

$$\mathbf{u}(\mathbf{x}) = -k(\mathbf{x} - \mathbf{x}_p) \tag{5.5}$$

provides an easy formula for wall following within specified bounds, even in the absence of obstacle curvature information. This allows for fast computation, which is critical in our legged robot setting. The reader is again referred to (4.8) for the choice of wall following direction and to Theorem 4.3 for an extension to differential drive robots.

5.2.3 Gait Layer

Hybrid Dynamical System Structure

The gait layer's primary function is to interpret simple unicycle commands $\mathbf{u}_{ku} = (v, \omega)$, as well as simple gripper commands by mapping them into physical joint level robot behaviors and transitions between them that realize the reactive layer's abstracted gripping/releasing unicycle model in the physical world. This structure naturally lends itself to the hybrid dynamical systems framework and we conjecture that the following architecture meets the requirements of a formal simple hybrid dynamical manipulation and self-manipulation system [87].

Let $\mathbf{x}_M \in \mathcal{X}_M$ be the robot pose and joint state and let $\mathbf{g} := (g_s, g_v, g_a) \in \{0,1\}^3$ be a vector representing gripper state, where $g_s \in \{0,1\}$ representing "open" and "closed" respectively, $g_v \in \{0,1\}$, representing zero and non-zero gripper transition velocity, and $g_a \in \{0,1\}$ representing zero and non-zero gripper command from the reactive layer, with $\mathbf{g}$ arranged as yet another component of the gait layer's state. Thus, taking $\mathbf{x}_{M^+} = (\mathbf{x}_M, \mathbf{g}) \in$

$\mathcal{X}_M \times \{0,1\}^3$ as the hybrid state, our hybrid system modes arise from the disjoint union of the geometric placement indexed by the 4 mutually exclusive gripper conditions as follows:

$$\mathbf{x}_{M^+} \in \begin{cases} \mathcal{M}_W = \{\mathbf{x}_{M^+} \,|\, g_s = 0, g_v = 0\} & \textit{“Walk”} \\ \mathcal{M}_M = \{\mathbf{x}_{M^+} \,|\, g_s = 0, g_v = 1\} & \textit{“Mount”} \\ \mathcal{M}_P = \{\mathbf{x}_{M^+} \,|\, g_s = 1, g_v = 0\} & \textit{“Push Walk”} \\ \mathcal{M}_D = \{\mathbf{x}_{M^+} \,|\, g_s = 1, g_v = 1\} & \textit{“Dismount”} \end{cases}$$

where the guard condition, $g_a = 1$, triggers appropriate resets so that the hybrid mode system changes in the recurring sequence: $\mathcal{M}_W \to \mathcal{M}_M \to \mathcal{M}_P \to \mathcal{M}_D \to \mathcal{M}_W ...$

Mode Dynamics

A formal representation of the legged controllers and resulting closed loop dynamics used to realize the abstracted unicycle grip/release behaviors lies beyond the scope of this work. Instead, we now provide a brief, informal account of each mode as follows.

Walk: Incorporated here as reported in Section 3.1, this behavior is adapted from the still developing insights of [49]. While the kinematic model of the Minitaur platform prevents it from literal unicycle behavior in quasi-static operation, the underlying family of controllers overcomes this deficiency by dynamically exploiting higher-order effects, such as bending of the limbs and frame, as well as toe-slipping.

Mount: The mounting behavior, a physical realization of the abstract $(g_s, g_v) = (0, 1)$ state, comprises a sequential composition that we conjecture can be placed within the formal framework of [35]. Informally, the behavior begins by leaping with the front legs, while maintaining ground contact with the rear, as shown in Fig. 5.3-2. During this “flight” phase, an attempt to servo to the desired yaw is made by generating a difference in ground reaction

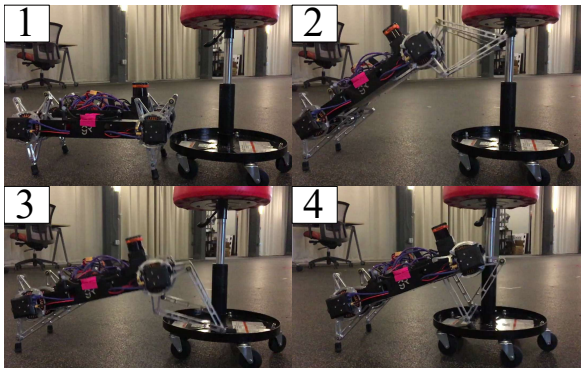

Figure 5.3: Consecutive snapshots of a successful "Mount" onto an object.

forces in the stance legs according to the control law:

$$F_{LH} = -\frac{1}{2}(k_p(\psi_{des} - \psi) - k_d\dot{\psi})$$
$$F_{RH} = \frac{1}{2}k_p(\psi_{des} - \psi) - k_d\dot{\psi}$$

where F_{LH} is the ground reaction force on the body generated by the left hip, and F_{RH} is the analogous force generated by the right hip. Note that this method does not use Minitaur's kinematic configuration as a means of measuring ψ, and as such is able to continue to servo to the desired heading even in the presence of toe slipping or bending in the body. However, as contact modes are not assured and Vicon data is not available to the gait layer, the measurement of ψ is obtained by integrating gyroscope data, which for this short behavior (less than a second) is reasonably accurate. In this Chapter, we implicitly assume that the mounting behavior is always successful. Since failures might occur, we intend to relax this assumption in the future by introducing feedback in the hybrid mode system presented above.

Push-Walk: This behavior attempts to mask the underlying dynamics of the system consisting of the Minitaur platform with the front two limbs in various contact modes with a holonomic (albeit not friction-less) stool, and the rear two in varying contact modes with the ground. In Section 4.4.2, we introduced a method for generating "virtual" commands for different points of interest in the holonomic robot-object pair when a gripper is utilized, and translating them to "actual" commands for the differential drive robot using simple

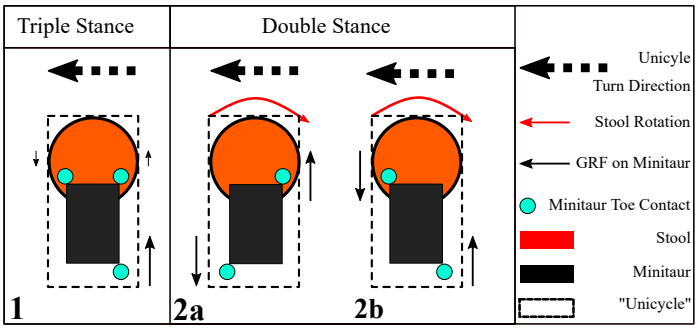

Figure 5.4: Intuition underlying how intermittent contact (yaw push-walk) provides larger moments on the system than the moments produced in a triple stance (fore-aft push-walk). In (1), the presence of both toes on the stool kinematically constrains it so that any reaction forces generated by those toes are internal forces of the Minitaur-Stool system, where as in (2a) and (2b), the stool is free to rotate, allowing the single front toe to generate a moment on the Minitaur body.

kinematic maps. The goal of this behavior is to exploit this result and use Minitaur's front legs as a virtual gripper.

The behavior is divided into two components; the fore-aft push-walk, and the yaw push-walk. The fore-aft push-walk is simply the previously described walking gait [49], modified such that the front limbs cannot retract to break contact with the stool. The yaw push-walk is a bit more dynamic, as the empirical application of the fore-aft walk in turning situations proved to have prohibitively small radius of curvature. To improve upon this, the front legs are allowed to retract as they would during walking, breaking and re-establishing contact with the stool on each step. The result is that the Minitaur is "freed" from the kinematic constraint of being unable to turn sharply enough in a manner described intuitively in Fig. 5.4, avoiding triple stance (Fig. 5.4.1) in favor of the more strongly yawing torques arising from double stance (Fig. 5.4.2a,b).

Dismount: Finally, the $(g_s, g_v) = (1, 1)$ state is encoded by employing the walking behavior with controller parameters set as - 1) the height of the walk, or the nominal length of a stance leg is made nearly maximum, and 2) a simple open-loop fore-aft trajectory is programmed to linearly ramp up the speed to a pre-determined backward rate and then back down to zero.

5.3 Extension of Reactive Layer to Non-Convex Obstacles

In this Section, we extend the result of Theorem 4.1 regarding the wall following law (5.5) to a class of non-convex obstacles satisfying specific criteria. We begin with some notation and basic definitions for non-convex obstacles.

Definition 5.1 ([44]). *Let $\mathcal{X}$ be a Hilbert space and $\mathcal{S}$ a closed set of $\mathcal{X}$. For $\mathbf{x} \in \mathcal{X}$ we denote by $Proj_{\mathcal{S}}(\mathbf{x})$ the (possibly empty) set of nearest points of $\mathbf{x}$ in $\mathcal{S}$. When $Proj_{\mathcal{S}}(\mathbf{x})$ is a singleton, its single point is called the metric projection and denoted by $\Pi_{\mathcal{S}}(\mathbf{x})$, i.e., $Proj_{\mathcal{S}}(\mathbf{x}) = \{\Pi_{\mathcal{S}}(\mathbf{x})\}$.*

Definition 5.2 ([44]). *A vector $\mathbf{v} \in \mathcal{X}$ is said to be a proximal normal vector of $\mathcal{S}$ at $\mathbf{x} \in \mathcal{S}$ whenever there exists $t > 0$ such that $\mathbf{x} \in Proj_{\mathcal{S}}(\mathbf{x} + t\mathbf{v})$. The set of such vectors is the proximal normal cone of $\mathcal{S}$ at $\mathbf{x}$, denoted by $N^P(\mathcal{S}; \mathbf{x})$.*

Definition 5.3 ([44]). *Given an extended real $r \in [0, +\infty]$ and a real $\alpha > 0$, we say that a closed set $\mathcal{S}$ of $\mathcal{X}$ is (r, α)-prox-regular at $\mathbf{x}_0 \in \mathcal{S}$ if for every $\mathbf{x} \in \mathcal{S} \cap \mathsf{B}(\mathbf{x}_0, \alpha)$ and every direction $\zeta \in N^P(\mathcal{S}; \mathbf{x}) \cap \mathsf{B}(\mathbf{0}, 1)$, we have that $\mathbf{x} \in Proj_{\mathcal{S}}(\mathbf{x} + t\zeta)$ for every real $t \in [0, r]$.*

We say that $\mathcal{S}$ is *r-prox-regular at $\mathbf{x}_0 \in \mathcal{S}$* if it is (r, α)-prox-regular at $\mathbf{x}_0$ for some $\alpha > 0$ and we simply say that $\mathcal{S}$ is *prox-regular at $\mathbf{x}_0$* if there exists $r \in [0, +\infty]$ such that $\mathcal{S}$ is r-prox-regular at $\mathbf{x}_0$. Finally, $\mathcal{S}$ is *prox-regular* (resp. *r-prox-regular*) if it is prox-regular (resp. r-prox-regular) at every point $\mathbf{x} \in \mathcal{S}$. It is known [44] that $\mathcal{S}$ is prox-regular if and only if there exists a continuous function $\rho : \mathcal{S} \to [0, \infty]$, called the *prox-regularity function*, such that for every $\mathbf{x} \in \mathcal{S}$ and every $\zeta \in N^P(\mathcal{S}; \mathbf{x}) \cap \mathsf{B}(\mathbf{0}, 1)$ one has $\mathbf{x} \in \mathrm{Proj}_{\mathcal{S}}(\mathbf{x} + t\zeta)$ for every real $t \in [0, \rho(\mathbf{x})]$. The definition of prox-regularity itself is relatively abstract, but we attempt to ground it in the following paragraphs and Fig. 5.5.

It is also useful to include the definitions of the following enlargements of the set $\mathcal{S}$,

according to [44]

$$\begin{aligned}\mathcal{R}_{\mathcal{S}}(\mathbf{x}_0, r, \alpha) :=& \{\mathbf{x} + t\mathbf{v} \,:\, \mathbf{x} \in \mathcal{S} \cap \mathsf{B}\left(\mathbf{x}_0, \alpha\right), t \in [0, r], \\ & \mathbf{v} \in N^P(\mathcal{S}; \mathbf{x}) \cap \mathsf{B}\left(\mathbf{0}, 1\right)\} \\ U_{\rho(\cdot)}(\mathcal{S}) :=& \{\mathbf{x} \in \mathcal{X} \,:\, \exists \mathbf{y} \in \mathrm{Proj}_{\mathcal{S}}(\mathbf{x}) \,\text{with}\, d_{\mathcal{S}}(\mathbf{x}) < \rho(\mathbf{y})\}\end{aligned}$$

With these definitions, we are led to the following lemma.

Lemma 5.1. *If $\mathcal{S}$ is $\rho(\cdot)$-prox-regular, then the collection of sets $\{\mathcal{R}_{\mathcal{S}}(\mathbf{x}, \rho(\mathbf{x}), \alpha) \,:\, \mathbf{x} \in \mathcal{S}\}$ with $\alpha > 0$ corresponding to the prox-regularity condition forms an open cover of $U_{\rho(\cdot)}(\mathcal{S})$.*

Proof. Included in Appendix C.3. □

In [44, Theorem 2.3], it is also shown that if $\mathcal{S}$ is (r, α)-prox-regular at a point $\mathbf{x}_0$, then $\Pi_{\mathcal{S}}$ is well-defined and locally Lipschitz continuous on the set $\mathcal{R}_{\mathcal{S}}(\mathbf{x}_0, r, \alpha)$. Hence, using Lemma 5.1, we arrive at the following result.

Lemma 5.2. *If $\mathcal{S}$ is $\rho(\cdot)$-prox-regular, then $\Pi_{\mathcal{S}}$ is well-defined and locally Lipschitz continuous on $U_{\rho(\cdot)}(\mathcal{S})$.*

In this way, we can formulate the following theorem, that extends the guarantees of our wall-following control law to $\rho(\cdot)$-prox-regular, non-convex obstacles.

Theorem 5.1. *In the presence of $\rho(\cdot)$-prox-regular, convex[2] or non-convex isolated obstacles $\mathcal{O} := (O_1, O_2, \ldots)$ in the workspace satisfying Assumption 4.1 with $\min \rho_{O_i} > r + \epsilon$ for each O_i and ϵ chosen as in (4.10), the wall following law (5.5) has no stationary points, leaves the robot's free space $\mathcal{F}$ positively invariant under its unique continuously differentiable flow, and steers the robot along the boundary of a unique obstacle in $\mathcal{O}$ in a clockwise or counterclockwise fashion (according to the selection of a) with a nonzero rate of progress, while maintaining a distance of at most $(r + \epsilon)$ and no less than $\left(r + \frac{\epsilon}{2}\right)$ from it.*

Proof. Included in Appendix C.3. □

[2] Convex bodies are $\rho(\cdot)$-prox-regular by construction [44].

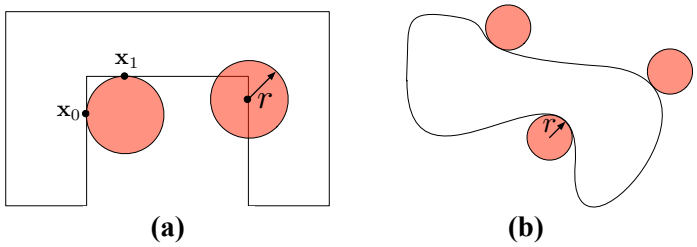

Figure 5.5: Intuitive description of prox-regularity, following Proposition 5.1: (a) An example of a non-convex body that fails to be r-prox-regular; since $0 < ||\mathbf{x}_1 - \mathbf{x}_0|| < 2r$, the existence of a tangent closed ball of radius r to both $\mathbf{x}_0$ and $\mathbf{x}_1$ violates the r-oval-segment criterion, (b) An example of an r-prox-regular non-convex body in $\mathbb{R}^2$, satisfying Proposition 5.1.

We include the following definition to provide some intuition on the abstract definition of prox-regularity and its relation to real obstacles in the physical world.

Definition 5.4 ([152]). *For any $r > 0$ and $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{X}$ with $||\mathbf{x}_1 - \mathbf{x}_0|| < 2r$, the r-oval segment $\Delta_r(\mathbf{x}_0, \mathbf{x}_1)$ in $\mathcal{X}$ with endpoints $\mathbf{x}_0, \mathbf{x}_1$ is defined as the intersection of all closed balls with radius r containing $\mathbf{x}_0, \mathbf{x}_1$.*

Then it can be shown that a closed set $\mathcal{S}$ of $\mathcal{X}$ is r-prox regular if and only if for any pair of points $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{S}$ with $0 < ||\mathbf{x}_1 - \mathbf{x}_0|| < 2r$, the r-oval segment $\Delta_r(\mathbf{x}_0, \mathbf{x}_1)$ contains a point of $\mathcal{S}$ different from $\mathbf{x}_0, \mathbf{x}_1$, or equivalently $\mathcal{S} \cap \Delta_r(\mathbf{x}_0, \mathbf{x}_1) \neq \{\mathbf{x}_0, \mathbf{x}_1\}$. Prox-regularity can, therefore, be seen as a means of defining an appropriate "length-scale" for the "nonconvexities" (e.g. valleys or traps) of the obstacle that do not result in a controller failure. Fig. 5.5 provides one example of non-convex body for which this prox-regularity criterion fails and one example for which it succeeds. As a guide, we provide the following sufficient condition, based on curvature, for prox-regularity in $\mathbb{R}^2$ without proof.

Proposition 5.1. *A closed, compact, simply-connected body $\mathcal{S} \subset \mathbb{R}^2$ is r-prox-regular if any tangent closed ball of radius r at its boundary $\partial\mathcal{S}$ has only one common point with $\mathcal{S}$.*

In the future, we would like to use the formal guarantees of Theorem 5.1, whose assumptions are mere sufficient conditions, to extend the application of doubly-reactive planners [7] to non-convex obstacles in Hilbert spaces.

5.4 Experimental Results

In this Section, we begin with a brief description of the hardware and software setup and continue with a description of the experiments run and our empirical results.

5.4.1 Setup

ROS Infrastructure

For the hardware and software experimental setup, we use a system structure similar to that presented in Section 3.4, shown in Fig. 5.6. A custom ROS node on the Raspberry Pi receives $\mathbf{u}_{ku}$ and the desired mode of operation ("Walk", "Mount", "Push-Walk", "Dismount") as ROS messages from the desktop computer and forwards them to the Minitaur mainboard (microcontroller implementing the gait layer functionalities) at 100Hz over a 115.2 Kbps USART connection. The Raspberry Pi acts as the ROS Master that resolves networking for the rest of the ROS nodes: a dedicated ROS node is activated as soon as the system boots and subscribes to the $\mathbf{u}_{ku}$ ROS topic (using the `Twist` message type), as well as an additional one capable of defining the desired behavior. A final ROS node running on the Raspberry Pi, taken from [166], forwards LIDAR measurements (using the `LaserScan` message type) to the desktop computer.

The pose information, consisting of the horizontal plane coordinates of the robot and all the objects and the orientation of the robot, is extracted from a Vicon Motion Capture System [212] at 100 Hz, using a set of motion capture cameras positioned around a 20m $\times$ 6m arena. The desktop computer receives the online data from Vicon using the ROS package `mocap_vicon` [114] and forwards it to the desktop computer running ROS. The reactive layer runs at approximately 30Hz, which is more than enough for the robot to recover if any obstacle is detected, and the gait controller runs at 1KHz.

LIDAR Measurement Handling

The LIDAR measurements are pre-processed by the desktop computer before being used by the reactive planner. First of all, following the requirements of [7], range measurements

greater than the limit R are set to R. All measurements are projected on the horizontal plane using the robot pitch angle measurement provided by the motion capture system. Finally, when the reactive layer executes a symbolic action MoveToObject$(i, \mathcal{P})$ or PositionObject$(i, \mathcal{P})$, it is critical to recognize the points of the LIDAR pointcloud associated with the object i and not use them for the calculation of the local freespace, since i should not be an obstacle. Hence, we look for points of the LIDAR pointcloud that are "close-enough" (within a δ_{object} tolerance) of the object i position and set the associated ranges to infinity. Unfortunately, this results in the object blocking the robot's line of sight during PositionObject$(i, \mathcal{P})$, meaning part of the workspace (that may or may not contain an obstacle) is completely invisible to the robot. However, as shown in the following experimental datasets and in the accompanying video of [202], this was not an important issue that prohibited experimental success.

Experimental Parameters

For the experiments reported in this Section, we use a wall following offset $\epsilon = 65$cm, an object detection threshold $\delta_{object} = 60$cm, an angular precision of 12° for successful alignment with each of the objects, a linear gain $k_l = 0.8$, an angular gain $k_a = 0.01$ and a maximum allowable LIDAR range of $R = 3$m. The stool-objects and the robot are treated as disks of radius $r = r_i = 0.2$m and we discretize the paths provided by the deliberative layer with a resolution of 1cm. Finally, the δ values (precision tolerances for landing zones) used for the MoveToObject, PositionObject and Move symbolic actions are 20cm, 40cm and 45cm respectively.

5.4.2 Task #1 - Single Object Positioning

In Fig. 5.7, we document the ability of the reactive layer's abstract unicycle control outputs (Section 5.2.2) to drive the gait layer's hybrid self-manipulation dynamics (Section 5.2.3) to follow the paths and manipulation directives given by the deliberative layer (Section 5.2.1). Starting from an initial position, the robot has to move to an object, mount it, push it to a desired location, dismount from it and then move to a predefined location. In order

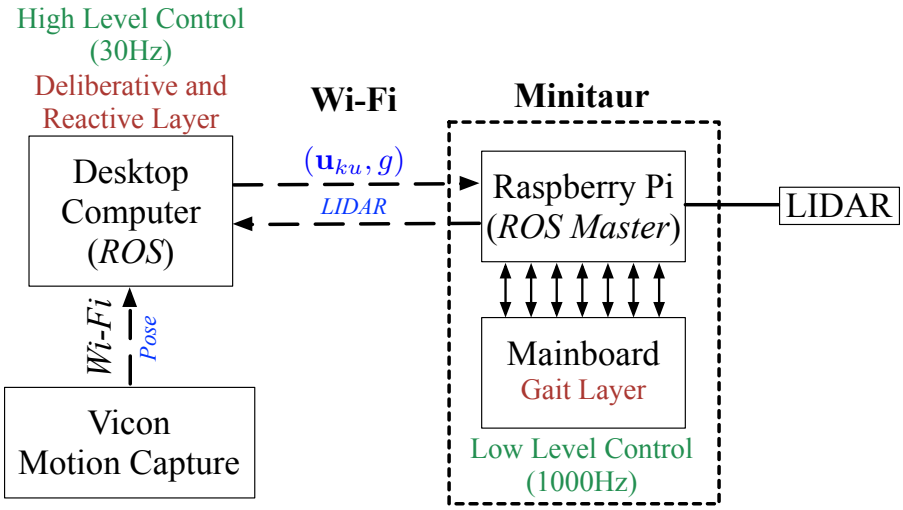

Figure 5.6: The system architecture, based on ROS, used for the experiments.

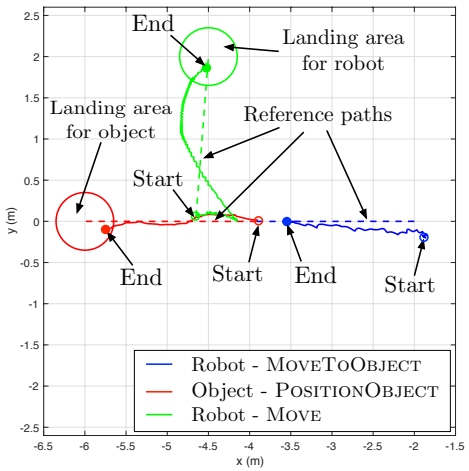

Figure 5.7: Task #1 - No Obstacles (Section 5.4.2): Vicon data showing the robot successfully following paths provided by the deliberative layer (dotted line segments): the robot has to approach (and then mount) the object (action MoveToObject), push the object inside a desired landing area (action PositionObject) and (first dismount) then retire to move to a predefined position (action Move), while following the reference paths (dotted lines).

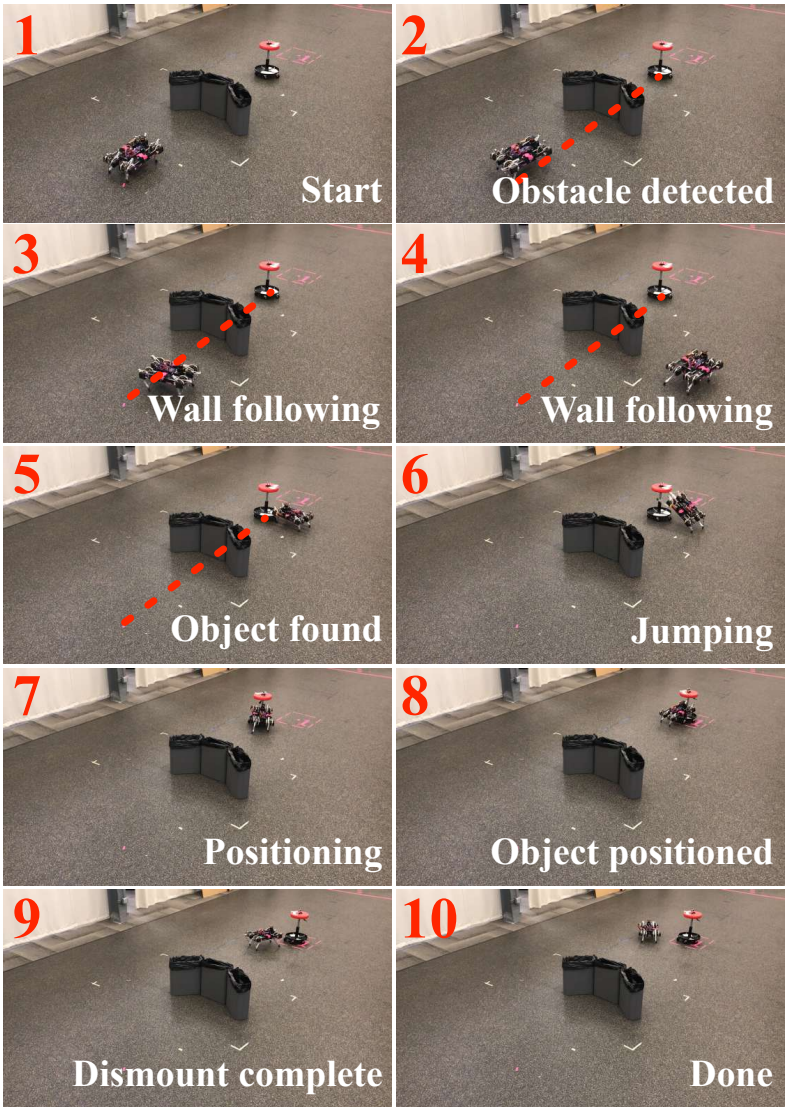

Figure 5.8: Task #1 - Unanticipated Obstacle (Section 5.4.2): The reactive layer allows for successful task completions even in the presence of non-convex obstacles, that have not been accounted for by the deliberative layer. The red dashed line represents the original (blocked by the obstacle) path given by the deliberative planner, associated with the action MoveToObject.

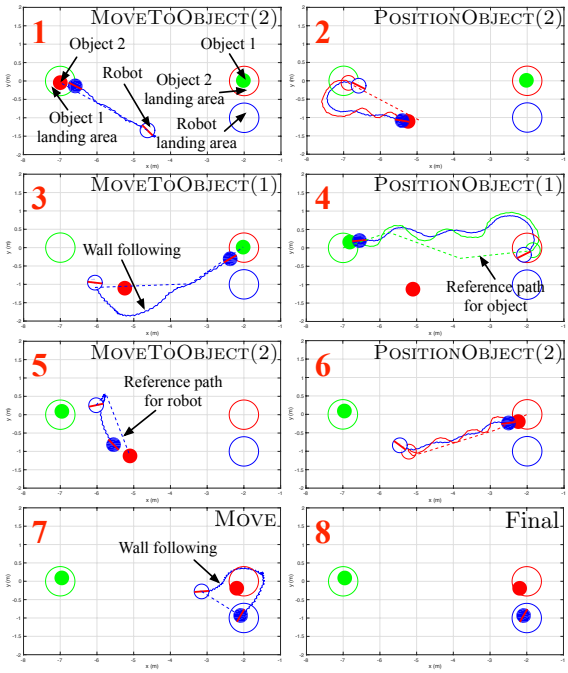

Figure 5.9: Task #2 (Section 5.4.3): Vicon data showing Minitaur swapping the positions of two objects. The dashed lines represent the reference paths for the robot or for the objects, provided by the deliberative layer. Non-filled and filled circles depict the start and end positions for each action execution. Any discrepancies of the final trajectories with the reference paths are caused by the controller's reactive nature and do not affect task completion.

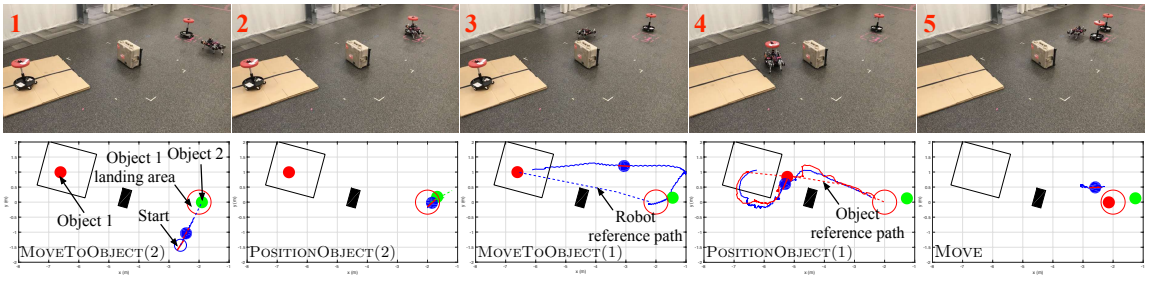

Figure 5.10: Task #3 (Section 5.4.4): Consecutive snapshots from a successful completion of a task where the robot must move an object that blocks the desired location of another object, highlighting the robustness of the approach. Apart from the presence of a convex obstacle (depicted in black) and terrain irregularities in the form of a 4cm-tall platform (depicted by a solid black line), the robot loses track of its pose estimation due to unfortunate network delays while executing MoveToObject(1). However, with the successful coordination of the reactive and the gait layer, it manages to find the reference path again once it reconnects. Also, as shown in the accompanying video[3] (and discernible from the relatively large oscillations of the robot's path in frame 4), although the wheels of the stool get caught by the platform during PositionObject(1), the persistence of the reactive layer allows for successful task completion while avoiding unexpected obstacles.

to validate the performance of the wall following law, presented in Section 5.3, a similar experiment is repeated, with the robot having to avoid a non-convex obstacle blocking its path to the object. As shown in Fig. 5.8 and in the accompanying video[3], the task is successfully completed, using the wall following algorithm.

5.4.3 Task #2 - Swapping Object Positions

The second task is more demanding for the deliberative planner, since the robot has to successfully swap the positions of two objects and then move to a "nest" location. As expected, the deliberative planner outputs a plan which includes an intermediate position for one of the objects. Using the reactive layer, the robot completes this task, as shown in Fig. 5.9 and in the accompanying video[3]. Notice how the robot switches to wall following when necessary and avoids any obstacles that block its path. The gait layer successfully executes the commands provided by the reactive layer.

5.4.4 Task #3 - Object Blocking the Position of Another Object

Finally, in the third set of experiments, we explore a similar task where the robot has to move an object in a location occupied by another object. We demonstrate several successful trials in the corresponding video[3], but here we focus on a special case where the online execution is incommoded by the presence of an obstacle and terrain irregularities, shown in Fig. 5.10. The robot also has to face other unfortunate events, such as network delays and getting the wheels of the stool stuck in the platform's step, but eventually completes the task. This illustrates the role of the reactive layer whose "persistence" can handle changes in the environment not predicted beforehand. It also highlights the value of legged over wheeled locomotion when "mobipulation" in unstructured environments with rough terrain is needed. We hope to report more on that in the future.

[3] `https://youtu.be/p0TcxosbOe0`

Part III

Reactive Navigation in Unfamiliar Semantic Environments

Chapter 6

Reactive Navigation in Partially Known Non-Convex Environments Cluttered with Star-Shaped Obstacles

This Chapter presents a provably correct method for robot navigation in 2D environments cluttered with familiar but unexpected non-convex, star-shaped obstacles as well as completely unknown, convex obstacles. We presuppose a limited range onboard sensor, capable of recognizing, localizing and (leveraging ideas from constructive solid geometry) generating online from its catalogue of the familiar, non-convex shapes an implicit representation of each one. These representations underlie an online change of coordinates to a completely convex model planning space wherein a previously developed online construction yields a provably correct reactive controller that is pulled back to the physically sensed representation to generate the actual robot commands. We extend the construction to differential drive robots, and suggest the empirical utility of the proposed control architecture using both formal proofs and numerical simulations.

The Chapter is organized as follows. Section 6.1 describes the problem and establishes our assumptions. Section 6.2 describes the physical, mapped and model planning layers used in the constructed diffeomorphism between the mapped and model layers, whose properties

are established next. Based on these results, Section 6.3 describes our control approach both for fully actuated and differential drive robots, and Section 6.4 presents a variety of illustrative numerical studies. Finally, the reader is referred to Appendix A for a sketch of the ideas from computational geometry [176] underlying our modular construction of implicit representations of polygonal obstacles.

6.1 Problem Formulation

We consider a disk-shaped robot with radius $r > 0$, centered at $\mathbf{x} \in \mathbb{R}^2$, navigating a closed, compact workspace $\mathcal{W} \subset \mathbb{R}^2$, with known convex boundary $\partial\mathcal{W}$. The robot is assumed to possess a sensor with fixed range R, capable of recognizing "familiar" objects, as well as estimating the distance of the robot to nearby obstacles[1].

The workspace is cluttered by an unknown number of fixed, disjoint obstacles, denoted by $\mathcal{O} := (O_1, O_2, \ldots)$. We adopt the notation in [7] and define the *freespace* as

$$\mathcal{F} := \left\{ \mathbf{x} \in \mathcal{W} \,\middle|\, \overline{\mathsf{B}(\mathbf{x}, r)} \subseteq \mathcal{W} \setminus \bigcup_i O_i \right\} \tag{6.1}$$

where $\mathsf{B}(\mathbf{x}, r)$ is the open ball centered at $\mathbf{x}$ with radius r, and $\overline{\mathsf{B}(\mathbf{x}, r)}$ denotes its closure. To simplify our notation, we neglect the robot dimensions, by dilating each obstacle in $\mathcal{O}$ by r, and assume that the robot operates in $\mathcal{F}$. We denote the set of dilated obstacles by $\tilde{\mathcal{O}}$.

Although none of the positions of any obstacles in $\tilde{\mathcal{O}}$ are à-priori known, a subset $\tilde{\mathcal{O}}^* \subseteq \tilde{\mathcal{O}}$ of these obstacles is assumed to be "familiar" in the sense of having an à-priori known, readily recognizable star-shaped geometry [164] (i.e., belonging to a known catalogue of star-shaped *geometry classes*), which the robot can efficiently identify and localize instantaneously from online sensory measurement. Although the implementation of such a sensory apparatus lies well beyond the scope of the present Section (but revisited in Chapter 7), recent work on semantic SLAM [30] provides an excellent example with empirically demonstrated technol-

[1] We refer the reader to an example of existing technology [144] generating 2D LIDAR scans from 3D point clouds for such an approach.

ogy for achieving this need for localizing, identifying and keeping track of all the familiar obstacles encountered in the otherwise unknown semantic environment. The à-priori unknown center of each catalogued star-shaped obstacle $\tilde{O}_i^*$ is denoted $\mathbf{x}_i^*$. Similarly to [165], each star-shaped obstacle $\tilde{O}_i^* \in \tilde{\mathcal{O}}^*$ can be described by an *obstacle function*, a real-valued map providing an implicit representation of the form

$$\tilde{O}_i^* = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \beta_i(\mathbf{x}) \leq 0\} \tag{6.2}$$

which the robot must construct online from the catalogued geometry, after it has localized $\tilde{O}_i^*$. The remaining obstacles $\tilde{\mathcal{O}}_{convex} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{O}}^*$ are are assumed to be strictly convex but are in all other regards (location and specific shape) completely unknown to the robot, while nevertheless satisfying a curvature condition given in [7, Assumption 2].

For the obstacle functions, we require the technical assumptions introduced in [165, Appendix III], outlined as follows.

Assumption 6.1. *The obstacle functions satisfy the following requirements*

1. *For each $\tilde{O}_i^* \in \tilde{\mathcal{O}}^*$, there exists $\varepsilon_{1i} > 0$ such that for any two obstacles $\tilde{O}_i^*, \tilde{O}_j^* \in \tilde{\mathcal{O}}^*$*

$$\{\mathbf{x} \,|\, \beta_i(\mathbf{x}) \leq \varepsilon_{1i}\} \bigcap \{\mathbf{x} \,|\, \beta_j(\mathbf{x}) \leq \varepsilon_{1j}\} = \emptyset \tag{6.3}$$

   i.e., the "thickened boundaries" of any two stars still do not overlap.

2. *For each $\tilde{O}_i^* \in \tilde{\mathcal{O}}^*$, there exists $\varepsilon_{2i} > 0$ such that the set $\{\mathbf{x} \,|\, \beta_i(\mathbf{x}) \leq \varepsilon_{2i}\}$ does not contain the goal $\mathbf{x}_d \in \mathcal{F}$ and does not intersubsect with any other obstacle in $\tilde{\mathcal{O}}_{convex}$.*

3. *For each obstacle function β_i, there exists a pair of positive constants $(\delta_i, \varepsilon_{3i})$ satisfying the inner product condition*[2]

$$(\mathbf{x} - \mathbf{x}_i^*)^\top \nabla \beta_i(\mathbf{x}) \geq \delta_i \tag{6.4}$$

[2] A brief discussion on this condition is given in Appendix A.1.

for all $\mathbf{x} \in \mathbb{R}^2$ *such that* $\beta_i(\mathbf{x}) \leq \varepsilon_{3i}$.

For each obstacle $\tilde{O}_i^* \in \tilde{\mathcal{O}}^*$, we then define $\varepsilon_i = \min\{\varepsilon_{1i}, \varepsilon_{2i}, \varepsilon_{3i}\}$. Finally, we will assume that the range of the sensor R satisfies $R >> \varepsilon_i$ for all i.

Based on these assumptions and further positing first-order, fully-actuated robot dynamics $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x})$, the problem consists of finding a Lipschitz continuous controller $\mathbf{u} : \mathcal{F} \to \mathbb{R}^2$, that leaves the freespace $\mathcal{F}$ positively invariant and asymptotically steers almost all configurations in $\mathcal{F}$ to the given goal $\mathbf{x}_d \in \mathcal{F}$.

6.2 Multi-layer Representation of the Environment and Its Associated Transformations

In this Section, we introduce associated notation for, and transformations between three distinct representations of the environment that we will refer to as planning "layers" and use in the construction of our algorithm. Fig. 6.1 illustrates the role of these layers and the transformations that relate them in constructing and analyzing a realtime generated vector field that guarantees safe passage to the goal. The new technical contribution is an adaptation of the methods of [165] to the construction of a diffeomorphism, $\mathbf{h}$, where the requirement for fast, online performance demands an algorithm that is as simple as possible and with few tunable parameters. Hence, since the reactive controller in [7], also presented in Section 3.2, is designed to (provably) handle convex shapes, sensed obstacles not recognized by the semantic SLAM process are simply assumed to be convex (implemented by designing $\mathbf{h}$ to resolve to the identity transformation in the neighborhood of "unfamiliar" objects) and the control response defaults to that prior construction.

6.2.1 Description of Planning Layers

Physical Layer

The *physical layer* is a complete description of the geometry of the unknown actual world and while inaccessible to the robot is used for purposes of analysis. It describes the actual workspace $\mathcal{W}$, punctured with the obstacles $\mathcal{O}$. This gives rise to the freespace $\mathcal{F}$, given

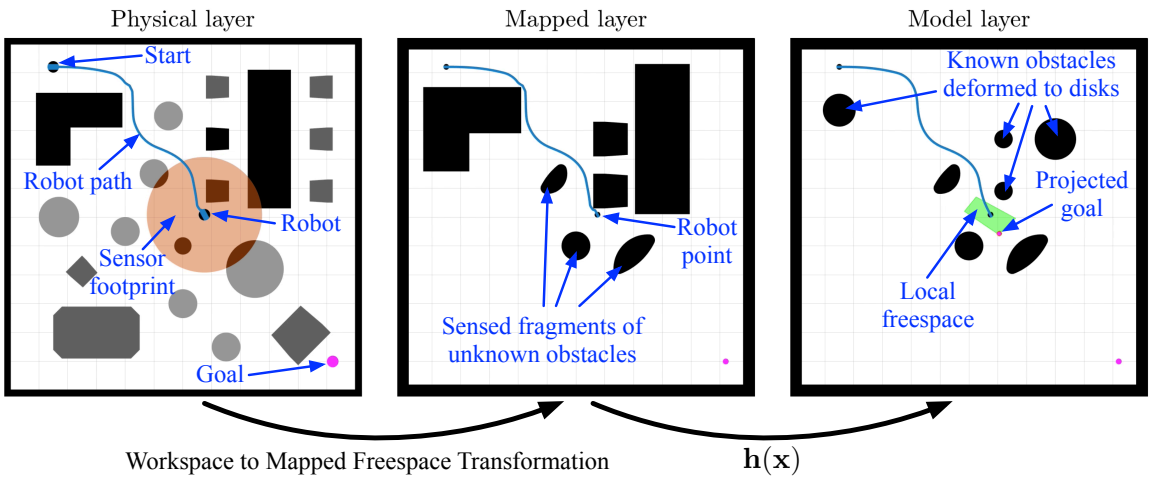

Figure 6.1: Snapshot Illustration of Key Ideas in Chapter 6. The robot in the physical layer (left frame, depicting in blue the robot's placement in the workspace along with the prior trajectory of its centroid) containing both familiar objects of known geometry but unknown location (dark grey) and unknown obstacles (light grey), moves towards a goal and discovers obstacles (black) with an onboard sensor of limited range (orange disk). These obstacles are localized and stored permanently in the mapped layer (middle frame, depicting in blue the robot's placement as a point in freespace rather than its body in the workspace) if they have familiar geometry or temporarily, with just the corresponding sensed fragments, if they are unknown. An online map $\mathbf{h}(\mathbf{x})$ is then constructed (Section 6.2), from the mapped layer to a geometrically simple model layer (right frame, now depicting the robot's placement and prior tractory amongst the $\mathbf{h}$-deformed convex images of the mapped obstacles). A doubly reactive control scheme for convex environments [7] (Section 3.2) defines a vector field on the model layer which is pulled back in realtime through the diffeomorphism to generate the input in the physical layer (Section 6.3).

in (6.1), consisting of all placements of the robot's centroid that entail no intersections of its body with any obstacles. The robot navigates this layer, and discovers and localizes new obstacles, which are then stored in its *semantic map* if their geometry is familiar.

Mapped Layer

The *mapped layer* $\mathcal{F}_{map}$ has the same boundary as $\mathcal{F}$ (i.e. $\partial\mathcal{F}_{map} := \partial\mathcal{F}$) and records the robot's evolving information about the environment aggregated from the raw sensor data about the observable portions of $N \geq 0$ unrecognized (and therefore, presumed convex) obstacles $\{\tilde{O}_1, \ldots, \tilde{O}_N\} \subseteq \tilde{\mathcal{O}}_{convex}$, together with the inferred star centers $\mathbf{x}_j^*$ and obstacle functions β_j of $M \geq 0$ star-shaped obstacles $\{\tilde{O}_1^*, \ldots, \tilde{O}_M^*\} \subseteq \tilde{\mathcal{O}}^*$, that are instantiated at the moment the sensory data triggers the "memory" that identifies and localizes a familiar obstacle. It is important to note that the star environment is constantly updated, both by discovering and storing new star-shaped obstacles in the semantic map and by discarding

old information and storing new information regarding obstacles in $\tilde{\mathcal{O}}_{convex}$. In this representation, the robot is treated as a point particle, since all obstacles are dilated by r in the passage from the workspace to the freespace representation of valid placements.

Model Layer

The *model layer* $\mathcal{F}_{model}$ has the same boundary as $\mathcal{F}$ (i.e. $\partial\mathcal{F}_{model} := \partial\mathcal{F}$) and consists of a collection of M Euclidean disks, each centered at one of the mapped star centers, $\mathbf{x}_j^*, j = 1, \ldots, M$, and copies of the sensed fragments of the N unrecognized visible convex obstacles in $\tilde{\mathcal{O}}_{convex}$. The radii $\{\rho_1, \ldots, \rho_M\}$ of the M disks are chosen so that $\overline{\mathsf{B}\left(\mathbf{x}_j^*, \rho_j\right)} \subseteq \{\mathbf{x} \,|\, \beta_j(\mathbf{x}) < 0\}$, as in [165].

This metric convex sphere world comprises the data generating the doubly reactive algorithm of Section r3.2, which will be applied to the physical robot via the online generated change of coordinates between the mapped layer and the model layer to be now constructed.

6.2.2 Description of the C^∞ Switches

In order to simplify the diffeomorphism construction, we depart from the construction of analytic switches [164] and rely instead on the C^∞ function $\zeta : \mathbb{R} \to \mathbb{R}$ [78] described by

$$\zeta(\chi) = \begin{cases} e^{-1/\chi}, & \chi > 0 \\ 0, & \chi \leq 0 \end{cases} \tag{6.5}$$

with derivative

$$\zeta'(\chi) = \begin{cases} \frac{\zeta(\chi)}{\chi^2}, & \chi > 0 \\ 0, & \chi \leq 0 \end{cases} \tag{6.6}$$

Based on that function, we can then define the C^∞ switches for each star-shaped obstacle $\tilde{O}_j^*$ in the semantic map as

$$\sigma_j(\mathbf{x}) = \eta_j \circ \beta_j(\mathbf{x}), \quad j = 1, \ldots, M \tag{6.7}$$

with $\eta_j(\chi) = \zeta(\varepsilon_j - \chi)/\zeta(\varepsilon_j)$ and ε_j given according to Assumption 6.1. The gradient of the switch σ_j is given by

$$\nabla\sigma_j(\mathbf{x}) = (\eta_j' \circ \beta_j(\mathbf{x})) \cdot \nabla\beta_j(\mathbf{x}) \tag{6.8}$$

Finally, we define

$$\sigma_d(\mathbf{x}) = 1 - \sum_{j=1}^{M} \sigma_j(\mathbf{x}) \tag{6.9}$$

Using the above construction, it is easy to see that $\sigma_j(\mathbf{x}) = 1$ on the boundary of the j-th obstacle and $\sigma_j(\mathbf{x}) = 0$ when $\beta_j(\mathbf{x}) > \varepsilon_j$ for each $j = 1, \ldots, M$. Based on Assumption 6.1 and the choice of ε_j for each j, we are, therefore, led to the following results.

Lemma 6.1. *At any point $\mathbf{x} \in \mathcal{F}_{map}$, at most one of the switches $\{\sigma_1, \ldots, \sigma_M\}$ can be nonzero.*

Corollary 6.1. *The set $\{\sigma_1, \ldots, \sigma_M, \sigma_d\}$ defines a partition of unity over $\mathcal{F}_{map}$.*

6.2.3 Description of the Star Deforming Factors

The deforming factors are the functions $\nu_j(\mathbf{x}) : \mathcal{F}_{map} \to \mathbb{R}, j = 1, \ldots, M$, responsible for transforming each star-shaped obstacle into a disk in $\mathbb{R}^2$. Once again, we use here a slightly different construction than [164], in that the value of each deforming factor ν_j at a point $\mathbf{x}$ does not depend on the value of $\beta_j(\mathbf{x})$. Namely, the deforming factors are given based on the desired final radii $\rho_j, j = 1, \ldots, M$ as

$$\nu_j(\mathbf{x}) = \frac{\rho_j}{||\mathbf{x} - \mathbf{x}_j^*||} \tag{6.10}$$

We also get

$$\nabla\nu_j(\mathbf{x}) = -\frac{\rho_j}{||\mathbf{x} - \mathbf{x}_j^*||^3}(\mathbf{x} - \mathbf{x}_j^*) \tag{6.11}$$

6.2.4 The Map Between the Mapped and the Model Layer

Construction

The map for M star-shaped obstacles centered at $\mathbf{x}_j^*, j = 1, \ldots, M$ is described by a function $\mathbf{h} : \mathcal{F}_{map} \to \mathcal{F}_{model}$ given by

$$\mathbf{h}(\mathbf{x}) = \sum_{j=1}^{M} \sigma_j(\mathbf{x}) \left[\nu_j(\mathbf{x})(\mathbf{x} - \mathbf{x}_j^*) + \mathbf{x}_j^*\right] + \sigma_d(\mathbf{x})\mathbf{x} \tag{6.12}$$

Note that the N visible convex obstacles $\{\tilde{O}_1, \ldots, \tilde{O}_N\} \subseteq \tilde{\mathcal{O}}_{convex}$ are not considered in the construction of the map. Since the reactive controller used in the model space $\mathcal{F}_{model}$ can handle convex obstacles and there is enough separation between convex and star-shaped obstacles according to Assumption 6.1-(b), we can "transfer" the geometry of those obstacles directly in the model space using the identity transformation.

Finally, note that Assumption 6.1-(b) implies that $\mathbf{h}(\mathbf{x}_d) = \mathbf{x}_d$, since the target location is assumed to be sufficiently far from all star-shaped obstacles.

Based on the construction of the map $\mathbf{h}$, the jacobian $D_{\mathbf{x}}\mathbf{h}$ at any point $\mathbf{x} \in \mathcal{F}_{map}$ is given by

$$D_{\mathbf{x}}\mathbf{h} = \sum_{j=1}^{M} \left\{\sigma_j(\mathbf{x})\nu_j(\mathbf{x})\mathbf{I} + (\mathbf{x} - \mathbf{x}_j^*)\left[\sigma_j(\mathbf{x})\nabla\nu_j(\mathbf{x})^\top + (\nu_j(\mathbf{x}) - 1)\nabla\sigma_j(\mathbf{x})^\top\right]\right\} + \sigma_d(\mathbf{x})\mathbf{I} \tag{6.13}$$

Qualitative Properties of the Map

We first verify that the construction is a smooth change of coordinates between the mapped and the model layers.

Lemma 6.2. *The map* $\mathbf{h}$ *from* $\mathcal{F}_{map}$ *to* $\mathcal{F}_{model}$ *is smooth.*

Proof. Included in Appendix C.4.1. □

Proposition 6.1. *The map* $\mathbf{h}$ *is a* C^∞ *diffeomorphism between* $\mathcal{F}_{map}$ *and* $\mathcal{F}_{model}$*.*

Proof. Included in Appendix C.4.1. □

Implicit representation of obstacles

To implement the diffeomorphism between $\mathcal{F}_{map}$ and $\mathcal{F}_{model}$, shown in (6.12), we rely on the existence of a smooth obstacle function $\beta_j(\mathbf{x})$ for each star-shaped obstacle $j = 1, \ldots, M$ stored in the semantic map. Since recently developed technology [93, 106, 148], revisited in Chapter 7, provides means of performing obstacle identification in the form of triangular meshes, in this work we focus on polygonal obstacles on the plane and derive implicit representations using so called "R-functions" from the constructive solid geometry literature [176]. In Appendix A.1, we describe the method used for the construction of such implicit functions for polygonal obstacles that have the desired property of being analytic everywhere except for the polygon vertices. For the construction, we assume that the sensor has already identified, localized and included each discovered star-shaped obstacle in $\mathcal{F}_{map}$; i.e., it has determined its pose in $\mathcal{F}_{map}$, given as a rotation $\mathbf{R}_j$ of its vertices on the plane followed by a translation of its center $\mathbf{x}_j^*$, and that the corresponding polygon has already been dilated by r for inclusion in $\mathcal{F}_{map}$.

6.3 Reactive Controller

6.3.1 Reactive Controller for Fully Actuated Robots

Construction

First, we consider a fully actuated particle with state $\mathbf{x} \in \mathcal{F}_{map}$, whose dynamics are described by

$$\dot{\mathbf{x}} = \mathbf{u} \tag{6.14}$$

The dynamics of the fully actuated particle in $\mathcal{F}_{model}$ with state $\mathbf{y} \in \mathcal{F}_{model}$ are described by $\dot{\mathbf{y}} = \mathbf{v}(\mathbf{y})$ with the control $\mathbf{v}(\mathbf{y})$ given in [7] as

$$\mathbf{v}(\mathbf{y}) = -k \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{x}_d)\right) \tag{6.15}$$

Here, the convex *local freespace* for $\mathbf{y}$, $\mathcal{LF}(\mathbf{y}) \subset \mathcal{F}_{model}$, is defined as in [7, Eqn. (30)] (see (3.5) in Chapter 3). Using the diffeomorphism construction in (6.12) and its jacobian in (6.13), we construct our controller as the vector field $\mathbf{u} : \mathcal{F}_{map} \to T\mathcal{F}_{map}$ given by

$$\mathbf{u}(\mathbf{x}) = [D_{\mathbf{x}}\mathbf{h}]^{-1} \cdot (\mathbf{v} \circ \mathbf{h}(\mathbf{x})) \tag{6.16}$$

Qualitative Properties

First of all, if the range of the virtual LIDAR sensor used to construct $\mathcal{LF}(\mathbf{y})$ in the model layer is smaller than R, the vector field $\mathbf{u}$ is Lipschitz continuous since $\mathbf{v}(\mathbf{y})$ is shown to be Lipschitz continuous in [7] and $\mathbf{y} = \mathbf{h}(\mathbf{x})$ is a smooth change of coordinates. We are led to the following result.

Corollary 6.2. *The vector field $\mathbf{u} : \mathcal{F}_{map} \to T\mathcal{F}_{map}$ generates a unique continuously differentiable partial flow.*

To ensure completeness (i.e. absence of finite time escape through boundaries in $\mathcal{F}_{map}$) we must verify that the robot never collides with any obstacle in the environment, i.e., leaves its freespace positively invariant.

Proposition 6.2. *The freespace $\mathcal{F}_{map}$ is positively invariant under the law* (6.16).

Proof. Included in Appendix C.4.2. □

Lemma 6.3. *1. The set of stationary points of control law* (6.16) *is given as*

$$\{\mathbf{x}_d\} \bigcup \{\mathbf{h}^{-1}(\mathbf{s}_j)\}_{j\in\{1,\dots,M\}} \bigcup_{i=1}^{N} \mathcal{G}_i$$

where

$$\mathbf{s}_j = \mathbf{x}_j^* - \rho_j \frac{\mathbf{x}_d - \mathbf{x}_j^*}{||\mathbf{x}_d - \mathbf{x}_j^*||} \tag{6.17a}$$

$$\mathcal{G}_i := \left\{ \mathbf{q} \in \mathcal{F}_{map} \,\middle|\, d(\mathbf{q}, O_i) = r, \frac{(\mathbf{q} - \Pi_{\overline{O}_i}(\mathbf{q}))^\top (\mathbf{q} - \mathbf{x}_d)}{||\mathbf{q} - \Pi_{\overline{O}_i}(\mathbf{q})||\,||\mathbf{q} - \mathbf{x}_d||} = 1 \right\} \tag{6.17b}$$

with j spanning the M star-shaped obstacles in $\mathcal{F}_{map}$ and i spanning the N convex obstacles in $\mathcal{F}_{map}$.

2. *The goal $\mathbf{x}_d$ is the only locally stable equilibrium of control law* (6.16) *and all the other stationary points $\{\mathbf{h}^{-1}(\mathbf{s}_j)\}_{j\in\{1,\dots,M\}} \bigcup_{i=1}^{N} \mathcal{G}_i$, each associated with an obstacle, are nondegenerate saddles.*

Proof. Included in Appendix C.4.2. □

Proposition 6.3. *The goal location $\mathbf{x}_d$ is an asymptotically stable equilibrium of* (6.16)*, whose region of attraction includes the freespace $\mathcal{F}_{map}$ excepting a set of measure zero.*

Proof. Included in Appendix C.4.2. □

We can now immediately conclude the following central summary statement.

Theorem 6.1. *The reactive controller in* (6.16) *leaves the freespace $\mathcal{F}_{map}$ positively invariant, and its unique continuously differentiable flow, starting at almost any robot placement $\mathbf{x} \in \mathcal{F}_{map}$, asymptotically reaches the goal location $\mathbf{x}_d$, while strictly decreasing $||\mathbf{h}(\mathbf{x}) - \mathbf{x}_d||$ along the way.*

6.3.2 Reactive Controller for Differential Drive Robots

In this Section, we extend our reactive controller to the case of a differential drive robot, whose state is $\overline{\mathbf{x}} := (\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1 \subset SE(2)$, and its dynamics are given by

$$\dot{\overline{\mathbf{x}}} = \mathbf{B}(\psi)\overline{\mathbf{u}} \tag{6.18}$$

with $\mathbf{B}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top$ and $\overline{\mathbf{u}} = (v, \omega)$ with $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ the linear and angular input respectively, as discussed in Part II. We will follow a similar procedure to the fully actuated case; we begin by describing a smooth diffeomorphism $\overline{\mathbf{h}} : \mathcal{F}_{map} \times S^1 \to \mathcal{F}_{model} \times S^1$ and then we establish the results about the controller.

Construction and Properties of the $SE(2)$ Diffeomorphism

We construct our map $\overline{\mathbf{h}}$ from $\mathcal{F}_{map} \times S^1$ to $\mathcal{F}_{model} \times S^1$ as

$$\overline{\mathbf{y}} = (\mathbf{y}, \varphi) = \overline{\mathbf{h}}(\overline{\mathbf{x}}) := (\mathbf{h}(\mathbf{x}), \xi(\overline{\mathbf{x}})) \tag{6.19}$$

with $\overline{\mathbf{x}} = (\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1$, $\overline{\mathbf{y}} := (\mathbf{y}, \varphi) \in \mathcal{F}_{model} \times S^1$ and

$$\varphi = \xi(\overline{\mathbf{x}}) := \angle(\mathbf{e}(\overline{\mathbf{x}})) \tag{6.20}$$

Here, $\angle\mathbf{e} := \text{atan2}(e_2, e_1)$ and

$$\mathbf{e}(\overline{\mathbf{x}}) = \Pi_{\mathbf{y}} \cdot D_{\overline{\mathbf{x}}}\overline{\mathbf{h}} \cdot \mathbf{B}(\psi) \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = D_{\mathbf{x}}\mathbf{h} \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \tag{6.21}$$

with $\Pi_{\mathbf{y}}$ denoting the projection onto the first two components. The reason for choosing φ as in (6.20) will become evident in the next paragraph, in our effort to control the equivalent differential drive robot dynamics in $\mathcal{F}_{model}$.

Proposition 6.4. *The map $\overline{\mathbf{h}}$ in (6.19) is a C^∞ diffeomorphism from $\mathcal{F}_{map} \times S^1$ to $\mathcal{F}_{model} \times S^1$.*

Proof. Included in Appendix C.4.2. □

Construction of the Reactive Controller

Using (6.19), we can find the pushforward of the differential drive robot dynamics in (6.18) as

$$\dot{\overline{\mathbf{y}}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\varphi} \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} \mathbf{h}(\mathbf{x}) \\ \xi(\overline{\mathbf{x}}) \end{bmatrix} = \left[D_{\overline{\mathbf{x}}}\overline{\mathbf{h}} \circ \overline{\mathbf{h}}^{-1}(\overline{\mathbf{y}}) \right] \cdot \left(\mathbf{B} \circ \overline{\mathbf{h}}^{-1}(\overline{\mathbf{y}}) \right) \cdot \overline{\mathbf{u}} \tag{6.22}$$

Based on the above, we can then write

$$\dot{\overline{\mathbf{y}}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\varphi} \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} \mathbf{h}(\mathbf{x}) \\ \xi(\overline{\mathbf{x}}) \end{bmatrix} = \mathbf{B}(\varphi)\overline{\mathbf{v}} \tag{6.23}$$

with $\overline{\mathbf{v}} = (\hat{v}, \hat{\omega})$, and the inputs $(\hat{v}, \hat{\omega})$ related to (v, ω) through

$$\hat{v} = ||\mathbf{e}(\overline{\mathbf{x}})||\, v \tag{6.24}$$

$$\hat{\omega} = v D_{\mathbf{x}}\xi \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} + \frac{\partial \xi}{\partial \psi}\omega \tag{6.25}$$

with $D_{\mathbf{x}}\xi = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \end{bmatrix}$. The calculation of $D_{\mathbf{x}}\xi$ can be tedious, since it involves derivatives of elements of $D_{\mathbf{x}}\mathbf{h}$, and is included in Appendix B.1.

Hence, we have found equivalent differential drive robot dynamics, defined on $\mathcal{F}_{model} \times S^1$. The idea now is to use the control strategy in Section 3.2 for the dynamical system in (6.23) to find *reference inputs* $\hat{v}, \hat{\omega}$, and then use (6.24), (6.25) to find the *actual inputs* v, ω that achieve those reference inputs as

$$v = \frac{\hat{v}}{||\mathbf{e}(\overline{\mathbf{x}})||} \tag{6.26a}$$

$$\omega = \left(\frac{\partial \xi}{\partial \psi}\right)^{-1} \left(\hat{\omega} - v D_{\mathbf{x}}\xi \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \right) \tag{6.26b}$$

Namely, our reference inputs $\hat{v}$ and $\hat{\omega}$ inspired by [7, 12] (also see (3.8) - (3.9)) are given as[3]

[3]In (6.19), we construct a diffeomorphism $\overline{\mathbf{h}}$ between $\mathcal{F}_{map} \times S^1$ and $\mathcal{F}_{model} \times S^1$. However, for practical purposes, we deal only with one specific chart of S^1 in our control structure, described by the angles $(-\pi, \pi]$. As shown in [12], the discontinuity at $\pm\pi$ does not induce a discontinuity in our controller due to the use of the atan function in (6.27b). On the contrary, with the use of (6.27b) as in [7, 12], the robot never changes heading in $\mathcal{F}_{model}$, which implies that the generated trajectories both in $\mathcal{F}_{model}$ and (by the properties of the diffeomorphism $\overline{\mathbf{h}}$) in $\mathcal{F}_{map}$ have no cusps, even though the robot might change heading in $\mathcal{F}_{map}$ because of the more complicated nature of the function ξ in (6.20).

$$\hat{v} = -k \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y}) \cap H_\parallel}(\mathbf{x}_d) \right) \tag{6.27a}$$

$$\hat{\omega} = k \operatorname{atan} \left(\frac{\begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}^\top \left(\mathbf{y} - \frac{\Pi_{\mathcal{LF}(\mathbf{y}) \cap H_G}(\mathbf{x}_d) + \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{x}_d)}{2} \right)}{\begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}^\top \left(\mathbf{y} - \frac{\Pi_{\mathcal{LF}(\mathbf{y}) \cap H_G}(\mathbf{x}_d) + \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{x}_d)}{2} \right)} \right) \tag{6.27b}$$

with $k > 0$ a fixed gain, $\mathcal{LF}(\mathbf{y}) \subset \mathcal{F}_{model}$ the convex polygon defining the local freespace at $\mathbf{y} = \mathbf{h}(\mathbf{x})$, and $H_\parallel$ and H_G the lines defined in [7] (also see (3.12) - (3.13)) as

$$H_\parallel = \left\{ \mathbf{z} \in \mathcal{F}_{model} \,\middle|\, \begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}^\top (\mathbf{z} - \mathbf{y}) = 0 \right\} \tag{6.28}$$

$$H_G = \{ \alpha\mathbf{y} + (1-\alpha)\mathbf{x}_d \in \mathcal{F}_{model} \,|\, \alpha \in \mathbb{R} \} \tag{6.29}$$

Qualitative Properties

The properties of the differential drive robot control law given in (6.26) can be summarized in the following theorem.

Theorem 6.2. *The reactive controller for differential drive robots, given in* (6.26)*, leaves the freespace $\mathcal{F}_{map} \times S^1$ positively invariant, and its unique continuously differentiable flow, starting at almost any robot configuration $(\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1$, asymptotically steers the robot to the goal location $\mathbf{x}_d$, without increasing $||\mathbf{h}(\mathbf{x}) - \mathbf{x}_d||$ along the way.*

Proof. Included in Appendix C.4.2. □

6.4 Numerical Experiments

In this Section, we present numerical experiments that verify our formal results. All simulations were run in MATLAB using `ode45`, with control gain $k = 0.4$ and $p = 20$ for the

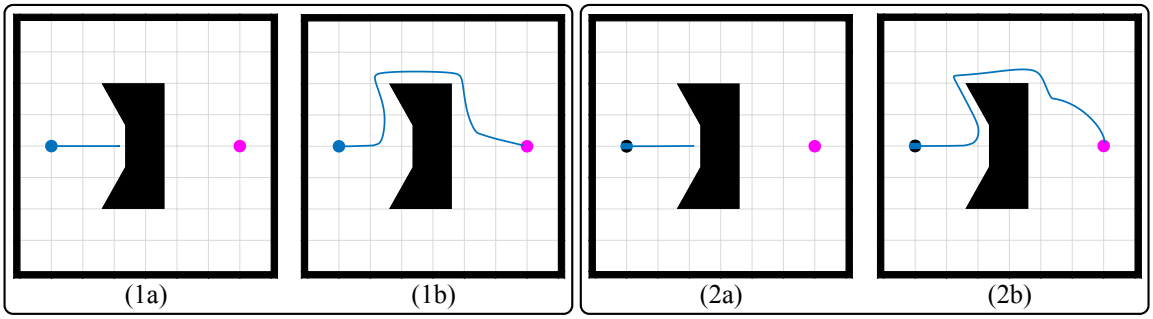

Figure 6.2: Navigation around a U-shaped obstacle: 1) Fully actuated particle: (a) Original doubly reactive algorithm [7], (b) Our algorithm, 2) Differential drive robot: (a) Original doubly reactive algorithm [7], (b) Our algorithm.

R-function construction. The reader is also referred to the video attachment of [200] for a visualization of the examples presented here and more numerical simulations[4].

6.4.1 Comparison with Original Doubly Reactive Algorithm

We begin with a comparison of our algorithm performance with the standalone version of the doubly reactive algorithm in [7], that we use in our construction. Fig. 6.2 demonstrates the basic limitation of this algorithm; in the presence of a non-convex obstacle or a flat surface, whose curvature violates [7, Assumption 2], the robot gets stuck in undesired local minima. On the contrary, our algorithm is capable of overcoming this limitation, on the premise that the robot can recognize the obstacle with star-shaped geometry at hand. The robot radius is 0.2m and the value of ε used for the obstacle is 0.3.

6.4.2 Navigation in a Cluttered Non-Convex Environment

In the next set of numerical experiments, we evaluate the performance of our algorithm in a cluttered environment, packed with instances of the same U-shaped obstacle, with star-shaped geometry, we use in Fig. 6.2. Both the fully actuated and the differential drive robot are capable of converging to the desired goal from a variety of initial conditions, as shown in Fig. 6.3. In the same figure, we also focus on a particular initial condition and include the trajectories observed in the physical, mapped and model layers. The robot radius is 0.25m and value of ε used for all the star-shaped obstacles in the environment is 0.3.

[4] `https://youtu.be/i-9AxWdal5s`

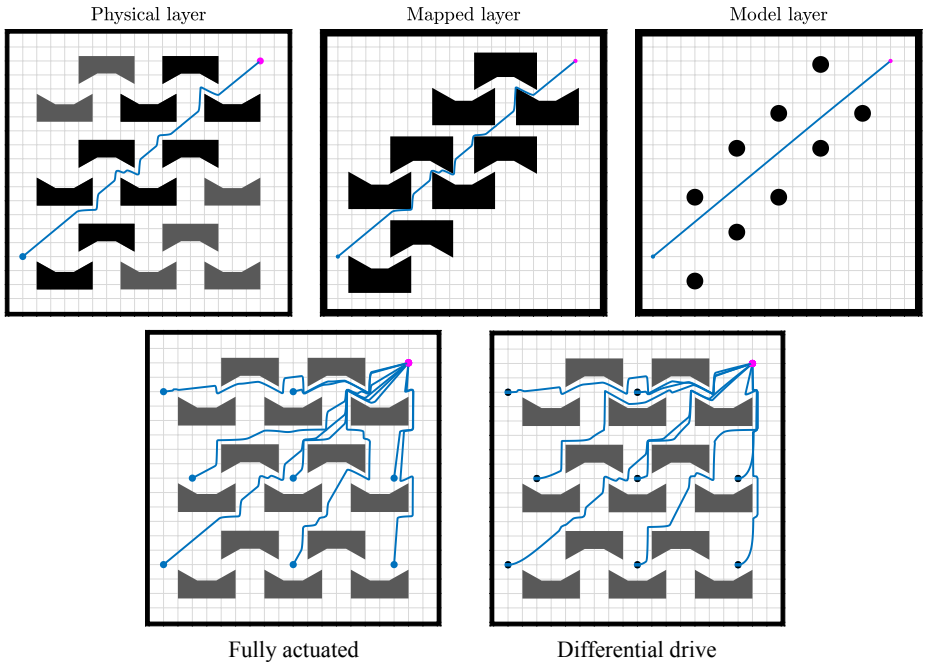

Figure 6.3: Navigation in a cluttered environment with U-shaped obstacles. Top - Trajectories in the physical, mapped and model layers from a particular initial condition. Bottom - Convergence to the goal from several initial conditions: left - fully actuated robot, right - differential drive robot.

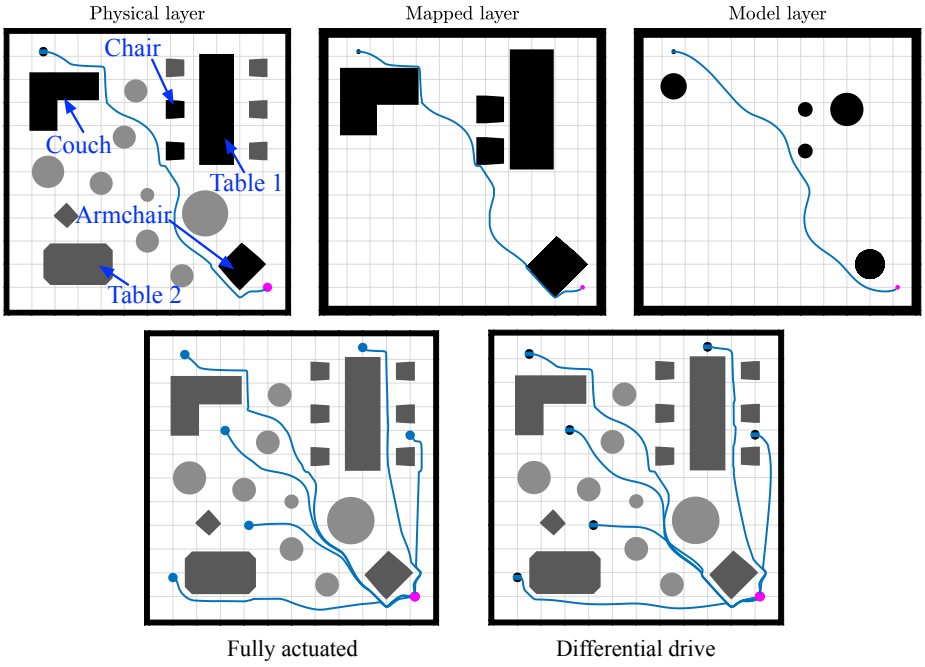

Figure 6.4: Navigating a room cluttered with known star-shaped and unknown convex obstacles. Top - Trajectories in the physical, mapped and model layers from a particular initial condition. Bottom - Convergence to the goal from several initial conditions: left - fully actuated robot, right - differential drive robot. Mapped obstacles are shown in black, known obstacles in dark grey and unknown obstacles in light grey.

6.4.3 Navigation Among Mixed Star-Shaped and Convex Obstacles

Finally, we report experiments in an environment cluttered with both star-shaped obstacles (with known geometry) and unknown convex obstacles. We consider a robot of radius 0.2m navigating a room towards a goal. The robot can recognize familiar star-shaped obstacles (e.g., the couch, tables, armchair, chairs) but is unaware of several other convex obstacles in the environment. Fig. 6.4 summarizes our results for several initial conditions. We also include trajectories observed in the physical, mapped and model layers during a single run. The value of ε used for all the star-shaped obstacles in the environment is 0.3.

Chapter 7

Reactive Navigation in Partially Familiar Planar Environments Using Semantic Perceptual Feedback

In this Chapter, we extend the results of Chapter 6 to solve the general planar navigation problem by recourse to an online reactive scheme that exploits recent advances in SLAM and visual object recognition to recast prior geometric knowledge in terms of an offline catalogue of familiar objects. The resulting vector field planner guarantees convergence to an arbitrarily specified goal, avoiding collisions along the way with fixed but arbitrarily placed instances from the catalogue as well as completely unknown fixed obstacles so long as they are strongly convex and well separated. We illustrate the generic robustness properties of such deterministic reactive planners as well as the relatively modest computational cost of this algorithm by supplementing an extensive numerical study with physical implementation on both a wheeled and legged platform in different settings.

The Chapter is organized as follows. Section 7.1 describes the problem and establishes our assumptions. Section 7.2 describes the physical, semantic, mapped and model planning spaces (summarized in Fig. 7.1) used in the diffeomorphism construction between the mapped and model spaces, whose properties are established next in Section 7.3. Section 7.4

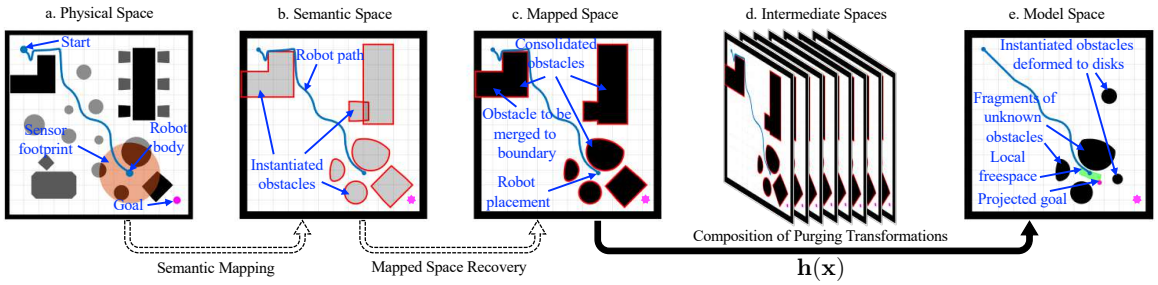

Figure 7.1: Snapshot Illustration of Key Realtime Computation and Associated Models related to Chapter 7: The robot moves in the physical space (a - Section 7.2.1), depicted as the blue trace of its centroid, toward a goal (pink) discovering along the way (black) both familiar objects of known geometry but unknown location (dark grey) and unknown obstacles (light grey), with an onboard sensor of limited range (orange disk). These obstacles are localized, dilated and stored permanently in the semantic space (b - Section 7.2.2) if they have familiar geometry, or temporarily, with just the corresponding sensed fragments, if they are unknown. The consolidated obstacles (resolved in real time from the unions of overlapping localized familiar obstacles), along with the sensed fragments of the unknown obstacles, are then stored in the mapped space (c - Section 7.2.3). A nonlinear change of coordinates, $\mathbf{h}(\mathbf{x})$, into a topologically equivalent but geometrically simplified model space (e - Section 7.2.4, depicting the robot's placement and prior trajectory amongst the $\mathbf{h}$-deformed convex images of the mapped obstacles) is computed instantaneously each time a new perceptual event instantiates more obstacles to be localized in the semantic space, thus redefining the mapped space. The map, $\mathbf{h}$, is a diffeomorphism, computed via composition of "purging" transformations between intermediate spaces (d - Section 7.3.2) that abstract the consolidated localized polygonal obstacles by successively pruning away their geometric details to yield topologically equivalent disks. A doubly reactive control scheme for convex environments [7] (Section 3.2) defines a vector field on the model space which is transformed in realtime through the diffeomorphism to generate the input in the physical space (Section 7.4).

provides the formal hybrid systems description framework and the correctness proofs for both a fully actuated (Theorem 7.3) and differential drive (Theorem 7.4) velocity controlled planar robot, comprising the central theoretical contribution of this Chapter.

Based on these results, Section 7.5 continues with a description of the implemented mapped space recovery and reactive planning algorithms, for both a fully actuated and a differential drive robot, shown in Fig. 7.2-(d),(e). Section 7.6 presents a variety of illustrative numerical studies, and Section 7.7 continues with a brief description of the experimental setup, realizing the deployed perception (relying on prior work and shown in Fig. 7.2-(a),(c)) and motion planning (Fig. 7.2-(d),(e)) algorithms on both the Turtlebot [194] and the Minitaur [65] robot. Finally, Section 7.8 continues with our experimental results.

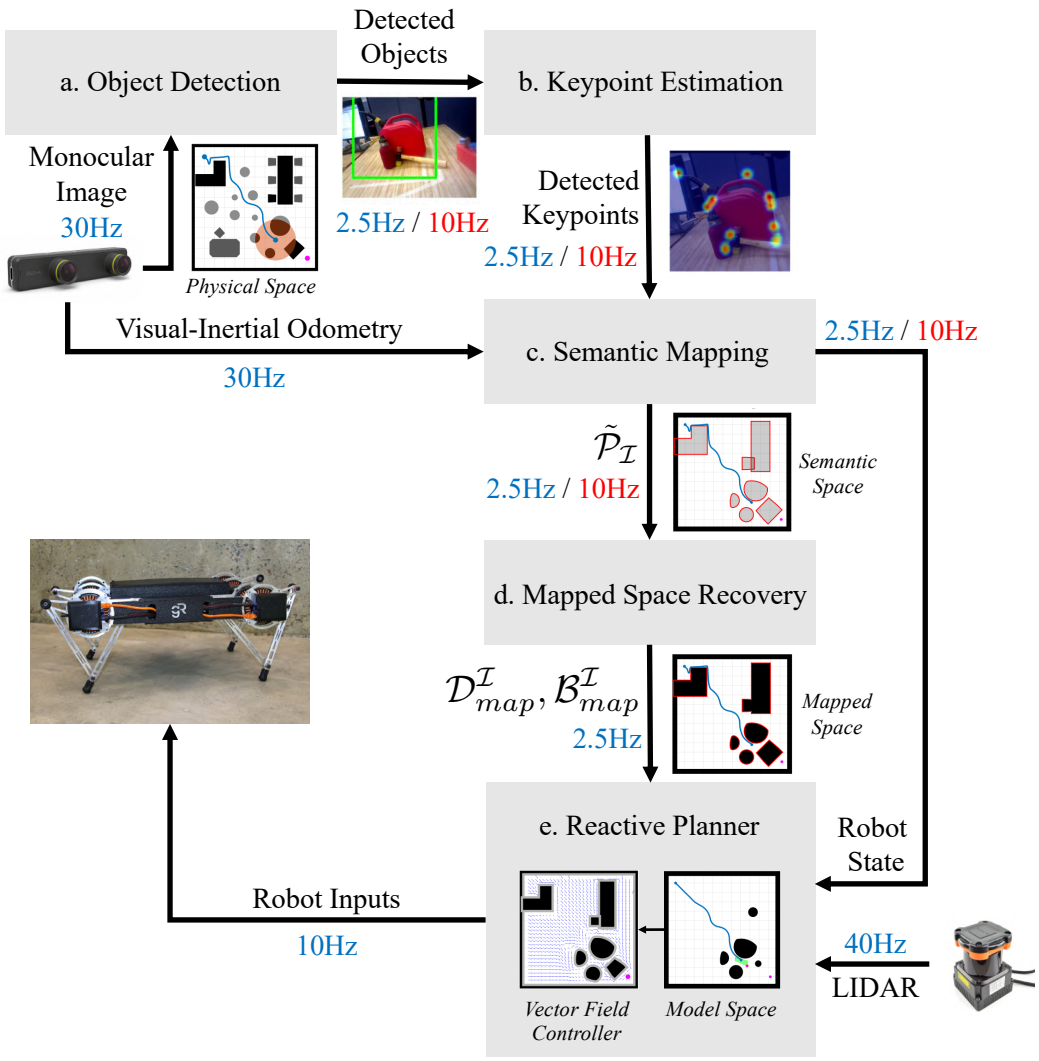

Figure 7.2: A summary of the online reactive planning architecture used in Chapter 7. Using the camera image, two separate neural network architectures (configured in serial and run either onboard at 2.5Hz, or offboard at 10Hz) (a) detect familiar obstacles [158] (Section 7.7.1) and (b) localize corresponding semantic keypoints [148] (Section 7.7.1). (c) The keypoint locations on the image and an egomotion estimate provided by visual inertial odometry are used by the semantic mapping module [30] (Section 7.7.2) to provide updated robot ($\mathbf{x}$) and obstacle poses ($\tilde{\mathcal{P}}_{\mathcal{I}}$) on the plane. (d) The mapped space tracking algorithm (Section 7.5.1 - Algorithm 7.1), run onboard at 2.5Hz, uses $\tilde{\mathcal{P}}_{\mathcal{I}}$ to generate the list of obstacles in the mapped space $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$. (e) The reactive planning module (Section 7.5.2 - Algorithm 7.2), run onboard at 10Hz, uses $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$, along with LIDAR data for unknown obstacles, to provide the robot inputs and close the control loop.

$\mathcal{W} \subset \mathbb{R}^2$	Closed, compact, polygonal workspace
$\mathcal{W}_e \subset \mathbb{R}^2$	Enclosing workspace (7.1)
$\mathcal{F} \subset \mathcal{W}$	Freespace (7.2)
$\mathcal{F}_e \subset \mathcal{W}_e$	Enclosing freespace (7.3)
$r \in \mathbb{R}$	Robot radius
$R \in \mathbb{R}$	Sensor range
$\mathbf{x}_d \in \mathcal{F}$	Goal location
$\tilde{\mathcal{O}} := \{\tilde{O}_1, \tilde{O}_2, \ldots\} \subseteq \mathbb{R}^2$	Set of fixed, disjoint obstacles
$\tilde{\mathcal{P}} := \{\tilde{P}_i\}_{i \in \mathcal{N}_\mathcal{P}} \subseteq \tilde{\mathcal{O}}$	Set of "familiar", polygonal obstacles, indexed by the set $\mathcal{N}_\mathcal{P} := \{1, \ldots, N_P\} \subset \mathbb{N}$
$\tilde{\mathcal{C}} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{P}} = \{\tilde{C}_i\}_{i \in \mathcal{N}_\mathcal{C}}$	Set of completely unknown obstacles, indexed by the set $\mathcal{N}_\mathcal{C} := \{1, \ldots, N_C\} \subset \mathbb{N}$
$\mathcal{O}, \mathcal{P}, \mathcal{C}$	Set of obstacles in $\tilde{\mathcal{O}}, \tilde{\mathcal{P}}, \tilde{\mathcal{C}}$ respectively, dilated by the robot radius, r

Table 7.1: Key symbols used throughout Chapter 7, associated with the Problem Formulation in Section 7.1. See also Table 7.2 for notation associated with the environment representation in Section 7.2, Table 7.3 for notation associated with the diffeomorphism construction in Section 7.3, and Table 7.4 for notation associated with our reactive controller in Section 7.4.

7.1 Problem Formulation

Similarly to Chapter 6, we consider a disk-shaped robot with radius $r > 0$, centered at $\mathbf{x} \in \mathbb{R}^2$, navigating a closed, compact, polygonal, potentially non-convex workspace $\mathcal{W} \subset \mathbb{R}^2$, with known outer boundary $\partial\mathcal{W}$, towards a target location $\mathbf{x}_d \in \mathcal{W}$. The robot is assumed to possess a sensor with fixed range R, capable of recognizing "familiar" objects, as well as estimating the distance of the robot to nearby obstacles[1]. We also define the *enclosing workspace*, as the convex hull of the closure of the workspace $\mathcal{W}$:

$$\mathcal{W}_e := \left\{ \mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \mathrm{Conv}(\overline{\mathcal{W}}) \right\} \tag{7.1}$$

The workspace is cluttered by a finite, unknown number of fixed, disjoint obstacles, denoted by $\tilde{\mathcal{O}} := \{\tilde{O}_1, \tilde{O}_2, \ldots\}$. By convention, the set $\tilde{\mathcal{O}}$ also includes potentially non-convex "intrusions" of the boundary of the physical workspace $\mathcal{W}$ into the enclosing workspace $\mathcal{W}_e$, that can be described as the connected components of $\mathcal{W}_e \backslash \mathcal{W}$. We again use the notation

[1]For our hardware implementation, this idealized sensor is reduced to a combination of a LIDAR for distance measurements to obstacles and a monocular camera for object recognition and pose identification.

in [7] and define the *freespace* as

$$\mathcal{F} := \left\{ \mathbf{x} \in \mathcal{W}_e \,\middle|\, \overline{\mathsf{B}(\mathbf{x}, r)} \subseteq \mathcal{W}_e \setminus \bigcup_i \tilde{O}_i \right\} \tag{7.2}$$

where $\mathsf{B}(\mathbf{x}, r)$ is the open ball centered at $\mathbf{x}$ with radius r, and $\overline{\mathsf{B}(\mathbf{x}, r)}$ denotes its closure. Similarly to the enclosing workspace, $\mathcal{W}_e$, we define the *enclosing freespace*, $\mathcal{F}_e$ as

$$\mathcal{F}_e := \left\{ \mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \mathrm{Conv}(\overline{\mathcal{F}}) \right\} \tag{7.3}$$

Although none of the positions of any obstacles in $\tilde{\mathcal{O}}$ are à-priori known, a subset $\tilde{\mathcal{P}} := \{\tilde{P}_i\}_{i \in \mathcal{N}_\mathcal{P}} \subseteq \tilde{\mathcal{O}}$ of these obstacles, indexed by $\mathcal{N}_\mathcal{P} := \{1, \ldots, N_P\} \subset \mathbb{N}$, is assumed to be "familiar" in the sense of having an à-priori known, readily recognizable, potentially non-convex, polygonal geometry (i.e., belonging to a known catalogue of *geometry classes*), which the robot can identify and localize instantaneously from online sensory measurement, as described in Section 7.7. We require that this subset also includes all connected components of $\mathcal{W}_e \backslash \mathcal{W}$. The remaining obstacles in $\tilde{\mathcal{C}} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{P}}$, indexed by $\mathcal{N}_\mathcal{C} := \{1, \ldots, N_C\} \subset \mathbb{N}$ are assumed to be strictly convex but are in all other regards (location and specific shape) completely unknown to the robot, while nevertheless satisfying a curvature condition given in [7, Assumption 2], and described as follows.

Assumption 7.1. *The Jacobian matrix* $\mathbf{J}_{\Pi_{\overline{\tilde{C}}_i}}(\mathbf{s}_i)$ *of the metric projection* $\Pi_{\overline{\tilde{C}}_i}(\mathbf{s}_i)$ *of any point* $\mathbf{s}_i \in \mathcal{G}_i$ *with*

$$\mathcal{G}_i := \left\{ \mathbf{s} \in \mathcal{F} \,\middle|\, d(\mathbf{s}, \tilde{C}_i) = r, \frac{(\mathbf{s} - \Pi_{\overline{\tilde{C}}_i}(\mathbf{s}))^\top (\mathbf{s} - \mathbf{x}_d)}{||\mathbf{s} - \Pi_{\overline{\tilde{C}}_i}(\mathbf{s})|| \cdot ||\mathbf{s} - \mathbf{x}_d||} = 1 \right\} \tag{7.4}$$

onto the associated obstacle $\tilde{C}_i \in \tilde{\mathcal{C}}$ *for all* $i \in \{1, \ldots, N_C\}$ *satisfies*

$$\mathbf{J}_{\Pi_{\overline{\tilde{C}}_i}}(\mathbf{s}_i) \prec \frac{||\mathbf{x}_d - \Pi_{\overline{\tilde{C}}_i}(\mathbf{s}_i)||}{r + ||\mathbf{x}_d - \Pi_{\overline{\tilde{C}}_i}(\mathbf{s}_i)||} \mathbf{I} \tag{7.5}$$

where $d(A, B) := \inf\{||\mathbf{a} - \mathbf{b}||, \mathbf{a} \in A, \mathbf{b} \in B\}$.

This condition is related to the control law described in Section 7.4, and was interpreted in [9] as the requirement for the convex obstacle $\tilde{C}_i$ to be contained in the enclosing ball of radius $(||\mathbf{s}_i - \mathbf{x}_d|| - r)$ centered at $\mathbf{x}_d$, for all i.

To simplify our notation, we neglect the robot dimensions, by dilating each obstacle in $\tilde{\mathcal{O}}$ by r, and assume that the robot operates in $\mathcal{F}$. We denote the set of dilated obstacles derived from $\tilde{\mathcal{O}}, \tilde{\mathcal{P}}$ and $\tilde{\mathcal{C}}$, by $\mathcal{O}, \mathcal{P}$ and $\mathcal{C}$ respectively. Since obstacles in $\tilde{\mathcal{P}}$ are polygonal, and dilations of polygonal obstacles are not in general polygonal, we approximate obstacles in $\mathcal{P}$ with conservative polygonal supersets. Note that since the set $\tilde{\mathcal{P}}$ is required to contain all connected components of $\mathcal{W}_e \backslash \mathcal{W}$, that describe non-convex "intrusions" of the boundary of the physical workspace $\mathcal{W}$ into the enclosing workspace $\mathcal{W}_e$, the set $\mathcal{P}$ is similarly required to contain the dilations of these intrusions. For obstacles in $\mathcal{C}$ we require the following separation assumptions, introduced in [7].

Assumption 7.2. *Each obstacle $C_i \in \mathcal{C}$ has a positive clearance $d(C_i, C_j) > 0$ from any obstacle $C_j \in \mathcal{C}$, with $i \neq j$, and a positive clearance $d(C_i, \partial\mathcal{F}) > 0$ from the boundary of the freespace $\mathcal{F}$.*

Then, similarly to [165] and Chapter 6, we describe each polygonal obstacle $P_i \in \mathcal{P} \subseteq \mathcal{O}$ by an *obstacle function*, a real-valued map providing an implicit representation of the form

$$P_i = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \beta_i(\mathbf{x}) \leq 0\} \tag{7.6}$$

that the robot can construct online from the catalogued geometry after it has localized P_i, as detailed in Appendix A.1. We also require the following technical assumption.

Assumption 7.3. *For each $P_i \in \mathcal{P}$, there exists $\varepsilon_i > 0$ such that the set $S_{\beta_i} := \{\mathbf{x} \,|\, \beta_i(\mathbf{x}) \leq \varepsilon_i\}$ has a positive clearance $d(S_{\beta_i}, C) > 0$ from any obstacle $C \in \mathcal{C}$.*

Note that Assumptions 7.2 and 7.3 constrain the shape (convex) and placements (sufficiently separated) only of obstacles that have never previously been encountered. Familiar (polygonal, dilated by r) obstacles $P_i \in \mathcal{P}$, while fixed, can be placed completely arbitrarily

with no further prior information: in particular, they can overlap unrestrictedly, with no jeopardy to our formal results, because we rely on the sensor oracle to recognize and locate them in real time. Obstacles in $\mathcal{P}$ are similarly allowed to overlap with the boundary of the enclosing freespace $\partial\mathcal{F}_e$. To control the scope of the present Section, we simply assume that a path to the goal always exists, i.e., the robot operates in a non-adversarial environment.

Assumption 7.4. *The freespace $\mathcal{F}$ is path-connected.*

Finally, in Section 7.4.2, we impose the technical Assumption 7.5 precluding the possibility that any of the (topologically unavoidable) unstable saddle points of our control law coincide with a catalogued "knot point" of any familiar obstacle (a condition that we conjecture should be generic in the configuration space of obstacle placements).

Based on these assumptions and further positing first-order, fully-actuated robot dynamics $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x})$, the problem consists of finding a Lipschitz continuous controller $\mathbf{u} : \mathcal{F} \to \mathbb{R}^2$, that leaves the freespace $\mathcal{F}$ positively invariant and asymptotically steers almost all configurations in $\mathcal{F}$ to the given goal $\mathbf{x}_d \in \mathcal{F}$. We have also summarized key symbols used throughout this Section in Table 7.1.

7.2 Navigational Representation of the Environment

In this Section, we introduce associated notation for the four distinct representations of the environment that we will refer to as *planning spaces* and use in the construction of our algorithm. Fig. 7.1 illustrates the role of these spaces and the transformations that relate them in constructing and analyzing a realtime generated vector field that guarantees safe passage to the goal.

7.2.1 Physical Space

The *physical space* is a complete description of the geometry of the unknown actual world and while inaccessible to the robot is used for purposes of analysis. It describes the enclosing workspace $\mathcal{W}_e$, punctured with the obstacles $\tilde{\mathcal{O}}$. This gives rise to the freespace $\mathcal{F}$, given in (7.2), consisting of all placements of the robot's centroid that entail no intersections of its body with any interior obstacles or intrusions from the boundary. The robot navigates

$\tilde{\mathcal{P}}_{\mathcal{I}} := \{\tilde{P}_i\}_{i\in\mathcal{I}} \subseteq \tilde{\mathcal{P}}$	Set of instantiated familiar polygonal obstacles
$\mathcal{I} \subseteq \mathcal{N}_{\mathcal{P}}$	Index set of the $\lvert\mathcal{I}\rvert$ instantiated obstacles in $\tilde{\mathcal{P}}_{\mathcal{I}}$
$\mathcal{F}^{\mathcal{I}}_{sem}$	Semantic space corresponding to $\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}$
$\mathcal{F}^{\mathcal{I}}_{map}$	Mapped space corresponding to $\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}$
$\mathcal{F}^{\mathcal{I}}_{model}$	Model space corresponding to $\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}$
$\mathcal{P}^{\mathcal{I}}_{sem} := \bigsqcup_{i\in\mathcal{I}} P_i$	Set of familiar, polygonal obstacles instantiated in the semantic space
$\mathcal{C}_{sem} := \{C_i\}_{i\in\mathcal{J}_{\mathcal{C}}} \subseteq \mathcal{C}$	Set of unknown obstacles in the semantic space, indexed by $\mathcal{J}_{\mathcal{C}} \subseteq \mathcal{N}_{\mathcal{C}}$
$\mathcal{P}^{\mathcal{I}}_{map} := \bigcup_{i\in\mathcal{I}} P_i = \{P_i\}_{i\in\mathcal{J}^{\mathcal{I}}}$	Set of consolidated familiar obstacles in the mapped space, indexed by $\mathcal{J}^{\mathcal{I}}$
$\mathcal{C}_{map} := \mathcal{C}_{sem}$	Set of unknown obstacles in the mapped space, indexed by $\mathcal{J}_{\mathcal{C}} \subseteq \mathcal{N}_{\mathcal{C}}$
$\mathcal{B}^{\mathcal{I}}_{map} := \{B_i\}_{i\in\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}}$	Connected components of $\mathcal{P}^{\mathcal{I}}_{map}$ to be merged into $\partial\mathcal{F}_e$, indexed by $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$
$\mathcal{D}^{\mathcal{I}}_{map} := \{D_i\}_{i\in\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}}$	Connected components of $\mathcal{P}^{\mathcal{I}}_{map}$ to be deformed into disks, indexed by $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$
$\mathbf{x}^*_i \in \mathbb{R}^2, i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$	Centers of the $\lvert\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}\rvert$ disks in the model space
$\rho_i \in \mathbb{R}, i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$	Radii of the $\lvert\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}\rvert$ disks in the model space

Table 7.2: Key symbols related to the environment representation in Section 7.2.

this space toward the goal, discovering and localizing new obstacles along the way. Those discovered obstacles which are not convex are (by assumption) "familiar" and are then "instantiated" — recalled, and registered from memory to populate the accumulating record of discovery in the semantic space — as we next discuss. Similarly to Chapter 6, those which are "unfamiliar" are presumed convex and registered as such in the companion spaces next to be presented[2].

We denote by $\tilde{\mathcal{P}}_{\mathcal{I}} := \{\tilde{P}_i\}_{i\in\mathcal{I}} \subseteq \tilde{\mathcal{P}}$ the finite set of (constantly updated) physically "instantiated" familiar objects, indexed by $\mathcal{I} \subseteq \mathcal{N}_{\mathcal{P}}$, that drives the construction of the semantic, mapped and model spaces described next. As explained in Section 7.4.1, such elements $\mathcal{I}$ of the power set $2^{\mathcal{N}_{\mathcal{P}}}$ also index the modes of our hybrid system.

[2] Although we make no use in the present work of the discovered unfamiliar objects beyond simply avoiding them, future work could relax the convexity requirement to build up in memory an increasingly complete geometric description (treated in the same manner as in the "familiar" case) from whatever subsequent encounters ensue along the way to the goal.

7.2.2 Semantic Space

The *semantic space* $\mathcal{F}_{sem}^{\mathcal{I}}$ records the robot's evolving information about the environment aggregated from the raw sensor data about the observable portions of a subset of unrecognized (and therefore, presumed convex) obstacles from $\mathcal{C}$, together with the polygonal boundaries of the $|\mathcal{I}|$ familiar obstacles, that are instantiated at the moment the sensory data triggers the identification and localization a familiar obstacle.

Definition 7.1. *A familiar obstacle $\tilde{P} \in \tilde{\mathcal{P}}$ is considered to be "instantiated", if it has been sensed, recognized, localized, and its dilation $P \in \mathcal{P}$ is permanently included in the semantic space. This means that there exists a time $t_P > 0$, such that $\overline{\mathsf{B}(\mathbf{x}^{t_P}, R)} \cap \tilde{P} \neq \varnothing$ and $\overline{\mathsf{B}(\mathbf{x}^{t}, R)} \cap \tilde{P} = \varnothing$, for all $t < t_P$.*

We denote the *set of unrecognized obstacles in the semantic space* by $\mathcal{C}_{sem} := \{C_i\}_{i \in \mathcal{J}_{\mathcal{C}}}$, indexed by $\mathcal{J}_{\mathcal{C}} \subseteq \mathcal{N}_{\mathcal{C}}$, and the *set of familiar obstacles in the semantic space* by $\mathcal{P}_{sem}^{\mathcal{I}} := \bigsqcup_{i \in \mathcal{I}} P_i$.

It is important to note that this environment is constantly updated, both by discovering and storing new familiar obstacles in the semantic map and by discarding old information and storing new information regarding obstacles in $\mathcal{C}$.[2] Here, the robot is treated as a point particle, since all obstacles are dilated by r in the passage from the workspace to the freespace representation of valid placements.

7.2.3 Mapped Space

Although the semantic space contains all the relevant geometric information (identity and pose) about the obstacles the robot has encountered, it does not explicitly contain any topological information about the explored environment, as represented by the disjoint union operation in the definition of $\mathcal{P}_{sem}^{\mathcal{I}}$. This is because Assumption 7.3 does not exclude overlaps between obstacles in $\mathcal{P}$. Their algorithmically effective consolidation in real time reduces the number while increasing the geometric complexity of the actual freespace obstacles the robot must negotiate along the way to the goal. To do so, we need therefore to take unions of overlapping obstacles in $\mathcal{P}_{sem}^{\mathcal{I}}$, making up $\mathcal{P}_{map}^{\mathcal{I}} := \bigcup_{i \in \mathcal{I}} P_i = \{P_i\}_{i \in \mathcal{J}^{\mathcal{I}}}$ (i.e., a new set

of *consolidated familiar obstacles* indexed by $\mathcal{J}^{\mathcal{I}}$ with $|\mathcal{J}^{\mathcal{I}}| \leq |\mathcal{I}|$), as well as copies of the sensed fragments of unknown obstacles from $\mathcal{C}_{sem}$ (i.e., $\mathcal{C}_{map} := \mathcal{C}_{sem}$) to form the *mapped space* $\mathcal{F}^{\mathcal{I}}_{map}$ as

$$\mathcal{F}^{\mathcal{I}}_{map} := \mathcal{F}_e \backslash (\mathcal{P}^{\mathcal{I}}_{map} \cup \mathcal{C}_{map}) \tag{7.7}$$

Note that, by Assumption 7.3, the convex obstacles are assumed to be far enough away from the familiar obstacles, such that no overlap occurs in the above union.

Next, we focus on the connected components of $\mathcal{P}^{\mathcal{I}}_{map}$; since Assumption 7.3 allows overlaps between obstacles in $\mathcal{P}$ and the boundary of the enclosing freespace $\partial\mathcal{F}_e$, for any connected component P of $\mathcal{P}^{\mathcal{I}}_{map}$ such that $P \cap \partial\mathcal{F}_e \neq \varnothing$, we take $B := P \cap \mathcal{F}_e$ and include B in a new set $\mathcal{B}^{\mathcal{I}}_{map}$, indexed by $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$. The rest of the connected components in $\mathcal{P}^{\mathcal{I}}_{map}$, which do not intersect $\partial\mathcal{F}_e$, are included in a separate set $\mathcal{D}^{\mathcal{I}}_{map}$, indexed by $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$. The idea here is that obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ should be merged to the boundary of the enclosing freespace $\partial\mathcal{F}_e$, and obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ should be deformed to disks, since $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ need to be diffeomorphic.

7.2.4 Model Space

The *model space* $\mathcal{F}^{\mathcal{I}}_{model}$ has the same boundary as $\mathcal{F}_e$ (i.e. $\partial\mathcal{F}^{\mathcal{I}}_{model} := \partial\mathcal{F}_e$) and consists of copies of the sensed fragments of the $|\mathcal{J}_{\mathcal{C}}|$ unrecognized visible convex obstacles in $\mathcal{C}_{map}$, and a collection of $|\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}|$ Euclidean disks corresponding to the $|\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}|$ consolidated obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ that are deformed to disks. The centers $\{\mathbf{x}^*_i\}_{i\in\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}}$ and radii $\{\rho_i\}_{i\in\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}}$ of the $|\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}|$ disks are chosen so that $\overline{\mathsf{B}(\mathbf{x}^*_i, \rho_i)}$ is contained in the interior of $D_i \in \mathcal{D}^{\mathcal{I}}_{map}$, as required in [164]. The obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ are merged into $\partial\mathcal{F}_e$, to make $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ topologically equivalent, through a map $\mathbf{h}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$. We can, therefore, write $\mathcal{F}^{\mathcal{I}}_{model}$ as

$$\mathcal{F}^{\mathcal{I}}_{model} = \mathcal{F}_e \Bigg\backslash \left(\bigcup_{i\in\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}} \overline{\mathsf{B}(\mathbf{x}^*_i, \rho_i)} \cup \mathcal{C}_{map} \right) \tag{7.8}$$

7.2.5 Implicit Representation of Obstacles

We note here that the construction of the map $\mathbf{h}^{\mathcal{I}}$ between the mapped space $\mathcal{F}^{\mathcal{I}}_{map}$ and the model space $\mathcal{F}^{\mathcal{I}}_{model}$, described next in Section 7.3 and shown in Fig. 7.1, relies on the existence of smooth implicit functions $\beta : \mathbb{R}^2 \to \mathbb{R}$ for polygons, constructed such that $\beta(\mathbf{x}) = 0$ implies that $\mathbf{x}$ lies on the boundary of the polygon. Although the construction of such functions is a separate problem on its own, here we derive implicit representations using so-called "R-functions" from the constructive solid geometry literature [176]. In Appendix A.1, we describe the method used for the construction of such implicit functions for polygonal obstacles that have the desired property of being analytic everywhere except for the polygon vertices.

7.3 The Diffeomorphism Construction Between the Mapped and Model Spaces

In this Section, we describe our method of constructing the diffeomorphism, $\mathbf{h}^{\mathcal{I}}$, between the mapped space $\mathcal{F}^{\mathcal{I}}_{map}$ and the model space $\mathcal{F}^{\mathcal{I}}_{model}$. We assume that the robot has already recognized, localized and stored the $|\mathcal{J}^{\mathcal{I}}|$ consolidated familiar polygonal obstacles in $\mathcal{P}^{\mathcal{I}}_{map}$ in its map, and has subsequently identified obstacles to be merged to the boundary of the enclosing freespace $\partial\mathcal{F}_e$, stored in $\mathcal{B}^{\mathcal{I}}_{map}$, and obstacles to be deformed to disks, stored in $\mathcal{D}^{\mathcal{I}}_{map}$.

The idea is then to compose a sequence of "purging" diffeomorphisms, which coincide with the identity map except on a small "collar" around each component of $\mathcal{P}^{\mathcal{I}}_{map}$, that produce successively less complicated isolated shapes. The final simplified shapes are then conveniently deformed into a disk if the corresponding obstacle belongs in $\mathcal{D}^{\mathcal{I}}_{map}$, or into the boundary of $\mathcal{F}_e$ if the corresponding obstacle belongs in $\mathcal{B}^{\mathcal{I}}_{map}$, in order to generate the model space $\mathcal{F}^{\mathcal{I}}_{model}$. Before describing these transformations, we first provide some background on the used obstacle representation methodology in Section 7.3.1, and provide associated notation in Table 7.3.

$\mathcal{T}_{P_i}$	Tree of triangles, constructed from the dual graph of the triangulation of P_i
$\mathcal{V}_{P_i}$	Set of vertices of $\mathcal{T}_{P_i}$ identified with triangles in the triangulation of P_i
$\mathcal{E}_{P_i}$	Set of edges of $\mathcal{T}_{P_i}$ encoding triangle adjacency in the triangulation of P_i
$\mathcal{F}^{\mathcal{I}}_{map,j_i} \subset \mathbb{R}^2$	Mapped space before the purging of leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\mathcal{F}^{\mathcal{I}}_{map,p(j_i)} \subset \mathbb{R}^2$	Mapped space after the purging of leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\hat{\mathcal{F}}^{\mathcal{I}}_{map} \subset \mathbb{R}^2$	Mapped space after the purging of all leaf triangles
$\mathbf{x}^*_{j_i} \in \mathbb{R}^2$	Admissible center for the purging transformation of a leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\mathbf{x}^*_i \in \mathbb{R}^2$	Admissible center for the transformation of a root triangle $r_i \in \mathcal{V}_{P_i}$
$\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$	Vertices of a leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\mathbf{x}_{1r_i}, \mathbf{x}_{2r_i}, \mathbf{x}_{3r_i}$	Vertices of a root triangle $r_i \in \mathcal{V}_{P_i}$
$\mathcal{Q}_{j_i} \subset \mathbb{R}^2$	Quadrilateral $\mathbf{x}_{3j_i}\mathbf{x}_{1j_i}\mathbf{x}^*_{j_i}\mathbf{x}_{2j_i}\mathbf{x}_{3j_i}$ associated with a leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\mathcal{Q}_{r_i} \subset \mathbb{R}^2$	Quadrilateral $\mathbf{x}_{3r_i}\mathbf{x}_{1r_i}\mathbf{x}^*_i\mathbf{x}_{2r_i}\mathbf{x}_{3r_i}$ associated with a root triangle $r_i \in \mathcal{V}_{P_i}$
$\overline{\mathcal{Q}}_{j_i} \subset \mathbb{R}^2$	Admissible polygonal collar associated with a leaf triangle $j_i \in \mathcal{V}_{P_i}$
$\overline{\mathcal{Q}}_{r_i} \subset \mathbb{R}^2$	Admissible polygonal collar associated with a root triangle $r_i \in \mathcal{V}_{P_i}$
$\gamma_{j_i}, \gamma_{r_i} : \mathbb{R}^2 \to \mathbb{R}$	Implicit function associated with $\mathcal{Q}_{j_i}$ or $\mathcal{Q}_{r_i}$
$\delta_{j_i}, \delta_{r_i} : \mathbb{R}^2 \to \mathbb{R}$	Implicit function associated with $\overline{\mathcal{Q}}_{j_i}$ or $\overline{\mathcal{Q}}_{r_i}$
$\sigma_{\gamma_{j_i}}, \sigma_{\gamma_{r_i}} : \mathbb{R}^2 \to [0,1]$	Auxiliary C^∞ switch associated with $\mathcal{Q}_{j_i}$ or $\mathcal{Q}_{r_i}$
$\sigma_{\delta_{j_i}}, \sigma_{\delta_{r_i}} : \mathbb{R}^2 \to [0,1]$	Auxiliary C^∞ switch associated with $\overline{\mathcal{Q}}_{j_i}$ or $\overline{\mathcal{Q}}_{r_i}$
$\sigma_{j_i}, \sigma_{r_i} : \mathbb{R}^2 \to [0,1]$	C^∞ switch of the transformation of a leaf triangle j_i or a root triangle r_i
$\nu_{j_i}, \nu_{r_i} : \mathbb{R}^2 \to [0,1]$	Deforming factor for a leaf triangle j_i or a root triangle r_i
$\mathbf{h}^{\mathcal{I}}_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$	Purging transformation mapping a leaf triangle j_i to its parent $p(j_i)$
$\mathbf{g}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \hat{\mathcal{F}}^{\mathcal{I}}_{map}$	Composition of purging transformations mapping $\mathcal{F}^{\mathcal{I}}_{map}$ to $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$
$\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$	Diffeomorphism between $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$
$\mathbf{h}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$	Diffeomorphism between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$

Table 7.3: Key symbols related to the diffeomorphism construction from $\mathcal{F}^{\mathcal{I}}_{map}$ to $\mathcal{F}^{\mathcal{I}}_{model}$, described in Section 7.3.

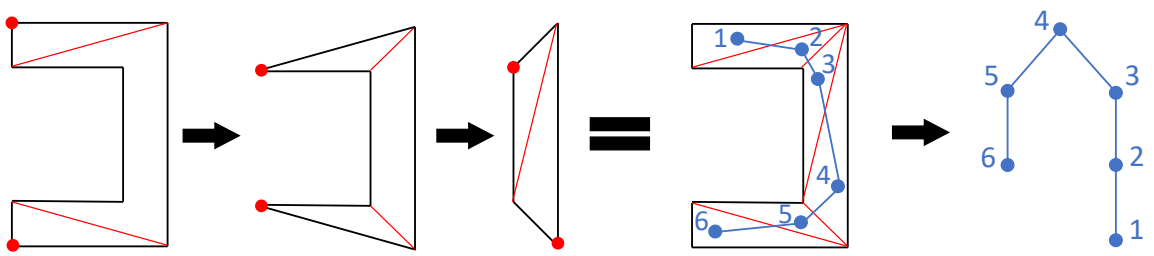

Figure 7.3: Triangulation of a non-convex obstacle using the Ear Clipping Method. The original polygon is guaranteed to have at least two ears (red dots) by the Two Ears Theorem, which induce triangles that can be removed from the polygon. By repeating this process, we get the final triangulation and its dual graph, which is guaranteed to be a tree. This tree can be restructured by setting the root to be the triangle of maximal surface area, to yield the order of purging transformations in descending depth; in this particular example this order is $1 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 4$.

7.3.1 Obstacle Representation

In order to construct the map $\mathbf{h}^{\mathcal{I}}$ between the mapped space and the model space, we assume that the robot has access to the triangulation of each one of the obstacles stored in both $\mathcal{D}^{\mathcal{I}}_{map}$ and $\mathcal{B}^{\mathcal{I}}_{map}$. This triangulation can be efficiently constructed online upon recognition of each obstacle, using the *Two Ears Theorem* [133], and the associated *Ear Clipping Method* [55][3].

Briefly, an *ear* of a simple polygon is a vertex of the polygon such that the line segment between the two neighbors of the vertex lies entirely in the interior of the polygon. The Two Ears Theorem guarantees that every simple polygon has at least two such ears, and the Ear Clipping Method uses this result to efficiently construct polygon triangulations in $\mathcal{O}(n^2)$ time. Namely, an ear and its two neighbors form a triangle that is not crossed by any other part of the polygon and can be, therefore, safely removed. Removing a triangle of this type produces a polygon with one less vertex than the original polygon; we can repeat the process to eventually get a single triangle and complete the triangulation. An example is shown in Fig. 7.3.

Except for its utility in constructing triangulations, the Two Ears Theorem guarantees that the dual graph of the triangulation of a simple polygon with no holes constructed with the Ear Clipping Method (i.e., a graph with one vertex per triangle and one edge per pair

[3]Special thanks to Prof. Elon Rimon for pointing out these results.

of adjacent triangles) is in fact a tree [142].

Therefore, in order to construct a tree of triangles $\mathcal{T}_{P_i} := (\mathcal{V}_{P_i}, \mathcal{E}_{P_i})$ corresponding to a polygon P_i, with $\mathcal{V}_{P_i}$ a set of vertices identified with triangles (i.e., vertices of the dual of the formal triangulation) and $\mathcal{E}_{P_i}$ a set of edges encoding triangle adjacency, we can triangulate P_i using the Ear Clipping Method, pick any triangle as root, and construct $\mathcal{T}_{P_i}$ based on the adjacency properties induced by the dual graph of the triangulation, as shown in Fig. 7.3. If $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, we pick as root the triangle with the largest surface area, whereas if $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, we pick as root a triangle adjacent to $\partial\mathcal{F}_e$. This will give us a *tree-of-triangles* for P_i in a notion similar to [165]. Our goal is then to successively "purge" this tree, triangle by triangle, in order of descending depth, until we reach the root triangle. Then, we can use a diffeomorphism similar to that presented in Chapter 6 to map the exterior and boundary of the root triangle onto the exterior and boundary of a topologically equivalent disk if $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, or merge the root triangle into $\partial\mathcal{F}_e$ if $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$. These operations are all performed online; we provide some computational performance metrics with our experiments in Section 7.8.

We describe the algorithm for each purging transformation of the leaf nodes in Section 7.3.2 and the (final) root triangle purging transformation in Section 7.3.3. Finally, Section 7.3.4 defines the diffeomorphism between the mapped and model spaces, along with associated qualitative properties.

7.3.2 Intermediate Spaces Related by Leaf Purging Transformations

In this Section, we describe the purging transformation that maps the boundary of a leaf triangle $j_i \in \mathcal{V}_{P_i}$ onto the boundary of its parent $p(j_i) \in \mathcal{V}_{P_i}$, as shown in Fig. 7.4-(1a), (2a). This gives rise to a composition of transformations between a succession of *intermediate spaces*, each including the triangle j_i, and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$, where j_i has been mapped onto the boundary of its parent. Each of these transformations is in principle similar and performing a role analogous to the corresponding purging transformation in [165], with two important differences. First, it deforms space only "locally" around the triangle, without taking into consideration other triangles or polygons, allowing the use of fewer tunable parameters and

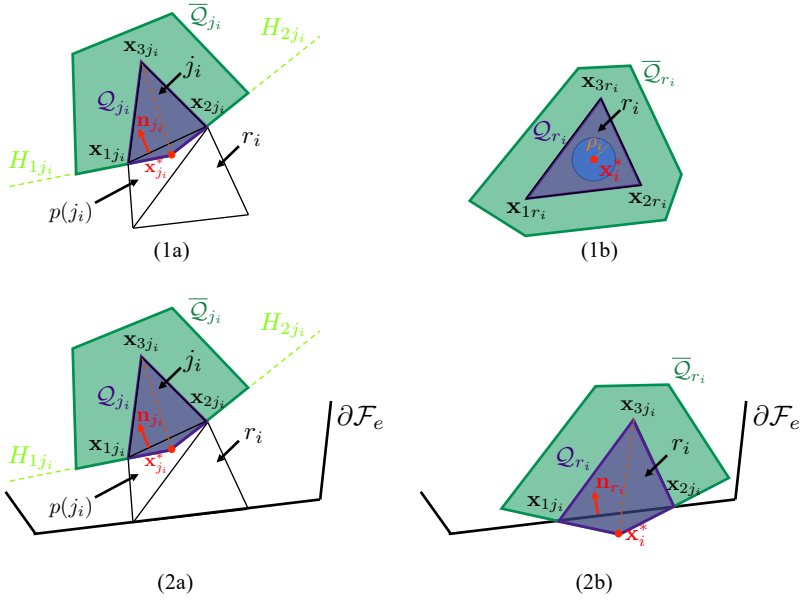

Figure 7.4: Illustration of features used in the transformation of - Top: (1a) a leaf triangle j_i onto its parent $p(j_i)$, and (1b) a root triangle r_i onto a disk centered at $\mathbf{x}_i^*$ with radius ρ_i for an obstacle in $\mathcal{D}_{map}^{\mathcal{I}}$, Bottom: (2a) a leaf triangle j_i onto its parent $p(j_i)$, and (2b) a root triangle r_i onto $\partial\mathcal{F}_e$ for an obstacle in $\mathcal{B}_{map}^{\mathcal{I}}$.

affording better numerical stability since only one triangle is considered at a time. Second, the method presented in [165] is limited to parent-child pairs that strongly overlap instead of just being adjacent, which makes the method impractical for the arbitrary polygonal shapes and meshes involved in this work.

Center of the Transformation and Surrounding Polygonal Collars

Let the vertices of the triangle $j_i \in \mathcal{V}_{P_i}$ be $\mathbf{x}_{1j_i}$, $\mathbf{x}_{2j_i}$ and $\mathbf{x}_{3j_i}$ in counterclockwise order, with $\mathbf{x}_{1j_i}\mathbf{x}_{2j_i}$ the common edge between j_i and $p(j_i)$.

Definition 7.2. *An admissible center for the purging transformation of the leaf triangle $j_i \in \mathcal{V}_{P_i}$, denoted by $\mathbf{x}_{j_i}^*$, is a point in $p(j_i)$ such that the polygon $\mathcal{Q}_{j_i}$ with vertices the original vertices of j_i and $\mathbf{x}_{j_i}^*$ is convex.*

Such a point is always possible to be found, since the two triangles share a common edge; see e.g., Fig. 7.4-(1a),(2a), where we use the median from $\mathbf{x}_{3j_i}$ to find $\mathbf{x}_{j_i}^*$ in $p(j_i)$.

Definition 7.3. *An admissible polygonal collar for the purging transformation of the leaf triangle j_i is a convex polygon $\overline{\mathcal{Q}}_{j_i}$ such that:*

1. *$\overline{\mathcal{Q}}_{j_i}$ does not intersect the interior of any triangle $k \in \mathcal{V}_P$ with $k \neq j_i, p(j_i)$, for all polygons P involved in the construction of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$, or any $C \in \mathcal{C}_{map}$.*

2. *$\mathcal{Q}_{j_i} \subset \overline{\mathcal{Q}}_{j_i}$, and $\overline{\mathcal{Q}}_{j_i} \backslash \mathcal{Q}_{j_i} \subset \mathcal{F}^{\mathcal{I}}_{map,j_i}$.*

Examples of such polygons are shown in Fig. 7.4-(1a),(2a). This polygon is responsible for limiting the effect of the purging transformation in its interior, while keeping its value equal to the identity everywhere else. Intuitively, the requirements in Definition 7.3 will limit the effect of the purging transformation in a region that encloses the triangle j_i and is away from the boundary of any other obstacle. Note that Definition 7.3 forces the edges $\mathbf{x}_{1j_i}\mathbf{x}^*_{j_i}$ and $\mathbf{x}^*_{j_i}\mathbf{x}_{2j_i}$ to be edges of $\overline{\mathcal{Q}}_{j_i}$; the importance of this requirement will become evident in the construction of the provable properties of the diffeomorphism, summarized below in Proposition 7.1. We provide more details about the construction of admissible collars in Appendix A.2.

For the following, we also construct implicit functions $\gamma_{j_i}(\mathbf{x})$ and $\delta_{j_i}(\mathbf{x})$ corresponding to the leaf triangle $j_i \in \mathcal{V}_{P_i}$, as described in Appendix A.1, such that

$$\mathcal{Q}_{j_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \gamma_{j_i}(\mathbf{x}) \leq 0\} \tag{7.9}$$

$$\overline{\mathcal{Q}}_{j_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \delta_{j_i}(\mathbf{x}) \geq 0\} \tag{7.10}$$

Description of the C^∞ switches

As in Chapter 6, we depart from the construction of analytic switches [164] and rely instead on the C^∞ function $\zeta_\mu : \mathbb{R} \to \mathbb{R}$ [78] described by

$$\zeta_\mu(\chi) = \begin{cases} e^{-\mu/\chi}, & \chi > 0 \\ 0, & \chi \leq 0 \end{cases} \tag{7.11}$$

and parametrized by $\mu > 0$, that has derivative

$$\zeta'_\mu(\chi) = \begin{cases} \frac{\mu\, \zeta_\mu(\chi)}{\chi^2}, & \chi > 0 \\ 0, & \chi \leq 0 \end{cases} \tag{7.12}$$

Based on that function, we can then define the auxiliary C^∞ switches

$$\sigma_{\gamma_{j_i}}(\mathbf{x}) := \eta_{\mu_{\gamma_{j_i}},\epsilon_{j_i}} \circ \gamma_{j_i}(\mathbf{x}) \tag{7.13}$$

$$\sigma_{\delta_{j_i}}(\mathbf{x}) := \zeta_{\mu_{\delta_{j_i}}} \circ \frac{\delta_{j_i}(\mathbf{x})}{||\mathbf{x} - \mathbf{x}^*_{j_i}||} \tag{7.14}$$

with $\eta_{\mu,\epsilon}(\chi) := \zeta_\mu(\epsilon - \chi)/\zeta_\mu(\epsilon)$, and $\mu_{\gamma_{j_i}}, \mu_{\delta_{j_i}}, \epsilon_{j_i} > 0$ tunable parameters. Notice that $\sigma_{\gamma_{j_i}}$ is exactly equal to 1 on the boundary of $\mathcal{Q}_{j_i}$ and equal to 0 when $\gamma_{j_i}(\mathbf{x}) \geq \epsilon_{j_i}$, whereas $\sigma_{\delta_{j_i}}$ is 0 outside $\overline{\mathcal{Q}}_{j_i}$. The parameters $\mu_{\gamma_{j_i}}$ and $\mu_{\delta_{j_i}}$ are used to tune the "slope" of $\sigma_{\gamma_{j_i}}$ on the boundary of $\mathcal{Q}_{j_i}$ and how fast $\sigma_{\delta_{j_i}}$ approaches 1 in the interior of $\overline{\mathcal{Q}}_{j_i}$ respectively.

Based on the above, we define the C^∞ *switch of the purging transformation for the leaf triangle* $j_i \in \mathcal{V}_{P_i}$ as a function $\sigma_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathbb{R}$, defined by

$$\sigma_{j_i}(\mathbf{x}) := \begin{cases} \frac{\sigma_{\gamma_{j_i}}(\mathbf{x})\sigma_{\delta_{j_i}}(\mathbf{x})}{\sigma_{\gamma_{j_i}}(\mathbf{x})\sigma_{\delta_{j_i}}(\mathbf{x}) + \left(1 - \sigma_{\gamma_{j_i}}(\mathbf{x})\right)}, & \mathbf{x} \neq \mathbf{x}_{1j_i}, \mathbf{x}_{2j_i} \\ 1, & \mathbf{x} = \mathbf{x}_{1j_i}, \mathbf{x}_{2j_i} \end{cases} \tag{7.15}$$

In this way, we see that $\sigma_{j_i}(\mathbf{x}) = 0$ when $\sigma_{\gamma_{j_i}}(\mathbf{x}) = 0$ or $\sigma_{\delta_{j_i}}(\mathbf{x}) = 0$ (i.e., when $\gamma_{j_i}(\mathbf{x}) \geq \epsilon_{j_i}$ or outside $\overline{\mathcal{Q}}_{j_i}$), $\sigma_{j_i}(\mathbf{x}) = 1$ when $\sigma_{\gamma_{j_i}}(\mathbf{x}) = 1$ (i.e., on the boundary of $\mathcal{Q}_{j_i}$) and σ_{j_i} varies between 0 and 1 everywhere, since $\sigma_{\gamma_{j_i}}$ and $\sigma_{\delta_{j_i}}$ also vary between 0 and 1. Based on Definitions 7.2 and 7.3, it is straightforward to show the following lemma.

Lemma 7.1. *The function* $\sigma_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathbb{R}$ *is smooth away from the triangle vertices* $\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$*, none of which lies in the interior of* $\mathcal{F}^{\mathcal{I}}_{map,j_i}$*.*

Proof. Included in Appendix C.5.1. □

Description of the Deforming Factors

The *deforming factors* are the functions $\nu_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathbb{R}$, responsible for mapping the boundary of the leaf triangle $j_i \in \mathcal{V}_{P_i}$ onto the boundary of its parent $p(j_i)$. Based on Definitions 7.2 and 7.3 and as shown in Fig. 7.4-(1a),(2a), this implies that the functions ν_{j_i} are responsible for mapping the polygonal chain $\mathbf{x}_{2j_i}\mathbf{x}_{3j_i}\mathbf{x}_{1j_i}$ onto the shared edge $\mathbf{x}_{2j_i}\mathbf{x}_{1j_i}$

between j_i and $p(j_i)$. For this reason, we construct ν_{j_i} as follows

$$\nu_{j_i}(\mathbf{x}) := \frac{\left(\mathbf{x}_{1j_i} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}}{\left(\mathbf{x} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}} \tag{7.16}$$

with

$$\mathbf{n}_{j_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}_{2j_i} - \mathbf{x}_{1j_i}}{||\mathbf{x}_{2j_i} - \mathbf{x}_{1j_i}||}, \quad \mathbf{R}_{\frac{\pi}{2}} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{7.17}$$

the normal vector corresponding to the shared edge between j_i and $p(j_i)$.

The Map Between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$

Based on the above, we then construct the map between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ with the j_i-th leaf triangle of P_i purged, as

$$\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) := \sigma_{j_i}(\mathbf{x}) \left(\mathbf{x}^*_{j_i} + \nu_{j_i}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*_{j_i})\right) + (1 - \sigma_{j_i}(\mathbf{x}))\, \mathbf{x} \tag{7.18}$$

Qualitative Properties of the Map Between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$

We first verify that the construction is a smooth change of coordinates between the intermediate mapped spaces.

Lemma 7.2. *The map $\mathbf{h}^{\mathcal{I}}_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ is smooth away from the triangle vertices $\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$, none of which lies in the interior of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$.*

Proof. Included in Appendix C.5.1. □

Proposition 7.1. *The map $\mathbf{h}^{\mathcal{I}}_{j_i}$ is a C^∞ diffeomorphism between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ away from the triangle vertices $\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$, none of which lies in the interior of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$.*

Proof. Included in Appendix C.5.1. □

Composition of Leaf Purging Transformations

The application of the purging transformation described above will result in a tree for P_i with one less vertex and one less edge. Therefore, similarly to [165], we can keep applying

such purging transformations by composition, during execution time, for all leaf triangles of all obstacles P in $\mathcal{B}^{\mathcal{I}}_{map}$ and $\mathcal{D}^{\mathcal{I}}_{map}$ (in any order), until we reach their root triangles. We denote by $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ this final intermediate space, where all obstacles in $\mathcal{F}^{\mathcal{I}}_{map}$ have been deformed to their root triangles $\{r_i\}$, and by $\mathbf{g}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ the map between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$, arising from this composition of purging transformations. Since $\mathbf{g}^{\mathcal{I}}$ is a composition of diffeomorphisms, we immediately get the following result.

Corollary 7.1. *The map $\mathbf{g}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ is a C^∞ diffeomorphism between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ away from sharp corners, none of which lie in the interior of freespace for any of the intermediate or final spaces.*

7.3.3 Purging of Root Triangles

After the successive application of the leaf purging transformations presented in Section 7.3.2, familiar obstacles in $\mathcal{F}^{\mathcal{I}}_{map}$ are reduced to triangles in $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$. These triangles either are homeomorphic to a disk in the interior of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ if they correspond to obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$, or have a common edge with $\partial\mathcal{F}_e$ if they correspond to obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$. Therefore, the final step to generate the model space $\mathcal{F}^{\mathcal{I}}_{model}$ is to transform each of the root triangles corresponding to obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ to disks, following a procedure similar to Chapter 6, and merge the root triangles corresponding to obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ to $\partial\mathcal{F}_e$.

Center of the Transformation and Surrounding Polygonal Collars for Obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$

Here we assume that $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$. Let the vertices of the root triangle $r_i \in \mathcal{V}_{P_i}$ be $\mathbf{x}_{1r_i}$, $\mathbf{x}_{2r_i}$ and $\mathbf{x}_{3r_i}$ in counterclockwise order, as shown in Fig. 7.4-(1b).

Definition 7.4. *An admissible center for the transformation of the root triangle r_i, corresponding to a polygon $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, is a point $\mathbf{x}^*_i$ in the interior of r_i.*

Without loss of generality, we pick $\mathbf{x}^*_i$ to be the barycenter of r_i, and the *radius of the transformation* to be a number $\rho_i < d(\mathbf{x}^*_i, \partial r_i)$, following the admissibility assumptions made in [165]. We also set $\mathcal{Q}_{r_i}$ to be the closure of r_i itself, and define a polygonal collar $\overline{\mathcal{Q}}_{r_i}$ for the root triangle transformation as follows.

Definition 7.5. *An admissible polygonal collar for the transformation of the root triangle r_i, corresponding to a polygon $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, is a convex polygon $\overline{\mathcal{Q}}_{r_i}$ such that:*

1. *$\overline{\mathcal{Q}}_{r_i}$ does not intersect the interior of any other triangle r_j, or any $C \in \mathcal{C}_{map}$.*
2. *$\mathcal{Q}_{r_i} \subset \overline{\mathcal{Q}}_{r_i}$, and $\overline{\mathcal{Q}}_{r_i} \backslash \mathcal{Q}_{r_i} \subset \hat{\mathcal{F}}^{\mathcal{I}}_{map}$.*

An example of such a polygon is shown in Fig. 7.4-(1b). Again, this polygon is responsible for limiting the effect of the transformation in its interior, while keeping it equal to the identity map everywhere else. Similarly to Section 7.3.2, we also construct implicit functions $\gamma_{r_i}(\mathbf{x})$ and $\delta_{r_i}(\mathbf{x})$ for each root triangle r_i, such that

$$\mathcal{Q}_{r_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \gamma_{r_i}(\mathbf{x}) \leq 0\} \tag{7.19}$$

$$\overline{\mathcal{Q}}_{r_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \delta_{r_i}(\mathbf{x}) \geq 0\} \tag{7.20}$$

Description of the C^∞ switches for Obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$

Following the notation of Section 7.3.2, we can define the auxiliary C^∞ switches

$$\sigma_{\gamma_{r_i}}(\mathbf{x}) := \eta_{\mu_{\gamma_{r_i}}, \epsilon_{r_i}} \circ \gamma_{r_i}(\mathbf{x}) \tag{7.21}$$

$$\sigma_{\delta_{r_i}}(\mathbf{x}) := \zeta_{\mu_{\delta_{r_i}}} \circ \frac{\delta_{r_i}(\mathbf{x})}{||\mathbf{x} - \mathbf{x}_i^*||} \tag{7.22}$$

with $\eta_{\mu,\epsilon}(\chi) := \zeta_\mu(\epsilon - \chi)/\zeta_\mu(\epsilon)$, ζ defined as in (7.11) and $\mu_{\gamma_{r_i}}, \mu_{\delta_{r_i}}, \epsilon_{r_i} > 0$ tunable parameters.

Based on the above, we then define the C^∞ *switch of the transformation of the root triangle* r_i as the function $\sigma_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{map}$ given by

$$\sigma_{r_i}(\mathbf{x}) := \frac{\sigma_{\gamma_{r_i}}(\mathbf{x})\sigma_{\delta_{r_i}}(\mathbf{x})}{\sigma_{\gamma_{r_i}}(\mathbf{x})\sigma_{\delta_{r_i}}(\mathbf{x}) + \big(1 - \sigma_{\gamma_{r_i}}(\mathbf{x})\big)} \tag{7.23}$$

It can be seen that the function σ_{r_i} will be 0 outside $\overline{\mathcal{Q}}_{r_i}$, exactly equal to 1 on the boundary of $\mathcal{Q}_{r_i}$ (i.e., on the boundary of the root triangle r_i) and varies smoothly between 0 and 1 everywhere else.

We can easily show the following lemma, as the function ζ eliminates all the singular points of δ_{r_i} that correspond to the vertices of $\overline{\mathcal{Q}}_{r_i}$.

Lemma 7.3. *The switch $\sigma_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathbb{R}$ is smooth away from the triangle vertices $\mathbf{x}_{1r_i}$, $\mathbf{x}_{2r_i}$, $\mathbf{x}_{3r_i}$, none of which lies in the interior of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$.*

Description of the Deforming Factors for Obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$

Here, the deforming factors are the functions $\nu_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathbb{R}$, responsible for transforming each root triangle corresponding to an obstacle in $\mathcal{D}^{\mathcal{I}}_{map}$ to a disk in $\mathbb{R}^2$. The deforming factors we use are inspired by those in [165], but do not depend on the values of the implicit functions γ_{r_i}. Namely, the deforming factors are given based on the desired final radii ρ_i as

$$\nu_{r_i}(\mathbf{x}) := \frac{\rho_i}{||\mathbf{x} - \mathbf{x}^*_i||} \tag{7.24}$$

Center of the Transformation and Surrounding Polygonal Collars for Obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$

Next we focus on obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$. The procedure here is slightly different, since we want to merge the root triangle r_i to the boundary of the enclosing freespace $\mathcal{F}_e$. Namely, we assume that the vertices of the triangle r_i are $\mathbf{x}_{1r_i}, \mathbf{x}_{2r_i}, \mathbf{x}_{3r_i}$ in counterclockwise order, with $\mathbf{x}_{1r_i}\mathbf{x}_{2r_i}$ the common edge between r_i and $\partial\mathcal{F}_e$, as shown in Fig. 7.4-(2b)[4]. Then, we pick an admissible center similarly to Definition 7.2, as follows.

Definition 7.6. *An admissible center for the transformation of the root triangle r_i, corresponding to a polygon $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, denoted by $\mathbf{x}^*_i$, is a point in $\mathbb{R}^2 \backslash \mathcal{F}_e$ such that the polygon $\mathcal{Q}_{r_i}$ with vertices the original vertices of r_i and $\mathbf{x}^*_i$ is convex.*

We also define $\mathcal{Q}_{r_i}$ to be the convex quadrilateral with boundary $\mathbf{x}_{3r_i}\mathbf{x}_{1r_i}\mathbf{x}^*_i\mathbf{x}_{2r_i}\mathbf{x}_{3r_i}$. The collars used are defined similarly to Definition 7.3, as the transformation itself is designed to be quite similar with the purging transformation.

[4]If r_i and $\partial\mathcal{F}_e$ share two common edges, we just pick one of them at random.

Definition 7.7. *An admissible polygonal collar for the transformation of the root triangle r_i, corresponding to a polygon $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, is a convex polygon $\overline{\mathcal{Q}}_{r_i}$ such that:*

1. *$\overline{\mathcal{Q}}_{r_i}$ does not intersect the interior of any other triangle r_j, or any $C \in \mathcal{C}_{map}$.*
2. *$\mathcal{Q}_{r_i} \subset \overline{\mathcal{Q}}_{r_i}$, and $\overline{\mathcal{Q}}_{r_i} \backslash \mathcal{Q}_{r_i} \subset \hat{\mathcal{F}}^{\mathcal{I}}_{map}$.*

Description of the C^∞ switches for Obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$

With the definition of $\mathcal{Q}_{r_i}$ and $\overline{\mathcal{Q}}_{r_i}$ as described above, we associate implicit functions $\gamma_{r_i}(\mathbf{x})$ and $\delta_{r_i}(\mathbf{x})$ as in (7.19), (7.20), auxiliary switches $\sigma_{\gamma_{r_i}}$ and $\sigma_{\delta_{r_i}}$ as in (7.21) and (7.22), and overall C^∞ *switch of the transformation of the root triangle* r_i as the function $\sigma_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{map}$ given in (7.23).

Description of the Deforming Factors for Obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$

Finally, in order to merge the root triangle into the boundary $\partial\mathcal{F}_e$, we define the deforming factors similarly to (7.16), as the functions $\nu_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathbb{R}$, given by

$$\nu_{r_i}(\mathbf{x}) := \frac{\left(\mathbf{x}_{1r_i} - \mathbf{x}^*_i\right)^\top \mathbf{n}_{r_i}}{\left(\mathbf{x} - \mathbf{x}^*_i\right)^\top \mathbf{n}_{r_i}} \tag{7.25}$$

with

$$\mathbf{n}_{r_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}_{2r_i} - \mathbf{x}_{1r_i}}{||\mathbf{x}_{2r_i} - \mathbf{x}_{1r_i}||}, \quad \mathbf{R}_{\frac{\pi}{2}} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{7.26}$$

the normal vector corresponding to the shared edge between r_i and $\partial\mathcal{F}_e$.

The Map Between $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$

First of all, we define

$$\sigma_d(\mathbf{x}) := 1 - \sum_{i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{B}} \cup \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}} \sigma_{r_i}(\mathbf{x}) \tag{7.27}$$

Using the above constructions and Definitions 7.4, 7.5, 7.6 and 7.7 we are led to the following results.

Lemma 7.4. *At any point $\mathbf{x} \in \hat{\mathcal{F}}^{\mathcal{I}}_{map}$, at most one of the switches $\{\sigma_{r_i}\}_{i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{B}} \cup \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}}$ can be nonzero.*

Corollary 7.2. *The set* $\{\sigma_{r_i}\}_{i\in\mathcal{J}_{\mathcal{B}}^{\mathcal{I}}\cup\mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \cup \{\sigma_d\}$ *defines a partition of unity over* $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$.

With the construction of σ_{r_i} (as in (7.23)) and ν_{r_i} (as in either (7.24) or (7.25), depending on whether i belongs to $\mathcal{J}_{\mathcal{D}}^{\mathcal{I}}$ or $\mathcal{J}_{\mathcal{B}}^{\mathcal{I}}$ respectively) for each root triangle, we can now construct the map $\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}_{map}^{\mathcal{I}} \to \mathcal{F}_{model}^{\mathcal{I}}$ given by

$$\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) := \sum_{i\in\mathcal{J}_{\mathcal{B}}^{\mathcal{I}}\cup\mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \sigma_{r_i}(\mathbf{x})\left[\mathbf{x}_i^* + \nu_{r_i}(\mathbf{x})(\mathbf{x}-\mathbf{x}_i^*)\right] + \sigma_d(\mathbf{x})\mathbf{x} \tag{7.28}$$

Qualitative Properties of the Map Between $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$ and $\mathcal{F}_{model}^{\mathcal{I}}$

We can again verify that the construction is a smooth change of coordinates between $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$ and $\mathcal{F}_{model}^{\mathcal{I}}$. Using Lemma 7.3 and the fact that the deforming factors ν_{r_i} are smooth in $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$ (because the centers $\mathbf{x}_i^*$ do not belong in $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$) for all i, we get the following result.

Lemma 7.5. *The map* $\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}_{map}^{\mathcal{I}} \to \mathcal{F}_{model}^{\mathcal{I}}$ *is smooth away from any sharp corners, none of which lie in the interior of* $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$.

Proposition 7.2. *The map* $\hat{\mathbf{h}}^{\mathcal{I}}$ *is a* C^∞ *diffeomorphism between* $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$ *and* $\mathcal{F}_{model}^{\mathcal{I}}$ *away from any sharp corners, none of which lie in the interior of* $\hat{\mathcal{F}}_{map}^{\mathcal{I}}$.

Proof. Included in Appendix C.5.1. □

7.3.4 The Map Between the Mapped Space and the Model Space

Based on the construction of $\mathbf{g}^{\mathcal{I}} : \mathcal{F}_{map}^{\mathcal{I}} \to \hat{\mathcal{F}}_{map}^{\mathcal{I}}$ in Section 7.3.2 and $\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}_{map}^{\mathcal{I}} \to \mathcal{F}_{model}^{\mathcal{I}}$ in Section 7.3.3, we can finally write the map between the mapped space and the model space as the function $\mathbf{h}^{\mathcal{I}} : \mathcal{F}_{map}^{\mathcal{I}} \to \mathcal{F}_{model}^{\mathcal{I}}$ given by

$$\mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \hat{\mathbf{h}}^{\mathcal{I}} \circ \mathbf{g}^{\mathcal{I}}(\mathbf{x}) \tag{7.29}$$

It is straightforward to get the following result, since both $\mathbf{g}^{\mathcal{I}}$ and $\hat{\mathbf{h}}^{\mathcal{I}}$ are C^∞ diffeomorphisms away from sharp corners.

Corollary 7.3. *The map* $\mathbf{h}^{\mathcal{I}}$ *is a* C^∞ *diffeomorphism between* $\mathcal{F}_{map}^{\mathcal{I}}$ *and* $\mathcal{F}_{model}^{\mathcal{I}}$ *away from any sharp corners, none of which lie in the interior of* $\mathcal{F}_{map}^{\mathcal{I}}$.

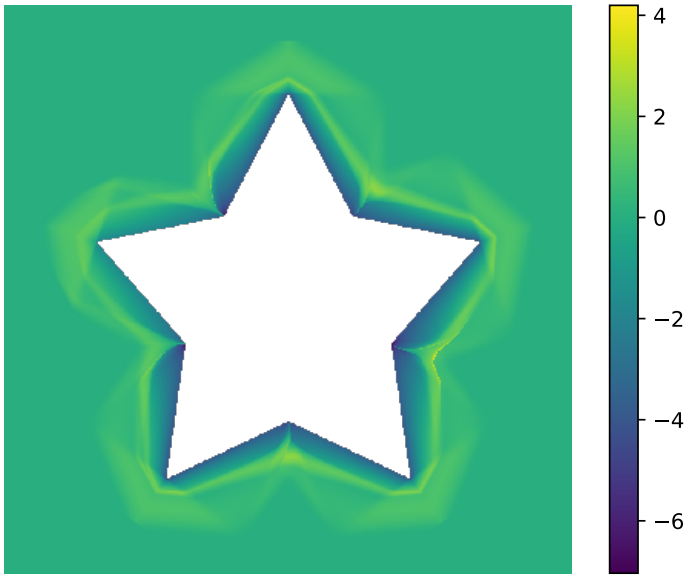

Figure 7.5: Values of $\det(D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}})$ for a single polygon in logarithmic scale, showing the local nature of the diffeomorphism ($\mathbf{h}^{\mathcal{I}}$ becomes equal to the identity transform away from the polygon) and the fact that $\mathbf{h}^{\mathcal{I}}$ is smooth away from sharp corners, that do not lie in the interior of the freespace.

An illustration of the behavior of the map $\mathbf{h}^{\mathcal{I}}$ through a visualization of the values of $\det(D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}})$ for a specific example with a single polygon is included in Fig. 7.5.

7.4 Reactive Controller

The preceding analysis in Section 7.3 describes the diffeomorphism construction between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ for a given index set $\mathcal{I}$ of instantiated familiar obstacles. However, the onboard sensor might discover new obstacles and, subsequently, incorporate them in the semantic map, updating the set $\mathcal{I}$. Therefore, in this Section, we enhance the formal results of Chapter 6 by providing a hybrid systems description of our reactive controller, where each mode is defined by an index set $\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}$ of familiar obstacles stored in the semantic map, the guards describe the sensor trigger events where a previously "unexplored" obstacle is discovered and incorporated in the semantic map (thereby changing $\mathcal{P}^{\mathcal{I}}_{map}$, along with $\mathcal{D}^{\mathcal{I}}_{map}$, $\mathcal{B}^{\mathcal{I}}_{map}$), and the resets describe transitions to new modes that might result in discrete "jumps" of the robot position in the model space. We then need to show that the resulting hybrid controller, for both the fully actuated robot and the differential drive robot, must succeed in the navigation task.

$\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}$	Mode of the hybrid system
$\mathcal{F}^\mathcal{I}, \mathcal{F}^\mathcal{I}_{sem}, \mathcal{F}^\mathcal{I}_{map}, \mathcal{F}^\mathcal{I}_{model} \in \mathbb{R}^2$	Physical, semantic, mapped, model freespace for $\mathcal{I}$
$\mathbf{h}^\mathcal{I} : \mathcal{F}^\mathcal{I}_{map} \to \mathcal{F}^\mathcal{I}_{model}$	Map between $\mathcal{F}^\mathcal{I}_{map}$ and $\mathcal{F}^\mathcal{I}_{model}$ for mode $\mathcal{I}$
$\Gamma \subset 2^{\mathcal{N}_\mathcal{P}} \times 2^{\mathcal{N}_\mathcal{P}}$	Directed graph of discrete mode transitions
$\mathsf{D} := \bigsqcup_{\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}} \mathcal{F}^\mathcal{I}$	Collection of domains for a fully actuated robot
$\mathsf{G} := \bigsqcup_{(\mathcal{I},\mathcal{I}') \in \Gamma} G^{\mathcal{I},\mathcal{I}'}$	Collection of guards for a fully actuated robot
$\mathsf{R} : \mathsf{G} \to \mathsf{D}$	Continuous reset map for a fully actuated robot
$\mathsf{U} : \mathsf{D} \to T\mathsf{D}$	Hybrid vector field for a fully actuated robot
$\mathsf{H} := \left(2^{\mathcal{N}_\mathcal{P}}, \Gamma, \mathsf{D}, \mathsf{U}, \mathsf{G}, \mathsf{R}\right)$	Hybrid system for a fully actuated robot
$\overline{\mathsf{D}} := \bigsqcup_{\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}} (\mathcal{F}^\mathcal{I} \times S^1)$	Collection of domains for a differential drive robot
$\overline{\mathsf{G}} := \bigsqcup_{(\mathcal{I},\mathcal{I}') \in \Gamma} \overline{G}^{\mathcal{I},\mathcal{I}'}$	Collection of guards for a differential drive robot
$\overline{\mathsf{R}} : \overline{\mathsf{G}} \to \overline{\mathsf{D}}$	Continuous reset map for a differential drive robot
$\overline{\mathsf{U}} : \overline{\mathsf{D}} \to T\overline{\mathsf{D}}$	Hybrid vector field for a differential drive robot
$\overline{\mathsf{H}} := \left(2^{\mathcal{N}_\mathcal{P}}, \Gamma, \overline{\mathsf{D}}, \overline{\mathsf{U}}, \overline{\mathsf{G}}, \overline{\mathsf{R}}\right)$	Hybrid system for a differential drive robot
$\mathbf{x} \in \mathcal{F}^\mathcal{I}_{map}$	Fully actuated robot position in $\mathcal{F}^\mathcal{I}_{map}$
$\mathbf{y} := \mathbf{h}^\mathcal{I}(\mathbf{x}) \in \mathcal{F}^\mathcal{I}_{model}$	Fully actuated robot position in $\mathcal{F}^\mathcal{I}_{model}$
$\mathbf{y}_d := \mathbf{h}^\mathcal{I}(\mathbf{x}_d)$	Goal position in $\mathcal{F}^\mathcal{I}_{model}$
$\mathcal{LF}(\mathbf{y}) \subset \mathcal{F}^\mathcal{I}_{model}$	Local freespace at $\mathbf{y} \in \mathcal{F}^\mathcal{I}_{model}$
$\mathbf{v}^\mathcal{I} : \mathcal{F}^\mathcal{I}_{model} \to T\mathcal{F}^\mathcal{I}_{model}$	Vector field controller for a fully actuated robot in $\mathcal{F}^\mathcal{I}_{model}$
$\mathbf{u}^\mathcal{I} : \mathcal{F}^\mathcal{I}_{map} \to T\mathcal{F}^\mathcal{I}_{map}$	Vector field controller for a fully actuated robot in $\mathcal{F}^\mathcal{I}_{map}$
$\overline{\mathbf{h}}^\mathcal{I} : \mathcal{F}^\mathcal{I}_{map} \times S^1 \to \mathcal{F}^\mathcal{I}_{model} \times S^1$	Diffeomorphism between $\mathcal{F}^\mathcal{I}_{map} \times S^1$ and $\mathcal{F}^\mathcal{I}_{model} \times S^1$
$\overline{\mathbf{x}} := (\mathbf{x}, \psi) \in \mathcal{F}^\mathcal{I}_{map} \times S^1$	Differential drive robot state in $\mathcal{F}^\mathcal{I}_{map} \times S^1$
$\overline{\mathbf{y}} := (\mathbf{y}, \varphi) = \overline{\mathbf{h}}^\mathcal{I}(\overline{\mathbf{x}}) \in \mathcal{F}^\mathcal{I}_{model} \times S^1$	Differential drive robot state in $\mathcal{F}^\mathcal{I}_{model} \times S^1$
$\xi^\mathcal{I} : S^1 \to S^1$	Angle transformation between $\mathcal{F}^\mathcal{I}_{map} \times S^1$ and $\mathcal{F}^\mathcal{I}_{model} \times S^1$
$\mathbf{y}_{d,\|}(\overline{\mathbf{y}}), \mathbf{y}_{d,G}(\overline{\mathbf{y}}) \in \mathcal{F}^\mathcal{I}_{model}$	Linear and angular local goals for $\overline{\mathbf{y}} \in \mathcal{F}^\mathcal{I}_{model} \times S^1$
$\overline{\mathbf{v}}^\mathcal{I} := (\hat{v}^\mathcal{I}, \hat{\omega}^\mathcal{I}) \in \mathbb{R}^2$	Linear and angular inputs for a unicycle robot in $\mathcal{F}^\mathcal{I}_{model} \times S^1$
$\overline{\mathbf{u}}^\mathcal{I} := (v^\mathcal{I}, \omega^\mathcal{I}) \in \mathbb{R}^2$	Linear and angular inputs for a unicycle robot in $\mathcal{F}^\mathcal{I}_{map} \times S^1$

Table 7.4: Key symbols related to the hybrid systems formulation (top - Section 7.4.1) and the reactive controller construction in each mode of the hybrid system (bottom - Section 7.4.2) for both a fully actuated robot and a differential drive robot.

In the following, Section 7.4.1 provides the hybrid systems description, Section 7.4.2 describes the reactive controller applied in each mode of the hybrid system, Section 7.4.3 summarizes the qualitative properties of our hybrid controller, and Section 7.4.4 describes our method of generating bounded inputs, each time for both the fully actuated and the differential drive robot. Table 7.4 summarizes associated notation used throughout this Section.

7.4.1 Hybrid Systems Description of Navigation Framework

Fully Actuated Robots

First, we consider a fully actuated particle with state $\mathbf{x} \in \mathcal{F}$, and dynamics

$$\dot{\mathbf{x}} = \mathbf{u} \tag{7.30}$$

Since different subsets of instantiated obstacles in $\tilde{\mathcal{P}}$, indexed by $\mathcal{I}$, result in different consolidated polygonal obstacles stored in $\mathcal{P}^{\mathcal{I}}_{map}$, it is natural to index the *modes of the hybrid controller* according to elements $\mathcal{I}$ of the power set $2^{\mathcal{N}_{\mathcal{P}}}$. Every execution, from any initial state, is required to start in the *initial mode*, indexed by $\mathcal{I} = \varnothing$. We also define a *terminal mode* as follows.

Definition 7.8. *The terminal mode of the hybrid system is indexed by the improper subset, $\mathcal{I} = \mathcal{N}_{\mathcal{P}}$, where all familiar obstacles in the workspace have been instantiated in the set $\mathcal{P}^{\mathcal{I}}_{sem}$, in the sense of Definition 7.1.*

We denote the freespace in the semantic, mapped and model spaces, associated with a unique subset $\mathcal{I}$ of $\mathcal{N}_{\mathcal{P}}$, by $\mathcal{F}^{\mathcal{I}}_{sem}, \mathcal{F}^{\mathcal{I}}_{map}, \mathcal{F}^{\mathcal{I}}_{model}$ respectively, as in Section 7.2. We also denote the corresponding perceived physical freespace by $\mathcal{F}^{\mathcal{I}}$, with $\mathcal{F}^{\mathcal{I}} := \mathcal{F}^{\mathcal{I}}_{map}$, since the dilation of obstacles by r in the passage from the physical to the semantic space and the obstacle merging in the passage from the semantic to the mapped space do not alter the freespace description. The domain D of our hybrid system is then defined as the collection $\mathsf{D} := \bigsqcup_{\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}} \mathcal{F}^{\mathcal{I}}$.

Following the notation in [87], we can then denote by $\Gamma \subset 2^{\mathcal{N}_\mathcal{P}} \times 2^{\mathcal{N}_\mathcal{P}}$ the *set of discrete transitions* for the hybrid system, forming a directed graph structure over the set of modes $2^{\mathcal{N}_\mathcal{P}}$. The collection of *guards* associated with Γ can be described as $\mathsf{G} := \bigsqcup_{(\mathcal{I},\mathcal{I}')\in\Gamma} G^{\mathcal{I},\mathcal{I}'}$, with $G^{\mathcal{I},\mathcal{I}'} \subset \mathcal{F}^{\mathcal{I}}$ given by

$$\begin{aligned} G^{\mathcal{I},\mathcal{I}'} :=& \{\mathbf{x} \in \mathcal{F}^{\mathcal{I}} \,|\, \mathcal{I}' = \mathcal{I} \cup \mathcal{I}_u \\ & \mathcal{I}_u \neq \varnothing, \mathcal{I}_u \cap \mathcal{I} = \varnothing, \\ & \overline{\mathsf{B}\left(\mathbf{x}, R\right)} \cap \tilde{P}_i \neq \varnothing \text{ for all } i \in \mathcal{I}_u, \\ & \overline{\mathsf{B}\left(\mathbf{x}, R\right)} \cap \tilde{\mathcal{P}}_{\mathcal{N}_\mathcal{P}\setminus(\mathcal{I}\cup\mathcal{I}_u)} = \varnothing\} \end{aligned} \tag{7.31}$$

with $\tilde{\mathcal{P}}_{\mathcal{N}_\mathcal{P}\setminus(\mathcal{I}\cup\mathcal{I}_u)} := \{\tilde{P}_i\}_{i\in\mathcal{N}_\mathcal{P}\setminus(\mathcal{I}\cup\mathcal{I}_u)}$.

Also, the *reset* $\mathsf{R} : \mathsf{G} \to \mathsf{D}$ is the continuous map that restricts simply as $R^{\mathcal{I},\mathcal{I}'} := \mathsf{R}|_{G^{\mathcal{I},\mathcal{I}'}} : G^{\mathcal{I},\mathcal{I}'} \to \mathcal{F}^{\mathcal{I}'}$, with

$$R^{\mathcal{I},\mathcal{I}'}(\mathbf{x}) = \mathbf{x} \tag{7.32}$$

the identity map. Note, however, that although the robot cannot experience discrete jumps in the physical space, the model space $\mathcal{F}^{\mathcal{I}'}_{model}$ is likely to be a discontinuously different space from $\mathcal{F}^{\mathcal{I}}_{model}$ (i.e., there is no guaranteed inclusion from $\mathcal{F}^{\mathcal{I}'}_{model}$ into $\mathcal{F}^{\mathcal{I}}_{model}$), hence the model position in the new space bears no obvious relationship to that in the prior. Namely, the position of the robot in the model space after a transition from mode $\mathcal{I}$ to mode $\mathcal{I}'$ will be given by $\mathbf{h}^{\mathcal{I}'} \circ (\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{y})$, with $\mathbf{y} \in \mathcal{F}^{\mathcal{I}}_{model}$.

Finally, we can construct the *hybrid vector field* $\mathsf{U} : \mathsf{D} \to T\mathsf{D}$ that restricts to a vector field $U^{\mathcal{I}} := \mathsf{U}|_{\mathcal{F}^{\mathcal{I}}} : \mathcal{F}^{\mathcal{I}} \to T\mathcal{F}^{\mathcal{I}}$, that can be written as

$$U^{\mathcal{I}}(\mathbf{x}) := \mathbf{u}^{\mathcal{I}}(\mathbf{x}) \tag{7.33}$$

with $\mathbf{u}^{\mathcal{I}}$ given in (7.40) and described in the next Section.

Based on the above definitions, we define the *navigational hybrid system for fully actuated robots* as the tuple $\mathsf{H} := \left(2^{\mathcal{N}_\mathcal{P}}, \Gamma, \mathsf{D}, \mathsf{U}, \mathsf{G}, \mathsf{R}\right)$ describing the modes, discrete transitions,

domains, associated vector fields, guards and resets.

Differential Drive Robots

Next, we focus on a differential drive robot, whose state is $\overline{\mathbf{x}} := (\mathbf{x}, \psi) \in \mathcal{F} \times S^1 \subset SE(2)$, and its dynamics are given by

$$\dot{\overline{\mathbf{x}}} = \mathbf{B}(\psi)\overline{\mathbf{u}} \tag{7.34}$$

with $\mathbf{B}(\psi) := \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top$ and $\overline{\mathbf{u}} := (v, \omega)$, with $v, \omega \in \mathbb{R}$ the linear and angular input respectively.

The analysis here is fairly similar; the modes and discrete transitions are identical. However, the robot operates on a subset of $SE(2)$ and, therefore, the domains must be described as $\overline{\mathsf{D}} := \bigsqcup_{\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}} (\mathcal{F}^\mathcal{I} \times S^1)$. Consequently, the collection of *guards* that will result in transitions between different modes according to $\mathsf{\Gamma}$ are described as $\overline{\mathsf{G}} := \bigsqcup_{(\mathcal{I},\mathcal{I}') \in \mathsf{\Gamma}} \overline{G}^{\mathcal{I},\mathcal{I}'}$, with $\overline{G}^{\mathcal{I},\mathcal{I}'} \subset (\mathcal{F}^\mathcal{I} \times S^1)$ given by

$$\begin{aligned} \overline{G}^{\mathcal{I},\mathcal{I}'} := \{ & \overline{\mathbf{x}} = (\mathbf{x}, \psi) \in \mathcal{F}^\mathcal{I} \times S^1 \,|\, \mathcal{I}' = \mathcal{I} \cup \mathcal{I}_u \\ & \mathcal{I}_u \neq \varnothing, \mathcal{I}_u \cap \mathcal{I} = \varnothing, \\ & \overline{\mathsf{B}\,(\mathbf{x}, R)} \cap \tilde{P}_i \neq \varnothing \text{ for all } i \in \mathcal{I}_u, \\ & \overline{\mathsf{B}\,(\mathbf{x}, R)} \cap \tilde{\mathcal{P}}_{\mathcal{N}_\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{I}_u)} = \varnothing \} \end{aligned} \tag{7.35}$$

Also, the *reset* $\overline{\mathsf{R}} : \overline{\mathsf{G}} \to \overline{\mathsf{D}}$ is the continuous map that restricts simply as $\overline{R}^{\mathcal{I},\mathcal{I}'} := \overline{\mathsf{R}}|_{\overline{G}^{\mathcal{I},\mathcal{I}'}} : \overline{G}^{\mathcal{I},\mathcal{I}'} \to (\mathcal{F}^{\mathcal{I}'} \times S^1)$, with

$$\overline{R}^{\mathcal{I},\mathcal{I}'}(\overline{\mathbf{x}}) = \overline{\mathbf{x}} \tag{7.36}$$

the identity map.

Finally, the fact that the robot operates in $SE(2)$ gives rise to a new *hybrid vector field* $\overline{\mathsf{U}} : \overline{\mathsf{D}} \to T\overline{\mathsf{D}}$, that restricts to a vector field $\overline{U}^\mathcal{I} := \overline{\mathsf{U}}|_{\mathcal{F}^\mathcal{I} \times S^1} : \mathcal{F}^\mathcal{I} \times S^1 \to T(\mathcal{F}^\mathcal{I} \times S^1)$, that can be written as

$$\overline{U}^\mathcal{I}(\overline{\mathbf{x}}) := \mathbf{B}(\psi)\overline{\mathbf{u}}^\mathcal{I} \tag{7.37}$$

with the inputs $\overline{\mathbf{u}}^{\mathcal{I}} = (v^{\mathcal{I}}, \omega^{\mathcal{I}})$ given as in (7.54) and described in the next Section.

Based on the above definitions, we define the *navigational hybrid system for differential drive robots* as the tuple $\overline{\mathsf{H}} := \left(2^{\mathcal{N}_{\mathcal{P}}}, \Gamma, \overline{\mathsf{D}}, \overline{\mathsf{U}}, \overline{\mathsf{G}}, \overline{\mathsf{R}}\right)$ describing the modes, discrete transitions, domains, associated vector fields, guards and resets.

7.4.2 Reactive Controller in Each Hybrid Mode

The preceding analysis of the hybrid system allows us to now describe the constituent controllers in each mode $\mathcal{I}$ of the hybrid system, for both the fully actuated and the differential drive robot. For the results pertaining to each separate mode, we are going to assume that $\mathcal{I}$ describes the terminal mode of the hybrid system, in the notion of Definition 7.8.

With this assumption, we can arrive to Theorems 7.1 and 7.2, that allow us to establish the main results about our hybrid controller in Theorems 7.3 and 7.4. We assume that the robot operates in $\mathcal{F}^{\mathcal{I}}_{map}$[5], and the set of consolidated obstacles $\mathcal{P}^{\mathcal{I}}_{map}$ in mode $\mathcal{I}$ has been identified.

Fully Actuated Robots

The dynamics of the fully actuated particle in $\mathcal{F}^{\mathcal{I}}_{model}$ with state $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x}) \in \mathcal{F}^{\mathcal{I}}_{model}$ can be described by $\dot{\mathbf{y}} = \mathbf{v}^{\mathcal{I}}(\mathbf{y})$ with the input $\mathbf{v}^{\mathcal{I}}(\mathbf{y})$ given in [7] (Section 3.2) as

$$\mathbf{v}^{\mathcal{I}}(\mathbf{y}) = -\left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \tag{7.38}$$

with $\mathbf{y}_d = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$, and the convex *local freespace* for $\mathbf{y}$, $\mathcal{LF}(\mathbf{y})$, defined as the Voronoi cell in [7, Eqns. (7), (24)]:

$$\mathcal{LF}(\mathbf{y}) := \left\{\mathbf{q} \in \mathcal{F}^{\mathcal{I}}_{model} \,|\, ||\mathbf{q} - \mathbf{y}|| \leq ||\mathbf{q} - \Pi_{\overline{O}_i}(\mathbf{y})||, \forall i\right\} \cap \overline{\mathsf{B}\left(\mathbf{y}, \tfrac{R_{model}}{2}\right)} \tag{7.39}$$

[5]This is afforded by the fact that the perceived physical freespace $\mathcal{F}^{\mathcal{I}}$ was explicitly constructed in Section 7.4.1 to be equal to $\mathcal{F}^{\mathcal{I}}_{map}$. To be accurate, one must write the identity map from $\mathcal{F}^{\mathcal{I}}$ into $\mathcal{F}^{\mathcal{I}}_{map}$ as $\iota : \mathcal{F}^{\mathcal{I}} \to \mathcal{F}^{\mathcal{I}}_{map}$ and, subsequently, define the control in the physical space as $[D_{\mathbf{x}}\iota]^{-1}\mathbf{u}^{\mathcal{I}}$, with $\mathbf{u}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to T\mathcal{F}^{\mathcal{I}}_{map}$ the control strategy in the mapped space, described next. However, since $[D_{\mathbf{x}}\iota]$ resolves to the identity matrix, this construction reduces to direct application of $\mathbf{u}^{\mathcal{I}}$ on the physical space.

Here, i spans the obstacles in both $\mathcal{D}^{\mathcal{I}}_{map}$ (represented in $\mathcal{F}^{\mathcal{I}}_{model}$ by $\overline{\mathsf{B}\left(\mathbf{x}^*_i, \rho_i\right)}$) and $\mathcal{C}_{map}$ (transferred to $\mathcal{F}^{\mathcal{I}}_{model}$ with an identity map), and R_{model} is the range of the virtual sensor used for obstacle detection in $\mathcal{F}^{\mathcal{I}}_{model}$. Similarly to Chapter 6, using the diffeomorphism construction in (7.29), we construct our controller as the vector field $\mathbf{u}^{\mathcal{I}} =: \mathcal{F}^{\mathcal{I}}_{map} \to T\mathcal{F}^{\mathcal{I}}_{map}$ given by

$$\mathbf{u}^{\mathcal{I}}(\mathbf{x}) = k \left[D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}\right]^{-1} \cdot \left(\mathbf{v}^{\mathcal{I}} \circ \mathbf{h}^{\mathcal{I}}(\mathbf{x})\right) \tag{7.40}$$

with $k > 0$. Note here that the strategy employed never requires the explicit computation of $(\mathbf{h}^{\mathcal{I}})^{-1}$, which would make our numerical realization quite difficult; instead, it merely requires inversion of $[D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]$.

We notice that if the range of the virtual sensor R_{model} used to construct $\mathcal{LF}(\mathbf{y})$ in the model space is smaller than the range of our sensor R, the vector field $\mathbf{u}^{\mathcal{I}}$ is Lipschitz continuous since $\mathbf{v}^{\mathcal{I}}(\mathbf{y})$ is shown to be Lipschitz continuous in [7], $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ is a smooth change of coordinates away from sharp corners, and the robot discovers obstacles before actually using them for navigation, because $R_{model} < R$. We are led to the following result.

Corollary 7.4. *With $\mathcal{I}$ the terminal mode of the hybrid controller, the vector field $\mathbf{u}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to T\mathcal{F}^{\mathcal{I}}_{map}$ generates a unique continuously differentiable partial flow.*

To ensure completeness (i.e., absence of finite time escape through boundaries in $\mathcal{F}^{\mathcal{I}}_{map}$) we must verify that the robot never collides with any obstacle in the environment, i.e., leaves its freespace positively invariant. However, this property follows almost directly from the fact that the vector field $\mathbf{u}^{\mathcal{I}}$ on $\mathcal{F}^{\mathcal{I}}_{map}$ is the pushforward of the complete vector field $\mathbf{v}^{\mathcal{I}}$ through $(\mathbf{h}^{\mathcal{I}})^{-1}$, guaranteed to insure that $\mathcal{F}^{\mathcal{I}}_{model}$ remain positively invariant under its flow as shown in [7], away from sharp corners on the boundary of $\mathcal{F}^{\mathcal{I}}_{map}$. Therefore, we immediately get the following result.

Proposition 7.3. *With $\mathcal{I} = \mathcal{N}_{\mathcal{P}}$ the terminal mode of the hybrid controller, the freespace interior $\mathcal{F}^{\mathcal{I}}_{map}$ is positively invariant under the law* (7.40).

Next, we focus on the stationary points of $\mathbf{u}^{\mathcal{I}}$.

Lemma 7.6. *With $\mathcal{I} = \mathcal{N}_{\mathcal{P}}$ the terminal mode of the hybrid controller:*

1. *The set of stationary points of control law* (7.40) *is given as*

$$\{\mathbf{x}_d\} \bigcup \{(\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{s}_i)\}_{i \in \mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \bigcup \{\mathcal{G}_k\}_{k \in \mathcal{J}_{\mathcal{C}}}$$

where

$$\mathbf{s}_i = \mathbf{x}_i^* - \rho_i \frac{\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d) - \mathbf{x}_i^*}{||\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d) - \mathbf{x}_i^*||} \tag{7.41a}$$

$$\mathcal{G}_k = \left\{ \mathbf{q} \in \mathcal{F}_{map}^{\mathcal{I}} \middle| d(\mathbf{q}, C_k) = r, \kappa(\mathbf{q}) = 1 \right\} \tag{7.41b}$$

with

$$\kappa(\mathbf{q}) := \frac{(\mathbf{q} - \Pi_{\overline{C}_k}(\mathbf{q}))^\top (\mathbf{q} - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))}{||\mathbf{q} - \Pi_{\overline{C}_k}(\mathbf{q})|| \cdot ||\mathbf{q} - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||}$$

2. *The goal* $\mathbf{x}_d$ *is the only locally stable equilibrium of control law* (7.40) *and all the other stationary points* $\{(\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{s}_i)\}_{i \in \mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \bigcup \{\mathcal{G}_k\}_{k \in \mathcal{J}_{\mathcal{C}}}$*, each associated with an obstacle, are nondegenerate saddles.*

Proof. Included in Appendix C.5.2. □

Note that there is a slight complication here; each stationary point $\mathbf{s}_i, i \in \mathcal{J}_{\mathcal{D}}^{\mathcal{I}}$ lies on the boundary of the corresponding ball $\overline{\mathsf{B}(\mathbf{x}_i^*, \rho_i)}$ in the model space and thus, by construction of the diffeomorphism $\mathbf{h}^{\mathcal{I}}$, it might not lie in the domain of $(\mathbf{h}^{\mathcal{I}})^{-1}$ because it could correspond to a sharp corner (i.e., a polygon vertex) in the mapped space. Although such problems can only occur for a thin subset of obstacle placements, we explicitly impose the following assumption to facilitate our formal results.

Assumption 7.5. *The stationary points of control law* (7.38) *in the model space lie in the domain of the map* $(\mathbf{h}^{\mathcal{I}})^{-1}$ *between* $\mathcal{F}_{model}^{\mathcal{I}}$ *and* $\mathcal{F}_{map}^{\mathcal{I}}$*.*

Then, using Lemma 7.6, we arrive at the following result, that establishes (almost) global convergence to the goal $\mathbf{x}_d$.

Proposition 7.4. *With* $\mathcal{I}$ *the terminal mode of the hybrid controller, the goal* $\mathbf{x}_d$ *is an*

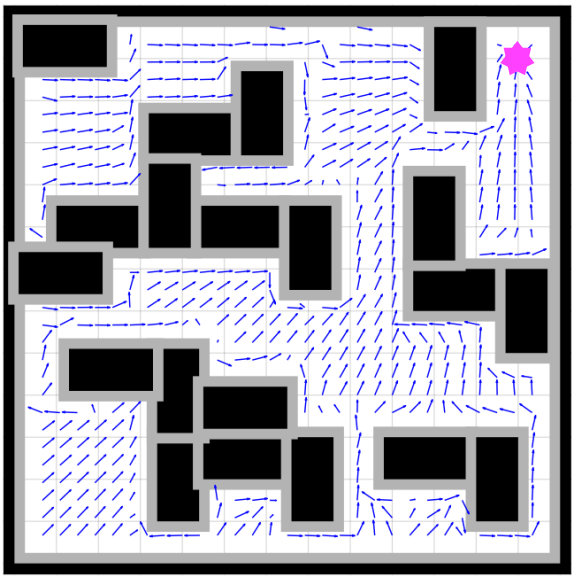

Figure 7.6: Depiction of the vector field in (7.40) for the terminal mode $\mathcal{I}$ from one of our numerical examples presented in Section 7.6 with several overlapping obstacles. Notice how the vector field guarantees safety around each obstacle, with the goal in purple attracting globally.

asymptotically stable equilibrium of (7.40)*, whose region of attraction includes the freespace* $\mathcal{F}^{\mathcal{I}}_{map}$ *except a set of measure zero.*

Proof. Included in Appendix C.5.2. □

We can now immediately conclude the following central summary statement.

Theorem 7.1. *With* $\mathcal{I}$ *the terminal mode of the hybrid controller, the reactive controller in* (7.40) *leaves the freespace* $\mathcal{F}^{\mathcal{I}}_{map}$ *positively invariant, and its unique continuously differentiable flow, starting at almost any robot placement* $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map}$*, asymptotically reaches the goal location* $\mathbf{x}_d$*, while strictly decreasing* $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$ *along the way.*

A depiction of the vector field in (7.40) for the terminal mode $\mathcal{I}$ (Definition 7.8) from one of our numerical examples presented in Section 7.6 is included in Fig. 7.6.

Differential Drive Robots

Since the robot operates in $SE(2)$ instead of $\mathbb{R}^2$, we first need to come up with a smooth diffeomorphism $\overline{\mathbf{h}}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \times S^1 \to \mathcal{F}^{\mathcal{I}}_{model} \times S^1$ away from sharp corners on the boundary of $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$, and then establish the results about our controller.

Following Section 6.3, we construct our map $\overline{\mathbf{h}}^{\mathcal{I}}$ from $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ to $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$ as

$$\overline{\mathbf{y}} = (\mathbf{y}, \varphi) = \overline{\mathbf{h}}^{\mathcal{I}}(\overline{\mathbf{x}}) := (\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \xi^{\mathcal{I}}(\overline{\mathbf{x}})) \tag{7.42}$$

with $\overline{\mathbf{x}} = (\mathbf{x}, \psi) \in \mathcal{F}^{\mathcal{I}}_{map} \times S^1$, $\overline{\mathbf{y}} := (\mathbf{y}, \varphi) \in \mathcal{F}^{\mathcal{I}}_{model} \times S^1$ and

$$\varphi = \xi^{\mathcal{I}}(\overline{\mathbf{x}}) := \angle(\mathbf{e}(\overline{\mathbf{x}})) \tag{7.43}$$

Here, $\angle\mathbf{e} := \text{atan2}(e_2, e_1)$ and

$$\mathbf{e}(\overline{\mathbf{x}}) = \Pi_{\mathbf{y}} \cdot D_{\overline{\mathbf{x}}}\overline{\mathbf{h}}^{\mathcal{I}} \cdot \mathbf{B}(\psi) \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}} \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \tag{7.44}$$

with $\Pi_{\mathbf{y}}$ denoting the projection onto the first two components. The reason for choosing φ as in (7.43) will become evident later, in our effort to control the equivalent differential drive robot dynamics in $\mathcal{F}^{\mathcal{I}}_{model}$.

Proposition 7.5. *The map $\overline{\mathbf{h}}^{\mathcal{I}}$ in (7.42) is a C^∞ diffeomorphism from $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ to $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$ away from sharp corners, none of which lie in the interior of $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$.*

Proof. Included in Appendix C.5.2. □

Then, using (7.42), we can find the pushforward of the differential drive robot dynamics in (7.34) as

$$\begin{aligned} \dot{\overline{\mathbf{y}}} =& \frac{d}{dt} \begin{bmatrix} \mathbf{h}^{\mathcal{I}}(\mathbf{x}) \\ \xi^{\mathcal{I}}(\overline{\mathbf{x}}) \end{bmatrix} \\ =& \left[D_{\overline{\mathbf{x}}}\overline{\mathbf{h}}^{\mathcal{I}} \circ (\overline{\mathbf{h}}^{\mathcal{I}})^{-1} \right] \cdot \left(\mathbf{B} \circ (\overline{\mathbf{h}}^{\mathcal{I}})^{-1}(\overline{\mathbf{y}}) \right) \cdot \overline{\mathbf{u}}^{\mathcal{I}} \end{aligned} \tag{7.45}$$

Based on the above, we can then write

$$\dot{\overline{\mathbf{y}}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\varphi} \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} \mathbf{h}^{\mathcal{I}}(\mathbf{x}) \\ \xi^{\mathcal{I}}(\overline{\mathbf{x}}) \end{bmatrix} = \mathbf{B}(\varphi)\overline{\mathbf{v}}^{\mathcal{I}} \tag{7.46}$$

with $\overline{\mathbf{v}}^{\mathcal{I}} = (\hat{v}^{\mathcal{I}}, \hat{\omega}^{\mathcal{I}})$, and the inputs $(\hat{v}^{\mathcal{I}}, \hat{\omega}^{\mathcal{I}})$ related to $(v^{\mathcal{I}}, \omega^{\mathcal{I}})$ through

$$\hat{v}^{\mathcal{I}} = ||\mathbf{e}(\overline{\mathbf{x}})||\, v^{\mathcal{I}} \tag{7.47}$$

$$\hat{\omega}^{\mathcal{I}} = v^{\mathcal{I}} D_{\mathbf{x}}\xi^{\mathcal{I}} \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} + \frac{\partial \xi^{\mathcal{I}}}{\partial \psi}\omega^{\mathcal{I}} \tag{7.48}$$

with $D_{\mathbf{x}}\xi^{\mathcal{I}} = \begin{bmatrix} \frac{\partial \xi^{\mathcal{I}}}{\partial x} & \frac{\partial \xi^{\mathcal{I}}}{\partial y} \end{bmatrix}$. Here, we can calculate

$$D_{\mathbf{x}}\xi^{\mathcal{I}} \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} = \frac{\alpha_1(\overline{\mathbf{x}})\alpha_3(\overline{\mathbf{x}}) + \alpha_2(\overline{\mathbf{x}})\alpha_4(\overline{\mathbf{x}})}{||\mathbf{e}(\overline{\mathbf{x}})||^2} \tag{7.49}$$

with the auxiliary terms $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ defined as

$$\alpha_1(\overline{\mathbf{x}}) := -\left([D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{21}\cos\psi + [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{22}\sin\psi\right) \tag{7.50}$$

$$\alpha_2(\overline{\mathbf{x}}) := [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{11}\cos\psi + [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{12}\sin\psi \tag{7.51}$$

$$\begin{aligned} \alpha_3(\overline{\mathbf{x}}) :=& \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{11}}{\partial [\mathbf{x}]_1}\cos^2\psi + \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{12}}{\partial [\mathbf{x}]_2}\sin^2\psi \\ &+ \left(\frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{11}}{\partial [\mathbf{x}]_2} + \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{12}}{\partial [\mathbf{x}]_1}\right)\sin\psi\cos\psi \end{aligned} \tag{7.52}$$

$$\begin{aligned} \alpha_4(\overline{\mathbf{x}}) :=& \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{21}}{\partial [\mathbf{x}]_1}\cos^2\psi + \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{22}}{\partial [\mathbf{x}]_2}\sin^2\psi \\ &+ \left(\frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{21}}{\partial [\mathbf{x}]_2} + \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{22}}{\partial [\mathbf{x}]_1}\right)\sin\psi\cos\psi \end{aligned} \tag{7.53}$$

We provide more details about the calculation of partial derivatives for elements of $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}$ used above in Section 7.5 and in Appendix B.2.

Hence, we have found equivalent differential drive robot dynamics, defined on $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$.

The idea now is to use the control strategy in [7] (Section 3.2) for the dynamical system in (7.46) to find inputs $\hat{v}^{\mathcal{I}}, \hat{\omega}^{\mathcal{I}}$ in $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$, and then use (7.47), (7.48) to find the actual inputs $v^{\mathcal{I}}, \omega^{\mathcal{I}}$ in $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ that achieve $\hat{v}^{\mathcal{I}}, \hat{\omega}^{\mathcal{I}}$ as

$$v^{\mathcal{I}} = \frac{k_v\, \hat{v}^{\mathcal{I}}}{||\mathbf{e}(\overline{\mathbf{x}})||} \tag{7.54a}$$

$$\omega^{\mathcal{I}} = \left(\frac{\partial \xi^{\mathcal{I}}}{\partial \psi}\right)^{-1} \left(k_\omega\, \hat{\omega}^{\mathcal{I}} - v^{\mathcal{I}} D_{\mathbf{x}} \xi^{\mathcal{I}} \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \right) \tag{7.54b}$$

with $k_v, k_\omega > 0$ fixed gains.

Namely, we design our inputs $\hat{v}^{\mathcal{I}}$ and $\hat{\omega}^{\mathcal{I}}$ as in (3.8) - (3.9)

$$\hat{v}^{\mathcal{I}} = -\begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}^{\top} \left(\mathbf{y} - \mathbf{y}_{d,\|}(\overline{\mathbf{y}})\right) \tag{7.55a}$$

$$\hat{\omega}^{\mathcal{I}} = \operatorname{atan}\left(\frac{\begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}^{\top} (\mathbf{y} - \mathbf{y}_{d,G}(\overline{\mathbf{y}}))}{\begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}^{\top} (\mathbf{y} - \mathbf{y}_{d,G}(\overline{\mathbf{y}}))} \right) \tag{7.55b}$$

with $\mathcal{LF}(\mathbf{y}) \subset \mathcal{F}^{\mathcal{I}}_{model}$ the convex polygon defining the local freespace at $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$, and linear and angular local goals $\mathbf{y}_{d,\|}(\overline{\mathbf{y}}), \mathbf{y}_{d,G}(\overline{\mathbf{y}})$ given by

$$\mathbf{y}_{d,\|}(\overline{\mathbf{y}}) := \Pi_{\mathcal{LF}(\mathbf{y}) \cap H_{\|}}(\mathbf{y}_d) \tag{7.56}$$

$$\mathbf{y}_{d,G}(\overline{\mathbf{y}}) := \frac{\Pi_{\mathcal{LF}(\mathbf{y}) \cap H_G}(\mathbf{y}_d) + \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)}{2} \tag{7.57}$$

with $H_{\|}$ and H_G the lines defined in [7] (see also (3.12) - (3.13)) as

$$H_{\|} = \left\{ \mathbf{z} \in \mathcal{F}^{\mathcal{I}}_{model} \,\middle|\, \begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}^{\top} (\mathbf{z} - \mathbf{y}) = 0 \right\} \tag{7.58}$$

$$H_G = \left\{ \alpha\mathbf{y} + (1-\alpha)\mathbf{y}_d \in \mathcal{F}^{\mathcal{I}}_{model} \,|\, \alpha \in \mathbb{R} \right\} \tag{7.59}$$

The properties of the differential drive robot control law given in (7.54) can be summarized in the following theorem.

Theorem 7.2. *With $\mathcal{I}$ the terminal mode of the hybrid controller, the reactive controller for differential drive robots, given in* (7.54)*, leaves the freespace $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ positively invariant, and its unique continuously differentiable flow, starting at almost any robot configuration $(\mathbf{x}, \psi) \in \mathcal{F}^{\mathcal{I}}_{map} \times S^1$, asymptotically steers the robot to the goal location $\mathbf{x}_d$, without increasing $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$ along the way.*

Proof. Included in Appendix C.5.2. □

7.4.3 Qualitative Properties of the Hybrid Controller

Fully Actuated Robots

First, we show that the navigational hybrid system H inherits the fundamental consistency properties outlined in [87, Theorems 5-9], in order to establish that the hybrid system is well-behaved in the sense of being both deterministic and non-blocking (i.e., generating executions defined for all future times).

Lemma 7.7. *The hybrid system H has disjoint guards.*

Proof. Included in Appendix C.5.2. □

An immediate result following Lemma 7.7, that does not allow a robot state $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}$ to be contained in more than one guard $G^{\mathcal{I},\mathcal{I}'}$, and the nice properties of the flow in each separate mode, summarized in Corollary 7.4, is the following important consistency property.

Corollary 7.5. *The hybrid system* H *is deterministic.*

Next, we focus on the non-blocking property. As stated in [87], a hybrid execution might be blocked either by conventional finite escape through the boundary of the hybrid domain at a point in the complement of all the guards, by escape through a point in the guard whose reset lies outside of the hybrid domain, or by hybrid ambiguity, i.e., by arriving at a point through the continuous flow that lies in the complement of the guard G and yet still on the boundary of G. We eliminate all cases in the proof of the following result.

Lemma 7.8. *The hybrid system* H *is non-blocking.*

Proof. Included in Appendix C.5.2. □

Finally, using the last part of the proof of Lemma 7.8 which shows that the (identity) reset from a given mode cannot lie in the guard of the next mode, we arrive at the following result about the discrete transitions of the hybrid system H.

Corollary 7.6. *An execution of the hybrid system* H *undergoes no more than one hybrid transition at a single time* t*.*

Based on the above, the central result about the hybrid controller for a fully actuated robot can be summarized in the following Theorem.

Theorem 7.3. *For a fully actuated robot with dynamics defined in* (7.30)*, the deterministic, non-blocking navigational hybrid system* $\mathsf{H} := \left(2^{\mathcal{N}_\mathcal{P}}, \Gamma, \mathsf{D}, \mathsf{U}, \mathsf{G}, \mathsf{R}\right)$*, with the restrictions of guards* G*, resets* R *and vector fields* U *defined as in* (7.31)*,* (7.32) *and* (7.33) *respectively, leaves the free space* $\mathcal{F}$ *positively invariant under the Lipschitz continuous, piecewise smooth flow associated with each of its hybrid domains, and, starting at almost any robot placement* $\mathbf{x} \in \mathcal{F}$ *at time* t_0 *with an initial mode* $\mathcal{I} = \varnothing$*, asymptotically reaches a designated goal location* $\mathbf{x}_d \in \mathcal{F}$*, in a previously unexplored environment satisfying Assumptions 7.1 - 7.4, with a uniquely defined (in both state and mode) execution for all* $t > t_0$*.*

Proof. Included in Appendix C.5.2. □

Differential Drive Robots

We can then follow exactly the same procedure to prove the following statement for the hybrid controller for differential drive robots.

Theorem 7.4. *For a differential drive robot with dynamics defined in* (7.34)*, the deterministic, non-blocking navigational hybrid system* $\overline{\mathsf{H}} := \left(2^{\mathcal{N}_\mathcal{P}}, \Gamma, \overline{\mathsf{D}}, \overline{\mathsf{U}}, \overline{\mathsf{G}}, \overline{\mathsf{R}}\right)$*, with the restrictions of guards* $\overline{\mathsf{G}}$*, resets* $\overline{\mathsf{R}}$ *and vector fields* $\overline{\mathsf{U}}$ *defined as in* (7.35), (7.36) *and* (7.37) *respectively, leaves the free space* $\mathcal{F} \times S^1$ *positively invariant under the Lipschitz continuous, piecewise smooth flow associated with each of its hybrid domains, and, starting at almost any robot placement* $\overline{\mathbf{x}} \in \mathcal{F} \times S^1$ *at time* t_0 *with an initial mode* $\mathcal{I} = \varnothing$*, asymptotically reaches a designated goal location* $\mathbf{x}_d \in \mathcal{F}$*, in a previously unexplored environment satisfying Assumptions 7.1 - 7.4, with a uniquely defined (in both state and mode) execution for all* $t > t_0$*.*

7.4.4 Generating Bounded Inputs

Although the control inputs for both a fully actuated robot and a differential drive robot, described in (7.40) and (7.54) respectively, can be used in the hybrid systems description of the controller (see (7.33) and (7.37)) to yield the desired results of Theorems 7.3 and 7.4, we have so far implicitly assumed that there is no bound in the magnitude of $\mathbf{u}^\mathcal{I}$ in (7.40) or the magnitudes of $v^\mathcal{I}, \omega^\mathcal{I}$ in (7.54) for each separate mode $\mathcal{I}$. In this Section, we show how to generate bounded inputs without affecting the results of Theorems 7.3 and 7.4.

Fully Actuated Robots

We focus on fully actuated robots first. Let $\mathbf{u}^\mathcal{I}_{nom} : \mathcal{F}^\mathcal{I}_{map} \to T\mathcal{F}^\mathcal{I}_{map}$ denote the nominal input for mode $\mathcal{I}$, defined using (7.40) as

$$\mathbf{u}^\mathcal{I}_{nom}(\mathbf{x}) := \left[D_\mathbf{x}\mathbf{h}^\mathcal{I}\right]^{-1} \cdot \left(\mathbf{v}^\mathcal{I} \circ \mathbf{h}^\mathcal{I}(\mathbf{x})\right) \tag{7.60}$$

We can then easily satisfy the requirement $||\mathbf{u}^\mathcal{I}|| \leq u_{\max}$ by picking a gain k such that

$0 < k \leq u_{\max}$ and defining our controller as

$$\mathbf{u}^{\mathcal{I}}(\mathbf{x}) := k \frac{\mathbf{u}^{\mathcal{I}}_{nom}(\mathbf{x})}{||\mathbf{u}^{\mathcal{I}}_{nom}(\mathbf{x})|| + \epsilon_{\mathbf{u}}} \tag{7.61}$$

with $\epsilon_{\mathbf{u}} > 0$ a small number. The modified (bounded) controller in (7.61) does not affect the results of Theorem 7.3, since it maintains the heading direction of the original (unbounded) controller in (7.40), and just limits its magnitude.

Differential Drive Robots

The analysis is slightly more complicated for differential drive robots, since we have to respect the fact that the actual inputs $v^{\mathcal{I}}$, $\omega^{\mathcal{I}}$ are related to the inputs $\hat{v}^{\mathcal{I}}$, $\hat{\omega}^{\mathcal{I}}$ through (7.54). However, an important observation, deriving from the proof of Theorem 7.2, is that the choice of gains $k_v, k_\omega > 0$ in (7.54) does not affect the positive invariance or convergence properties of the controller, which rely entirely on $\hat{v}^{\mathcal{I}}$, $\hat{\omega}^{\mathcal{I}}$, given in (7.55).

Therefore, the main idea is to adaptively change the gains online, in order to satisfy the constraints $|v^{\mathcal{I}}| \leq v_{\max}$, $|\omega^{\mathcal{I}}| \leq \omega_{\max}$. Namely, using (7.54), we look for gains $k_v(\overline{\mathbf{x}}), k_\omega(\overline{\mathbf{x}})$ such that

$$k_v(\overline{\mathbf{x}}) \frac{|\hat{v}^{\mathcal{I}}(\overline{\mathbf{x}})|}{||\mathbf{e}(\overline{\mathbf{x}})||} \leq v_{\max}$$
$$\left| k_\omega(\overline{\mathbf{x}})\, \hat{\omega}^{\mathcal{I}}(\overline{\mathbf{x}}) - k_v(\overline{\mathbf{x}}) \frac{\hat{v}^{\mathcal{I}}(\overline{\mathbf{x}})}{||\mathbf{e}(\overline{\mathbf{x}})||} \vartheta(\psi) \right| \leq \frac{\partial \xi^{\mathcal{I}}}{\partial \psi} \omega_{\max}$$

with $\vartheta(\psi) := D_{\mathbf{x}} \xi^{\mathcal{I}} \begin{bmatrix} \cos\psi & \sin\psi \end{bmatrix}^\top$, since $\frac{\partial \xi^{\mathcal{I}}}{\partial \psi} > 0$, as shown in the proof of Proposition 7.5. A conservative selection of gains that satisfies the above constraints can then be extracted using the triangle inequality as follows

$$k_v(\overline{\mathbf{x}}) = \min \Bigg(k_{v,nom}, \frac{||\mathbf{e}(\overline{\mathbf{x}})||}{|\hat{v}^{\mathcal{I}}(\overline{\mathbf{x}})|} v_{\max}, \lambda \frac{\partial \xi^{\mathcal{I}}}{\partial \psi} \frac{||\mathbf{e}(\overline{\mathbf{x}})||}{|\hat{v}^{\mathcal{I}}(\overline{\mathbf{x}})|\, |\vartheta(\psi)|} \omega_{\max} \Bigg) \tag{7.63}$$

$$k_\omega(\overline{\mathbf{x}}) = \min \left(k_{\omega,nom}, (1-\lambda) \frac{\partial \xi^{\mathcal{I}}}{\partial \psi} \frac{\omega_{\max}}{|\hat{\omega}^{\mathcal{I}}(\overline{\mathbf{x}})|} \right) \tag{7.64}$$

with $k_{v,nom}, k_{\omega,nom} > 0$ initially provided nominal gains and $\lambda \in (0,1)$ a tuning parameter. It can be seen that $k_v(\overline{\mathbf{x}}), k_\omega(\overline{\mathbf{x}})$ are always positive since $||\mathbf{e}(\overline{\mathbf{x}})||, \frac{\partial \xi^{\mathcal{I}}}{\partial \psi}$ are always positive.

7.5 Online Reactive Planning Algorithms

With the description of the diffeomorphism construction and the overall hybrid controller, we are now ready to describe the algorithm we use during execution time to generate our control inputs. As shown in Fig. 7.2 that summarizes the whole architecture, we divide the main algorithm that communicates with the semantic mapping and the perception pipelines[6] in two distinct components. First, the *mapped space recovery* component, described in Section 7.5.1, is responsible for keeping track of all encountered objects, and extracting the sets of obstacles $\mathcal{D}^{\mathcal{I}}_{map}$, $\mathcal{B}^{\mathcal{I}}_{map}$. Next, the *reactive planning* component, described in Section 7.5.2, uses the input from the mapped space recovery component to generate the diffeomorphism $\mathbf{h}^{\mathcal{I}}$ (described in Section 7.3) between the mapped space and the model space during execution time, and provide the commands for the robot according to the hybrid controller (described in Section 7.4).

7.5.1 Mapped Space Recovery

Given as input the aggregated set of localized, recognized familiar obstacles $\tilde{\mathcal{P}}_{\mathcal{I}}$, we first dilate all these elements of $\tilde{\mathcal{P}}_{\mathcal{I}}$ by the robot radius r, to form the components of $\mathcal{P}^{\mathcal{I}}_{sem}$, and consolidate the connected components resulting from their union into a new set of merged obstacles to form $\mathcal{P}^{\mathcal{I}}_{map}$. Then, for each connected component P of $\mathcal{P}^{\mathcal{I}}_{map}$ that intersects the boundary of the enclosing freespace $\mathcal{F}_e$, we take $B = P \cap \mathcal{F}_e$, as described in Section 7.2.3, and include B in the list of obstacles to be merged into $\partial\mathcal{F}_e$, $\mathcal{B}^{\mathcal{I}}_{map}$; the rest of the components of $\mathcal{P}^{\mathcal{I}}_{map}$ are included in the list of obstacles to be deformed into disks, $\mathcal{D}^{\mathcal{I}}_{map}$.

Note in consequence of these consolidations that the cardinality of the index $\mathcal{I}$ denoting the subset of familiar objects discovered and localized in the semantic space, $\mathcal{P}^{\mathcal{I}}_{sem}$, will in general be larger than the cardinality of connected components in $\mathcal{P}^{\mathcal{I}}_{map}$, whose cardinality in

[6]For our numerical studies, we simply assume an idealized sensor of fixed range that can instantly recognize and localize obstacles within its range. For our hardware implementation, the semantic mapping and perception pipelines rely mostly on prior work and are briefly described in Section 7.7.

Algorithm 7.1 Derivation of the sets of obstacles $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$, used in the diffeomorphism construction, and their associated properties, from the aggregated list of known obstacles in the physical space $\tilde{\mathcal{P}}_{\mathcal{I}}$.

function MAPPEDSPACERECOVERY($\tilde{\mathcal{P}}_{\mathcal{I}}$)
  $\mathcal{P}^{\mathcal{I}}_{map} \leftarrow$ `Union`(`dilate`($\tilde{\mathcal{P}}_{\mathcal{I}}, r$))
  do
    $P \leftarrow$ `pop`($\mathcal{P}^{\mathcal{I}}_{map}$) ▷ Pop next component
    if $P \cap \partial\mathcal{F}_e \neq \varnothing$ **then**
      B.`geometry` $\leftarrow P \cap \mathcal{F}_e$
      B.`tree` $\leftarrow$ `EarClipping`(B.`geometry`)
      Find root of tree B.`root` as in Section 7.3.1
      Restructure B.`tree` around B.`root`
      for $j \in B$.`tree`.`vertices` **do** ▷ Dfns. 7.2-7.7
        B.`tree`.`vertices`(j).`append`($\mathbf{x}^*_j$)
        B.`tree`.`vertices`(j).`append`($\overline{\mathcal{Q}}_j$)
      end for
      $\mathcal{B}^{\mathcal{I}}_{map}$.`append`($B$)
    else
      D.`geometry` $\leftarrow P$
      D.`tree` $\leftarrow$ `EarClipping`(D.`geometry`)
      Find root of tree D.`root` as in Section 7.3.1
      Restructure D.`tree` around D.`root`
      for $j \in D$.`tree`.`vertices` **do** ▷ Dfns. 7.2-7.7
        D.`tree`.`vertices`(j).`append`($\mathbf{x}^*_j$)
        D.`tree`.`vertices`(j).`append`($\overline{\mathcal{Q}}_j$)
      end for
      Find $\rho = D$.`radius` as in Section 7.3.3
      $\mathcal{D}^{\mathcal{I}}_{map}$.`append`($D$)
    end if
  while $\mathcal{P}^{\mathcal{I}}_{map} \neq \varnothing$
  return $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$
end function

turn will generally be larger than that of the connected components in $\mathcal{D}^{\mathcal{I}}_{map}$. Nevertheless, under the assumption of fixed obstacles, these cardinalities are also fixed functions of $\mathcal{I}$, constant over some fixed subset of robot placements, hence these subsets of semantically identified and localized familiar obstacles, $\mathcal{I} \subseteq \mathcal{N}_{\mathcal{P}}$, comprise the appropriate indices for the modes (i.e., they label the vertices of the graph Γ) of the hybrid system just analyzed in Section 7.4.1. This situation is illustrated in Section 7.8.

All these computational steps rely on underlying polygon operations (unions, intersections, differences); the development of such algorithms has been heavily explored in the computational geometry literature [41, 53, 54], and here we rely on their efficient implementations, either in the open-source C++ Boost library [170], or in the open-source Shapely package [174] in Python.

The next step is to triangulate every obstacle P_i in both $\mathcal{D}^{\mathcal{I}}_{map}$ and $\mathcal{B}^{\mathcal{I}}_{map}$ using the Ear Clipping Method, find its root triangle r_i and extract the corresponding tree of triangles $\mathcal{T}_{P_i} := (\mathcal{V}_{P_i}, \mathcal{E}_{P_i})$, as described in Section 7.3.1. For the implementation of the Ear Clipping Method, we use either the open-source Boost library [170], for our C++ implementation, or the open-source `tripy` package [193], for our Python implementation.

The final operation of the mapped space recovery algorithm is to extract the admissible centers of transformation, $\mathbf{x}^*_j$, according to Definitions 7.2 - 7.7, the corresponding radius of transformation, ρ_i (if $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$), and the admissible polygonal collars, $\overline{\mathcal{Q}}_j$ for all triangles $j \in \mathcal{V}_{P_i}$ and polygons P_i in $\mathcal{D}^{\mathcal{I}}_{map}$ and $\mathcal{B}^{\mathcal{I}}_{map}$. There is not a unique method of performing this operation, and we provide our implemented method along with other details in Appendix A.2. The mapped space recovery algorithm is summarized in Algorithm 7.1.

7.5.2 Reactive Planning Component

The mapped space recovery algorithm described above just informs the robot about its surroundings, by post-processing aggregated information from the semantic mapping pipeline. In this Section, we describe the algorithm for generating actual robot inputs, that closes our control loop.

Given the robot state in the mapped space, ($\mathbf{x}$ for a fully actuated robot or $\overline{\mathbf{x}}$ for a dif-

Algorithm 7.2 Description of the online reactive planning module that uses the state of the robot, LIDAR input, and $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$.

function REACTIVEPLANNING(`State`, `LIDAR`, $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$)
  if `RobotType` is `FullyActuated` **then**
    $\mathbf{x} \leftarrow$ `State`
    $\mathbf{y} \leftarrow \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ ▷ Sec. 7.3, Appendix B.2
    Compute $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}$ ▷ Appendix B.2
    Populate $\mathcal{F}^{\mathcal{I}}_{model}$ using `LIDAR`, $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$
    Construct $\mathcal{LF}(\mathbf{y})$ ▷ (7.39)
    Compute input $\mathbf{v}^{\mathcal{I}}$ ▷ (7.38)
    Compute input $\mathbf{u}^{\mathcal{I}}$ ▷ (7.61)
    `RobotInput` $\leftarrow \mathbf{u}^{\mathcal{I}}$
  else if `RobotType` is `DiffDrive` **then**
    $\overline{\mathbf{x}} \leftarrow$ `State`
    $\overline{\mathbf{y}} \leftarrow \overline{\mathbf{h}}^{\mathcal{I}}(\overline{\mathbf{x}})$ ▷ Sec. 7.4.2, (7.42), Appendix B.2
    Compute $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}, \frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{ij}}{\partial [\mathbf{x}]_k}$ ▷ Appendix B.2
    Populate $\mathcal{F}^{\mathcal{I}}_{model}$ using `LIDAR`, $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$
    Construct $\mathcal{LF}(\mathbf{y})$ ▷ (7.39)
    Compute inputs $\overline{\mathbf{v}}^{\mathcal{I}}$ ▷ (7.55)
    Compute input $\overline{\mathbf{u}}^{\mathcal{I}}$ ▷ (7.54) using (7.63),(7.64)
    `RobotInput` $\leftarrow \overline{\mathbf{u}}^{\mathcal{I}}$
  end if
  return `RobotInput`
end function

ferential drive robot), and the list of obstacles in the mapped space, $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$, along with their associated triangulation trees and their properties as computed with Algorithm 7.1, the first step of the reactive planning algorithm is to compute the state of the robot in the model space ($\mathbf{h}^{\mathcal{I}}(\mathbf{x})$ for a fully actuated robot (7.29) or $\overline{\mathbf{h}}^{\mathcal{I}}(\overline{\mathbf{x}})$ for a differential drive robot (7.42)), the diffeomorphism jacobian $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}$, and partial derivatives of the terms of the jacobian $\frac{\partial [D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}]_{ij}}{\partial [\mathbf{x}]_k}$ (needed in (7.54) for a differential drive robot), following the methods outlined in Section 7.3. We show in Appendix B.2 how to perform this operation inductively, given the general form of $\mathbf{h}^{\mathcal{I}}$ in (7.29).

Next, we need to properly populate the model space with obstacles, in order to compute the input (7.61) for a fully actuated robot, or the inputs (7.54) (using (7.63),(7.64)) for a differential drive robot. This procedure is straightforward for familiar obstacles; obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ are not taken into account in the model space, since they are merged into the boundary $\partial\mathcal{F}_e$, and obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ are represented in the model space as disks with radius ρ_i centered at $\mathbf{x}^*_i$, with i spanning the elements of $\mathcal{D}^{\mathcal{I}}_{map}$. For unknown obstacles in $\mathcal{C}_{map}$, we use the LIDAR measurements (see Fig. 7.2). Namely, we first pre-process the 2D LIDAR pointcloud by disregarding points that correspond to obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ or $\mathcal{B}^{\mathcal{I}}_{map}$, since those have already been considered. The pointcloud with the remaining points is then transferred with an identity transform to the model space; this is allowed because, by construction, the diffeomorphism $\mathbf{h}^{\mathcal{I}}$ between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ defaults to the identity transform (i.e., $\mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \mathbf{x}$) on the boundary of any unknown obstacle, provided that this obstacle is sufficienty separated from any obstacle in $\mathcal{P}^{\mathcal{I}}_{map}$ (see Assumption 7.3).

With the ("virtual") model space constructed, we can then construct the local freespace (7.39), as in [7, Eqn. (24)], and, subsequently, compute the input $\mathbf{v}^{\mathcal{I}}$ (7.38) for a fully actuated robot or the inputs $\overline{\mathbf{v}}^{\mathcal{I}}$ (7.55) for a differential drive robot. The final step is to compute the "pull-backs" of these inputs in the physical space and enforce bounds, by using (7.61) for a fully actuated robot, or (7.54) along with (7.63), (7.64) to adaptively modify the input gains for a differential drive robot. The reactive planning module functionality is summarized in Algorithm 7.2.

It must be highlighted that the presented reactive planning pipeline (summarized in Fig. 7.2) runs at 10Hz online and onboard our physical robots' Nvidia Jetson TX2 modules, during execution time.

7.6 Numerical Results

In this Section, we present numerical simulations that illustrate our formal results. Our simulations are run in MATLAB using `ode45`, and $p = 20$ for the R-function construction, as described in Appendix A.1. Our mapped space recovery (Section 7.5.1) and reactive planning (Section 7.5.2) algorithms are implemented in Python and communicate with MATLAB using the standard MATLAB-Python interface. For our numerical results, we assume perfect robot state estimation and localization of obstacles, using a fixed range sensor that can instantly identify and localize either the entirety of familiar obstacles that intersect its footprint, or the corresponding fragments of unknown obstacles within its range.

7.6.1 Comparison with Original Doubly Reactive Algorithm

We begin with a comparison of our algorithm performance with the original version of the doubly reactive algorithm in [7], that we use in the model space computed at each instant from the perceptual inputs as depicted in Fig. 7.2-(e) and described in Section 7.5.2. Fig. 7.7 demonstrates the well understood limitations of this algorithm (limitations of all online [28] or offline [59] reactive schemes we are aware of). Namely, in the presence of a flat surface or a non-convex obstacle, or when separation assumptions are violated, the robot gets stuck in undesired local minima. In contrast, our algorithm overcomes this limitation, by recourse to the robot's ability to recognize obstacles at hand (documented empirically in Section 7.8) and transform them appropriately (as detailed in Section 7.3) for both a fully actuated and a differential drive robot. The robot radius used in our simulation studies is 0.2m, the control gains are $k = k_v = k_\omega = 0.4$, and the values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in the diffeomorphism construction are 4.0, 0.05 and 2.0 respectively. Finally, the maximum input $u_{\max}$ for the fully actuated robot as well as the maximum linear and angular inputs $v_{\max}, \omega_{\max}$ for the differential drive robot are limited to 0.4.

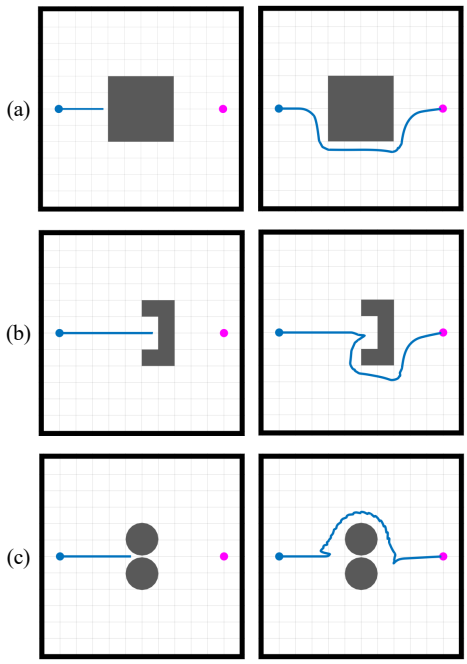

Figure 7.7: Comparison with original doubly reactive algorithm for a fully actuated robot (blue) navigating towards a goal (purple). (a) Convex obstacle with flat surfaces, (b) Non-convex obstacle, (c) Convex obstacles violating the separation assumptions of [7]. Left column: Original doubly reactive algorithm [7], Right column: Our algorithm.

7.6.2 Navigation in a Cluttered Environment with Obstacle Merging

For the next set of numerical studies, we focus on environments cluttered with several instances of the same familiar obstacle, in different, à-priori unknown poses. We illustrate the concept in Fig. 7.8. The robot abstracts away the familiar geometry to explore the unknown topology of the workspace online during execution time. In this particular example, the robot first adopts the hypothesis that an "opening" exists above the initially observed obstacle. With the observation and instantiation of the second obstacle in the semantic map, it is then capable of correcting this hypothesis by merging the obstacle to the boundary of $\mathcal{F}_e$. The properties of the hybrid controller presented in Section 7.4 guarantee convergence to the goal for both the fully actuated and the differential drive robot, as shown in Fig. 7.9.

We further illustrate the scope of formal results by presenting numerical simulations where the constellation of fixed obstacles incurs the need for multiple mergings between obstacles or between obstacles and the boundary of the enclosing freespace $\mathcal{F}_e$. As guaran-

teed, both the fully actuated (Theorem 7.3) and the differential drive (Theorem 7.4) robots converge to the desired goal from a variety of initial conditions (all but a set of measure zero must converge), as shown in Fig. 7.10. The robot radius used in our simulations is 0.25m, the control gains are $k = k_v = k_\omega = 0.4$, and the values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in the diffeomorphism construction are 2.0, 0.05 and 1.0 respectively. The maximum input $u_{\max}$ for the fully actuated robot as well as the maximum linear and angular inputs $v_{\max}, \omega_{\max}$ for the differential drive robot are limited to 0.4.

7.6.3 Navigation Among Mixed Known and Unknown Obstacles

Finally, Fig. 7.11 illustrates the convergence guarantees for both a fully actuated as well as a differential drive robot when confronted both by familiar obstacles (with à-priori unknown pose) as well as completely unknown obstacles (presumed to satisfy the convexity and separation assumptions of [7]), as outlined in Section 7.1. The robot radius used in our simulations is 0.25m, the control gains are $k = k_v = k_\omega = 0.4$, and the values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in the diffeomorphism construction are 1.6, 0.05 and 0.8 respectively. The maximum input $u_{\max}$ for the fully actuated robot as well as the maximum linear and angular inputs $v_{\max}, \omega_{\max}$ for the differential drive robot are limited to 0.4.

7.7 Experimental Setup

Because the reactive planners introduced in this Section take the form of first order vector fields (i.e., issuing velocity commands at each state), we use a quasi-static platform, the Turtlebot robot [194], for the bulk of physical experiments reported next. With the aim of merely suggesting the robustness of these feedback controllers, we also repeat two of those experiments using the more dynamic Minitaur robot (Section 3.1), whose rough approximation to the quasi-static differential drive motion model as reported in Section 3.1.3 is adequate to yield nearly indistinguishable navigation behavior.

The experimental setups for our robots are depicted in Fig. 7.12. In both cases, the main computer is an Nvidia TX2 GPU unit [139], responsible for running our mapped space recovery and reactive planning algorithms online, during execution time, according to

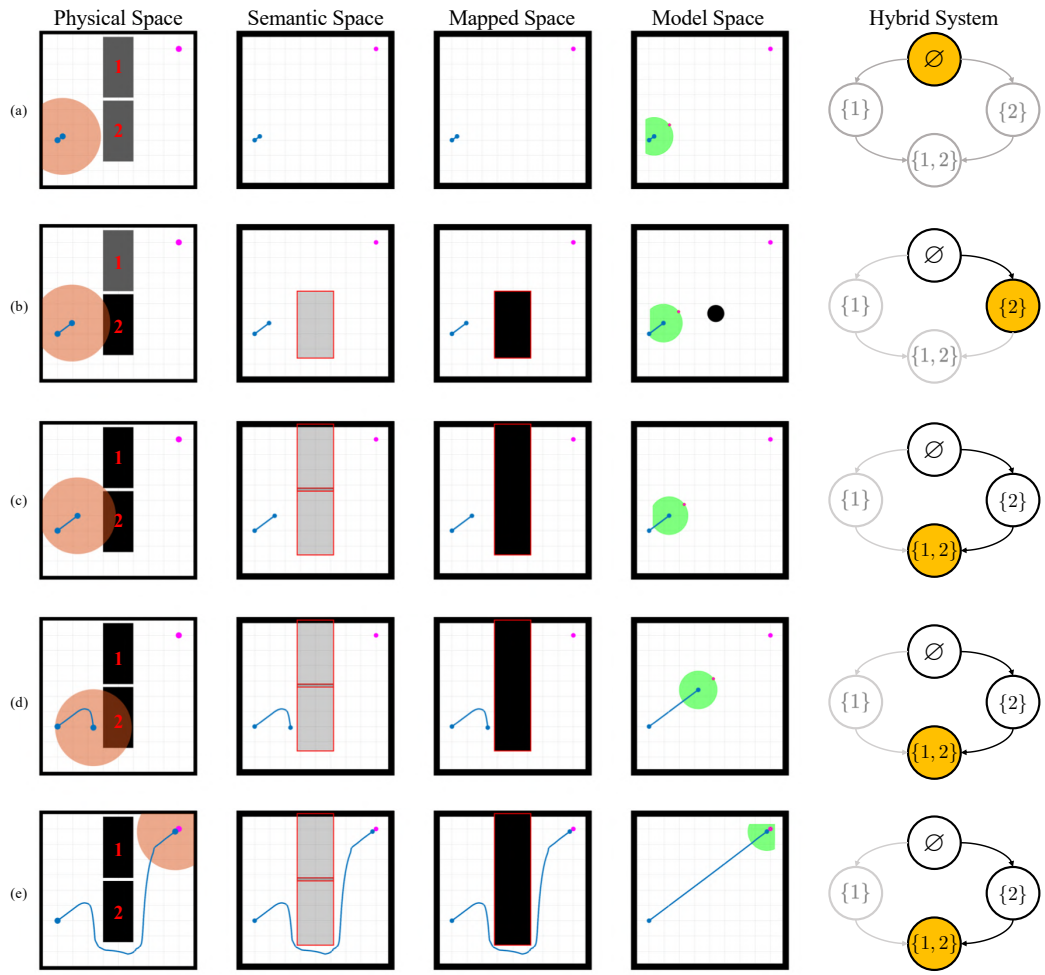

Figure 7.8: Illustration of the algorithm with successive snapshots of a single simulation run in the presence of two familiar obstacles with à-priori unknown pose. (a) The robot starts navigating towards the goal with no prior information about its environment. The initial mode of the hybrid controller is $\mathcal{I} = \varnothing$. (b) The robot discovers the first familiar obstacle (labeled 2 as shown in the physical space), driving the hybrid dynamical system (Section 7.4) into mode $\mathcal{I} = \{2\}$, wherein it makes an (incorrect) hypothesis about the topological state of the workspace (shown in the mapped space). The robot now computes according to [7] the model control input in the topological model space (shown in the fourth column). (c) The robot discovers the second familiar obstacle (labeled 1 in the physical space), driving the hybrid dynamical system into the terminal (Definition 7.8) mode $\mathcal{I} = \{1, 2\}$, wherein it corrects the initial hypothesis by merging the union of the two obstacles to the boundary. (d) The reactive field pushing the robot along a direct path to the goal in the unobstructed model space is deformed to generate a sharp correction of course in the geometrically accurate mapped space until, finally, (e) safely navigates to the goal. The deformation of space that aligns the geometrically informed mapped space with its topologically equivalent model space can be visualized by comparing the direct path to the goal the planner generates in the model space with its diffeomorphic image, the curved path connecting the robot's starting point to the goal in the mapped space. Note that the robot has no prior information about the structure of the hybrid system (depicted in the right-most column with unexplored modes in grey): it is driven around the hybrid graph, Γ, by its online perceptual experiences as it accumulates more information about its surroundings. Note, as well, that the cardinality of topological obstacles (the number of punctures in the model space) is independent of the number of semantically localized objects, $|\mathcal{I}|$.

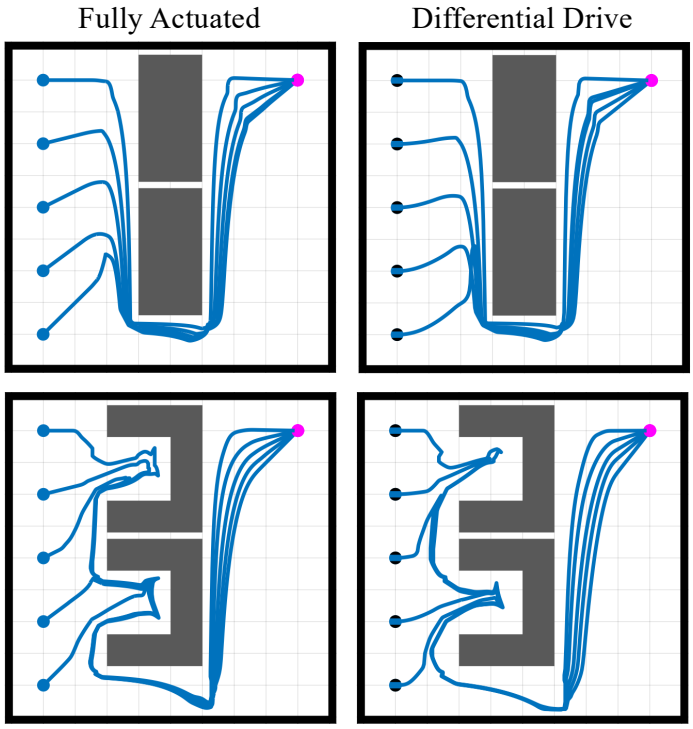

Figure 7.9: Numerically simulated illustrations of the navigation planner's behavior from multiple initial conditions for both a fully actuated and a differential drive robot, in the presence of two familiar obstacles with à-priori completely unknown placement in the workspace. Top: Obstacles with rectangular shape, Bottom: U-shaped obstacles. The hybrid systems theorems presented in Section 7.4 guarantee the robot will safely navigate to the goal with no collisions along the way.

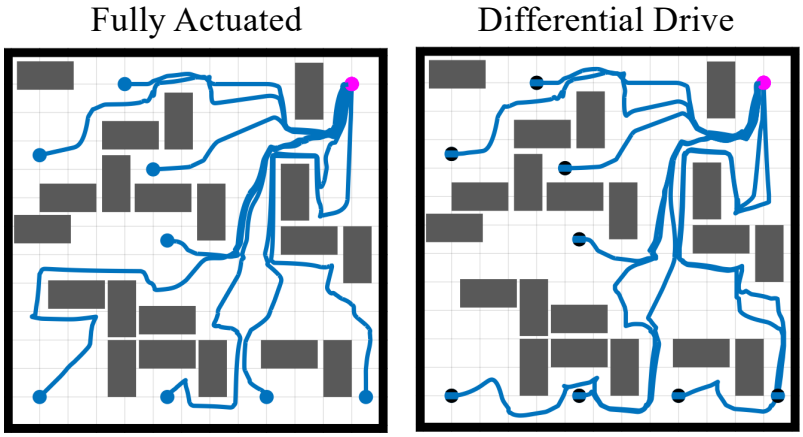

Figure 7.10: Simulated trajectories from multiple initial conditions for both a fully actuated and a differential drive robot, in the presence of many instances of the same familiar obstacle with à-priori unknown pose. The robot explores the geometry and topology of the workspace online during execution time, and the guarantees of the hybrid controller in Section 7.4 allow it to safely navigate to the goal, without converging to local minima arising from the complicated geometry of the workspace.

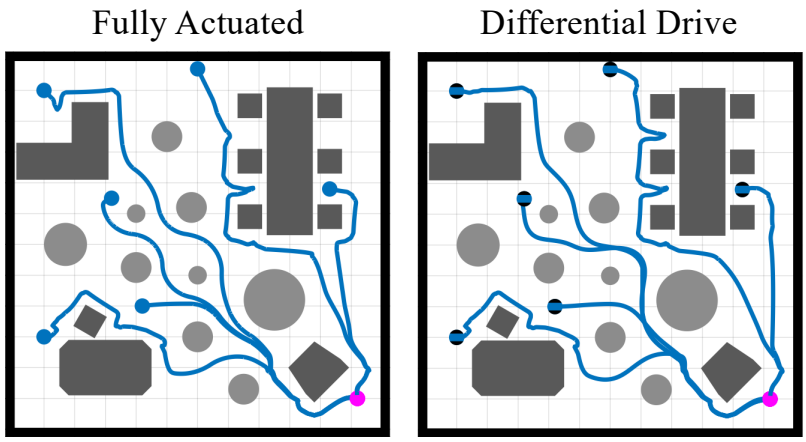

Figure 7.11: Simulated trajectories from multiple initial conditions for both a fully actuated robot and a differential drive robot, in the presence of both familiar obstacles with à-priori unknown pose (dark grey) and completely unknown obstacles (light grey). The guarantees of the hybrid controller in Section 7.4 allow the robot to always safely navigate to the goal.

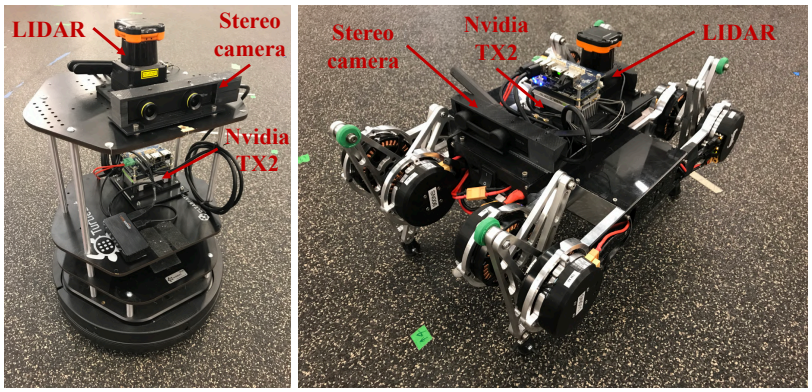

Figure 7.12: The platforms used in our experiments: (Left) Turtlebot, (Right) Minitaur, equipped with a Hokuyo LIDAR for avoidance of unknown obstacles, a stereo camera for object recognition and visual odometry, and an NVIDIA TX2 GPU module as the main onboard computer.

Fig. 7.2. The GPU unit communicates with a Hokuyo LIDAR [79], used to detect unknown obstacles, and a ZED Mini stereo camera [183], used for visual-inertial state estimation and for detecting familiar obstacles. As shown in Fig. 7.2, we choose to run our perception and semantic mapping pipelines described next either *onboard* (using the same Nvidia TX2 GPU unit) or *offboard* (on a desktop computer with an Nvidia GeForce RTX 2080 GPU), for faster inference and improved performance. We also assume that the differential drive robot model, presented in (7.34), is the most suitable motion model for both robots. This is indeed the case for Turtlebot, and we refer the reader to Section 3.1.3 for an extensive discussion on the empirical anchoring [61] of the unicycle template on Minitaur.

Since the main focus of this work is not the development of new perception or state estimation algorithms, but rather the development of a provably correct planning architecture for partially known environments, we rely to as great an extent as possible on off-the-shelf perception algorithms, implemented in ROS [155], and couple them with our motion planner for the hardware experiments. We are further motivated by the intent for our accompanying software to be modular and easily integrated to existing perception pipelines for future users. We briefly describe the perception and semantic mapping algorithms employed in the Sections below, and refer the reader once more to the summary illustration of the whole navigation stack in Fig. 7.2.

7.7.1 Object Detection and Keypoint Localization

The pipeline we use to detect the objects in the scene and extract the geometric properties needed in order to estimate their 3D pose relies on [148]. The two components involved in this procedure are:

- Object detection, which returns 2D bounding boxes for each object.
- Keypoint localization, which estimates the 2D locations for a set of predefined keypoints for the specific object instance and class.

The algorithm is described in detail in [148], but here we give a brief overview of each step in Sections 7.7.1 and 7.7.1, and provide training details for our neural networks in Section 7.7.1.

Object Detection

For the task of object detection, we only require the estimation of a 2D bounding box for each object that is visible on the image. We use the YOLOv3 detector [158] which offers a good trade-off between detection accuracy and inference speed. Given a single RGB image as input, the output of the detector is a 2D bounding box for each object instance, along with the estimated class for this bounding box.

Keypoint Localization

For the keypoint localization task, we use a Convolutional Neural Network to accurately estimate the 2D location of the keypoints within the object's bounding box. The keypoints are defined on the 3D model of the object and are selected in advance for each object instance. The keypoint localization network uses as input an RGB image of a specific object, which is cropped using the bounding box information from the detection step. The output of the network is a set of 2D heatmaps. Assuming we select k keypoints for an object, each heatmap is responsible for the localization of the corresponding keypoint. To estimate the locations $\mathbf{W} \in \mathbb{R}^{2\times k}$ of the k keypoints on the image, we use the 2D heatmap location with the maximum activation for each heatmap as the detected location for the corresponding keypoint. We also consider the value of the activation at this location as the detection confidence d_i for keypoint i. The architecture for this network follows the Stacked Hourglass design [137]. In practice, we train a single network for all objects of interest and at test time we use only the heatmaps for the specific class, which is already known from the detection step.

Training Details

The aforementioned neural networks are trained to detect a predefined set of object instances visualized in Figure 7.13. The object classes represented for our experiments are chair, table, ladder, cart, gascan and pelican case. Our goal is to include a variety of instances in terms of the size, shape and visual appearance, in an attempt to simulate the variety of objects that can be encountered in a partially familiar environment. The training data for the particular instances of interest are collected with a semi-automatic procedure, similarly to [148]. Given the bounding box and keypoint annotations for each image, the two networks were trained with their default configurations until convergence.

7.7.2 Semantic Mapping

Our semantic mapping infrastructure relies on the algorithm presented in [30], and implemented in C++ using GTSAM [50] and its iSAM2 implementation [90] as the optimization

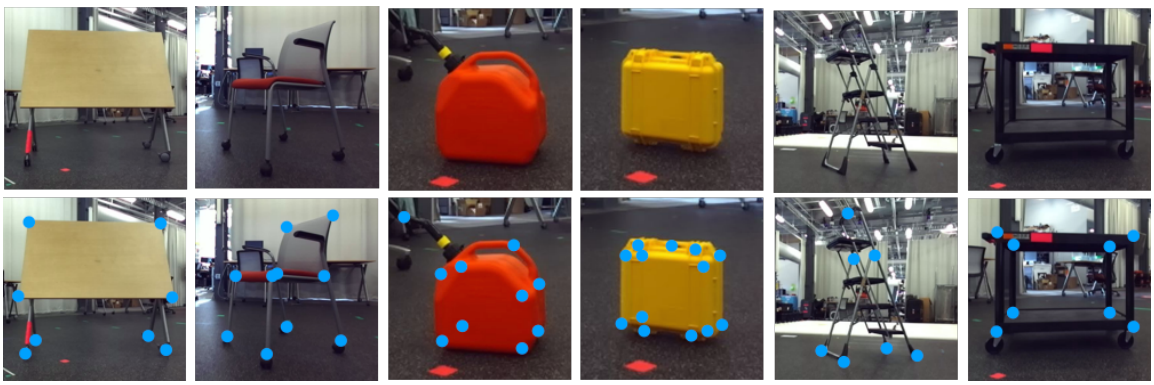

Figure 7.13: Top row: Objects used in our experimental setup: table, chair, gascan, pelican case, ladder, cart. Bottom row: Visualization of the semantic keypoints for each object class.

back-end. Briefly, this algorithm fuses inertial information (here simply provided by the position tracking implementation from StereoLabs on the ZED Mini stereo camera [182]), and semantic information (i.e., the detected keypoints and the associated object labels as described in Section 7.7.1) to provide a posterior estimate for both the robot state and the associated poses for all tracked objects, by simultaneously solving the data association problem arising when several objects of the same class exist in the map. As described in [30], except for providing an estimate for all poses tracked in the environment, this algorithm facilitates loop closure recognition based on viewpoint-independent semantic information (i.e., tracked objects), rather than low-level geometric features such as points, lines, or planes.

For a single frame detection, the 3D pose of each object with respect to the camera is recovered using the estimated 2D locations of the associated object keypoints. By denoting with $\mathbf{S} \in \mathbb{R}^{3\times k}$ the 3D locations of the keypoints in the canonical pose of an object instance with k keypoints, the goal is to estimate the rotation $\mathbf{R} \in \mathbb{R}^{3\times 3}$ and translation $\mathbf{T} \in \mathbb{R}^{3\times 1}$ of the object, such that the distance of the projected 3D keypoints from their corresponding detected 2D locations is minimized. To incorporate the detection confidence for each keypoint in the optimization, we define the matrix $\mathbf{D} \in \mathbb{R}^{k\times k}$. This is a diagonal matrix that features the detection confidences d_i for each keypoint i in its diagonal. The optimization

problem is then formulated as:

$$\min_{\mathbf{R},\mathbf{T}} \quad \frac{1}{2} \left\| (\tilde{\mathbf{W}}\mathbf{Z} - \mathbf{R}\mathbf{S} - \mathbf{T}\mathbf{I}^{\top})\mathbf{D}^{\frac{1}{2}} \right\|_F^2, \tag{7.65}$$

where $\tilde{\mathbf{W}} \in \mathbb{R}^{3\times k}$ represents the normalized homogeneous coordinates of the 2D keypoints and $\mathbf{Z} \in \mathbb{R}^{k\times k}$ is a diagonal matrix, that features the depths z_i for each keypoint i in its diagonal.

After the estimation of the object's 3D pose from a *single frame measurement* as described above, the 3D positions of its corresponding semantic keypoints are then independently tracked and the object's pose is appropriately updated, as more frame measurements are added. Once a sufficient number of frame measurements[7] has been incorporated so that the 3D keypoint positions can be triangulated, the object is considered to be *localized* and is permanently added to the map. The reader is referred to [30] for more details. Fig. 7.14 shows an example of this localization process. It should be noted that for our onboard implementation, where inference using the object detection and keypoint estimation neural networks is slower, we include in the semantic map both the localized objects, after several frame measurements, and objects resulting from a single frame measurement pose estimation, to allow for faster response to sensory input.

As shown in Fig. 7.2, the meshes of the objects in the semantic map, defined by the corresponding keypoint adjacency properties and the extracted 3D pose, are projected on the robot's plane of motion to provide the aggregated list of known obstacles in the physical space $\tilde{\mathcal{P}}_{\mathcal{I}}$, forwarded to our mapped space recovery module (described in Section 7.5.1). On the other hand, the posterior estimate of the robot pose on the plane, extracted by the semantic mapping module, is forwarded to our reactive planning module (described in Section 7.5.2).

[7]This number depends on the needed camera motion between successive measurements, in order to establish a good baseline for triangulation [73]; in this work, we found that 5 measurements for the offboard experiments and 3 measurements for the onboard experiments yielded reasonably fast keypoint localization.

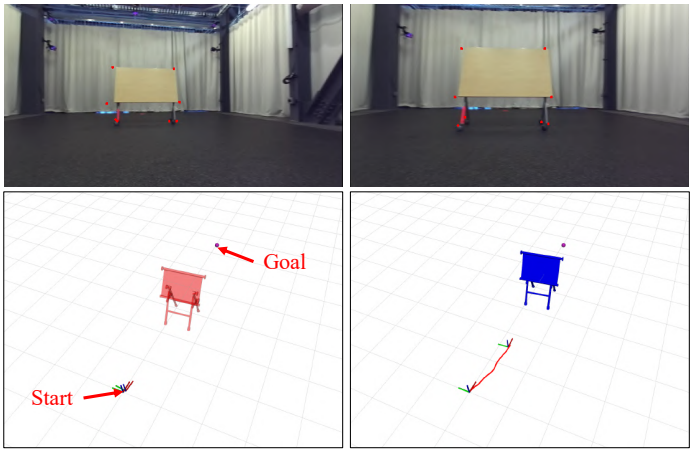

Figure 7.14: Illustration of the object localization process using the semantic mapping pipeline from [30]. Left: The robot starts navigating toward its goal and discovers a familiar obstacle (table). The obstacle is temporarily included in the semantic map, after its 3D pose is estimated using a single frame measurement (7.65) (red). Right: Once a sufficient number of frame measurements has been incorporated and the 3D pose has been accordingly updated, the object is permanently localized and included in the semantic map (blue).

7.8 Experimental Results

In this Section, we provide our experimental results using both the Turtlebot and the Minitaur robot, and the setup described in Section 7.7. We begin with experiments run using Turtlebot and offboard (Section 7.8.1) or onboard (Section 7.8.2) perception, and continue with Minitaur experiments using offboard perception (Section 7.8.3), to demonstrate the robustness of our method on a more dynamic legged platform. It should be noted that although the perception algorithms, described in Section 7.7, are run either offboard or onboard, our mapped space recovery and reactive planning modules, described in Algorithms 7.1 and 7.2 respectively, are always run onboard each robot's Nvidia TX2 module. The control gains used in our experiments are $k_v = k_\omega = 0.4$, and the maximum linear and angular inputs $v_{\max}, \omega_{\max}$ are set to 0.4.

Comparison with Original Doubly Reactive Algorithm

In this Section, we demonstrate experiments similar to the simulations reported in Section 7.6.1. We first illustrate various well understood failures of the original version of the doubly reactive algorithm in [7]. Collisions result from the presence of short obstacles that

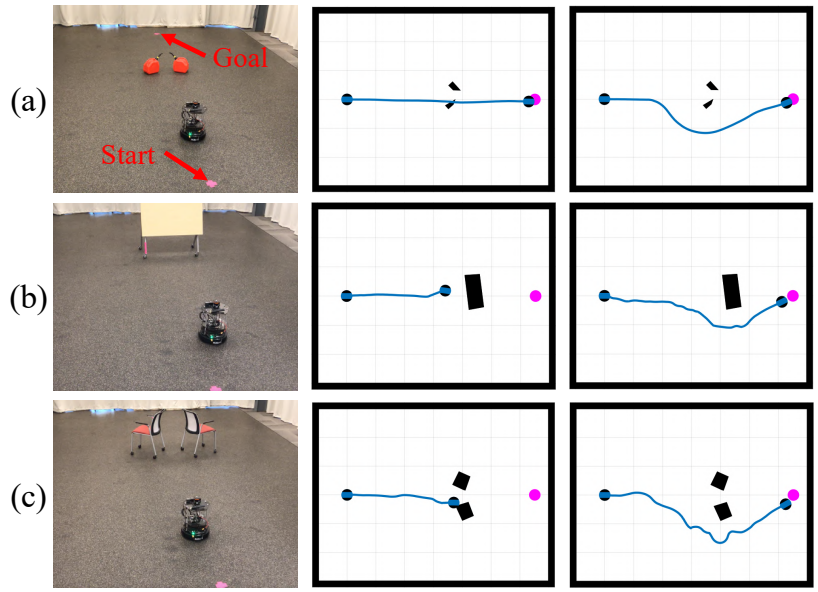

Figure 7.15: Physical experiments akin to the numerical simulations depicted in Fig. 7.7, comparing the original doubly reactive algorithm [7] (middle column) with our algorithm (right column) in different physical settings (left column), using Turtlebot and offboard perception. (a) Two gascans forming a non-convex trap, (b) Table used as a flat obstacle, (c) Two chairs violating the separation assumptions of [7].

cannot be detected by the 2D LIDAR (Fig. 7.15-(a)). Confronted by obstacles with flat surfaces (Fig. 7.15-(b)), or when separation assumptions are violated (Fig. 7.15-(c)), the original algorithm gets stuck in undesired local minima (Fig. 7.15-(b),(c)). In contrast, our new algorithm guarantees safe convergence to the goal in all these cases: short but familiar obstacles (in this case the gascan in column 3 of Fig. 7.13) are recognized by the camera system and localized; once localized, these known geometries can then be appropriately abstracted into the model space (Section 7.3) which is topologically equivalent but geometrically simplified to meet the requirements of [7]. Fig. 7.15 shows the groundtruth trajectory of the robot, recorded using Vicon, along with 2D projections on the horizontal plane of the obstacles' keypoint meshes, that were used for the construction of the semantic space (Section 7.2.2). The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in the diffeomorphism construction are 4.0, 0.05 and 2.0 respectively.

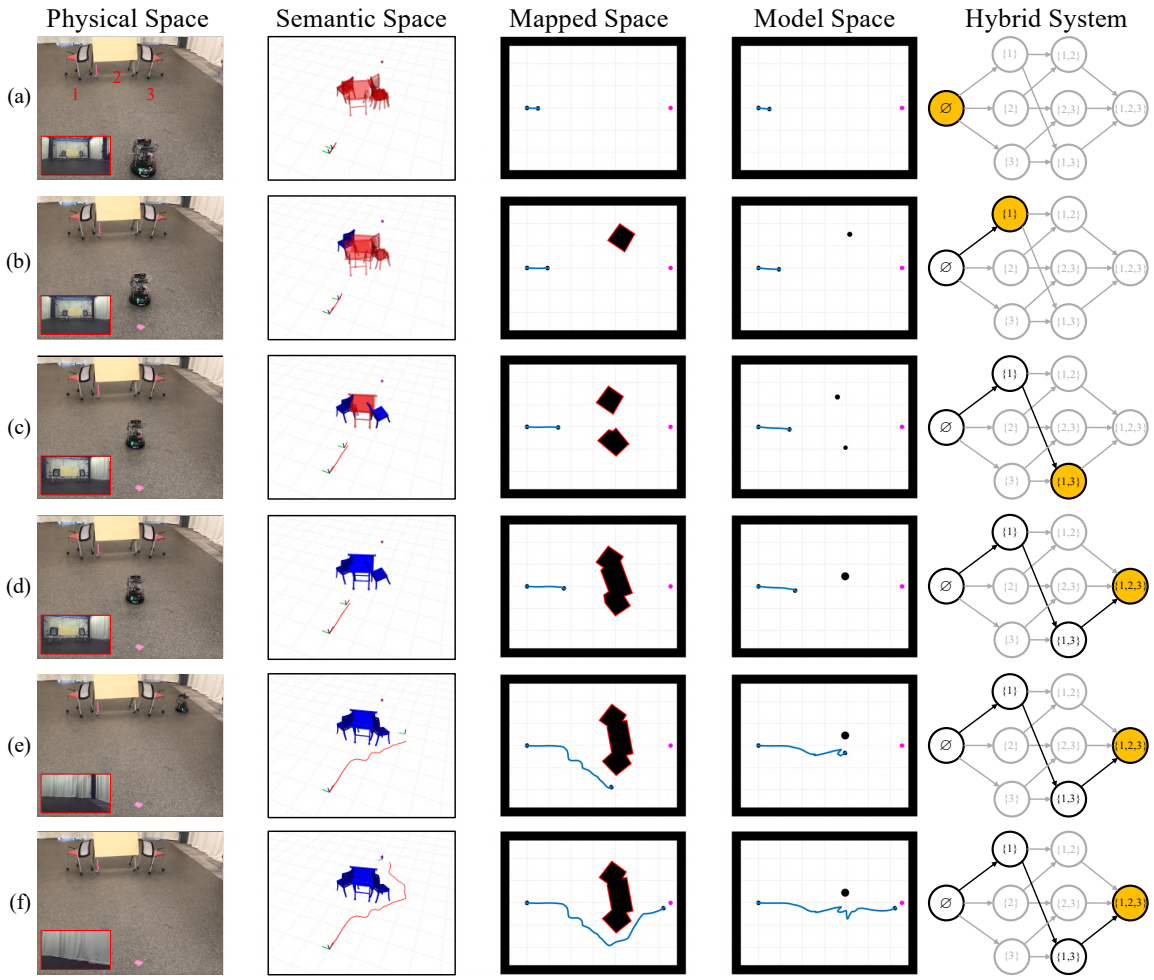

Figure 7.16: Illustration of the empirically implemented complete navigation scheme (akin to the numerical simulation depicted in Fig. 7.8) in a physical setting where three familiar obstacles (two chairs and a table) form a non-convex trap. (a) The robot starts navigating toward its designated target in a previously unknown environment, and detects familiar obstacles. The initial mode of the hybrid system is $\mathcal{I} = \varnothing$. (b)-(d) The robot keeps localizing familiar obstacles, and changes its belief about the topological state of the workspace (as evident in the column showing the corresponding model space). (e) Using the information in the semantic space and now being in the terminal (Definition 7.8) mode $\mathcal{I} = \{1, 2, 3\}$, wherein it has encountered and localized all the environment's familiar obstacles, the robot is driven by the mapped space transformation (Section 7.3) of the model space vector field [7] to avoid the obstacles, until (f) it converges to the designated goal as guaranteed by the results of Section 7.4. The right column shows how the robot experiences transitions in the (previously unknown) hybrid system (modes that are never experienced are shown in grey).

7.8.1 Experiments with Turtlebot and Offboard Perception

Navigation in a Cluttered Environment with Obstacle Merging

We begin the second set of experiments by demonstrating the merging process and the properties of the hybrid controller, reported in Section 7.4, in a physical setting. As shown in Fig. 7.16, the robot starts navigating toward its target and localizing obstacles in front of it, until it converges to its target; at the same time, by incorporating more information in its semantic map, it experiences transitions to different modes of the (previously unknown) hybrid system. The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in this experiment are 4.0, 0.05 and 2.0 respectively.

Finally, Fig. 7.17 demonstrates navigation in environments cluttered with multiple familiar obstacles. In the first illustration, the robot reactively chooses to navigate through a gap between the gascan and a chair. Despite the blockage of this gap by another familiar obstacle (pelican case) in the second illustration, the robot reactively chooses to follow another safe and convergent trajectory (as guaranteed by the theorems of Section 7.4), by merging the set gascan - pelican case - chair, and considering them as a single obstacle. The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in this experiment are 1.6, 0.05 and 0.8 respectively.

Navigation Among Mixed Known and Unknown Obstacles

In the next set of experiments, we consider navigation among multiple familiar and unknown obstacles. Fig. 7.18 shows that the robot safely converges to the goal from multiple initial conditions, using vision and the setup described in Section 7.7 for familiar obstacle detection and localization, and the onboard 2D LIDAR for all the unknown obstacles. In Fig. 7.18, we also overlay trajectories from a MATLAB simulation of a differential-drive robot with the same initial conditions and similar control gains; the simulated and physical platform follow similar trajectories in all three cases. The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in these experiments are 2.0, 0.05 and 1.0.

It should be highlighted that even when the object localization process fails, collision

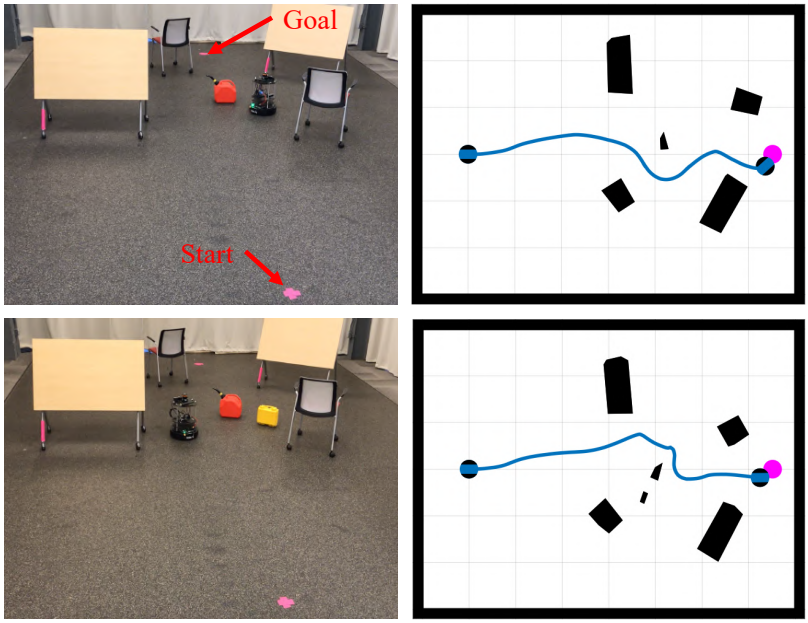

Figure 7.17: Navigation among multiple familiar obstacles, using Turtlebot and offboard perception. Top: The robot exploits the gap between the gascan and the chair to safely navigate to the goal. Bottom: When we block this gap by another familiar obstacle (pelican case), the robot reactively chooses to follow another safe and convergent trajectory, by consolidating the semantic triad {gascan, pelican case, chair} into a single, "mapped" obstacle in $\mathcal{D}^{\mathcal{I}}_{map}$.

avoidance is still guaranteed with the use of the onboard LIDAR. Nevertheless, collisions could result with obstacles that cannot be detected by the 2D horizontal LIDAR (e.g., see Fig. 7.15-(a)). One could still think of extensions to the presented sensory infrastructure (e.g., the use of a 3D LIDAR) that could still guarantee safety under such circumstances.

7.8.2 Experiments with Turtlebot and Onboard Perception

This Section briefly reports on experiments using onboard perception. As described in Section 7.7.2, here we use both the localized obstacles by the semantic mapping pipeline and raw, not permanently localized obstacles, resulting from a single semantic keypoint frame measurement and the optimization problem given in (7.65). Fig. 7.19 illustrates an example; the robot detects and avoids the two chairs in front of it, even if they are only temporarily included in the semantic map (in the absence of more frame measurements). The robot then proceeds to localize and avoid the gascan and the two tables and safely converge to the designated goal. The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in

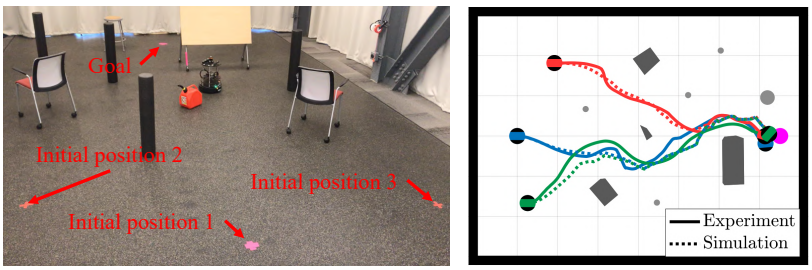

Figure 7.18: Navigation among familiar and unknown obstacles, using Turtlebot and offboard perception, from three different initial conditions. Left: A snapshot of the physical workspace. Right: A "bird's-eye" view of the workspace, with 2D projections of the localized familiar obstacles (dark grey) and unknown obstacles (light grey - groundtruth locations recorded using Vicon), along with groundtruth trajectories from the physical experiments and overlaid numerical simulations in MATLAB.

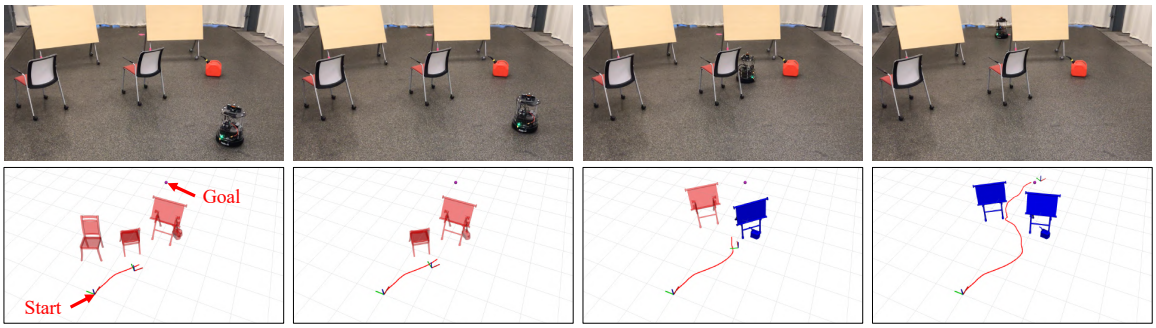

Figure 7.19: Navigation among familiar obstacles, using Turtlebot and onboard perception. Top: snapshots of the physical workspace, Bottom: illustrations of the recorded semantic map and the robot's trajectory in RViz [155]. The robot detects and avoids the two chairs in front of it, though they are only temporarily included in the semantic map (in the absence of more frame measurements). Then it proceeds to localize and avoid the two tables and the gascan, to safely converge to the goal.

this experiment are 2.0, 0.05 and 1.0 respectively.

It should be noted that the object impermanence in the semantic map violates the formal assumptions of Theorems 7.3 and 7.4; without permanently localizing an object, the robot could get stuck in an endless loop trying to avoid obstacles that it then "forgets", in unfavorable workspace configurations (e.g., like those reported in Fig. 7.9).

7.8.3 Experiments with Minitaur

Finally, Fig. 7.20 presents illustrative snapshots of two navigation examples on the much more dynamic Minitaur platform. Despite the fact that Minitaur is an imperfect kinematic unicycle and the overall shakiness of the platform, the robot is capable of detecting and localizing familiar obstacles of interest and using that information to safely converge to the

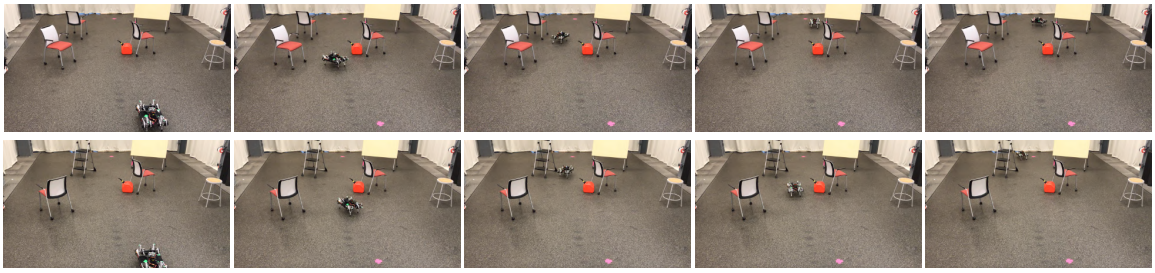

Figure 7.20: Snapshots of Minitaur avoiding multiple familiar obstacles in two different settings, using offboard perception.

target. The values of $\mu_{\gamma_{j_i}} = \mu_{\gamma_{r_i}}$, $\mu_{\delta_{j_i}} = \mu_{\delta_{r_i}}$ and $\epsilon_{j_i} = \epsilon_{r_i}$ used in this experiment are 2.0, 0.05 and 1.0 respectively.

Chapter 8

Reactive Semantic Planning in Unexplored Semantic Environments Using Deep Perceptual Feedback

This Chapter streamlines the diffeomorphism construction and extends the empirical results of Chapter 7 by incorporating a human mesh estimation algorithm, rendering our system capable of reacting and responding in real time to semantically labeled human motions and gestures. Moreover, new formal results allow tracking of suitably non-adversarial moving targets, while maintaining the same collision avoidance guarantees. We also suggest the empirical utility of the proposed control architecture with a numerical study including comparisons with a state-of-the-art dynamic replanning algorithm, and physical implementation on both a wheeled and legged platform in different settings with both geometric and semantic goals.

After stating the problem and introducing technical notation in Section 8.1, Section 8.2 describes the diffeomorphism between the mapped and model spaces, and Section 8.3 includes our main formal results. Section 8.4 and Section 8.5 continue with our numerical and experimental studies, and Section 8.6 concludes with a brief discussion of our findings. We also include pointers to open-source software implementations, for both our MATLAB

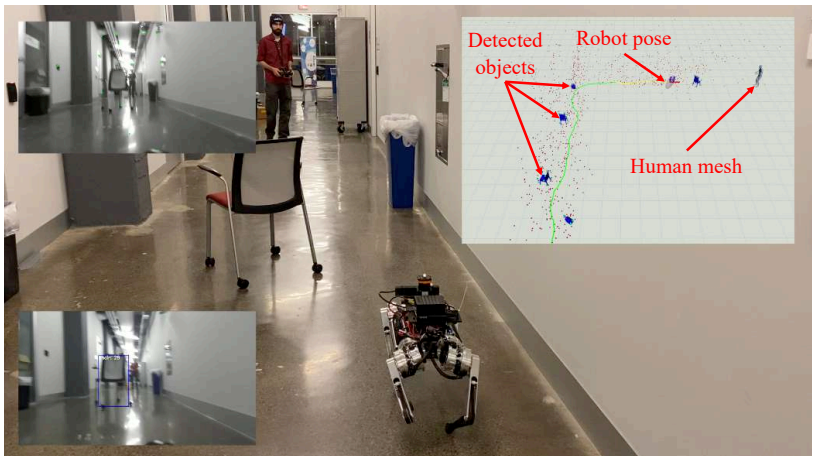

Figure 8.1: Ghost Spirit [67] following a human, while avoiding some familiar and some novel obstacles in a previously unexplored environment. Familiar obstacles are recognized and localized using visually detected semantic keypoints (bottom left inset) [148], combined with geometric features (top left inset) [30] and avoided by a local deformation of space (Fig. 8.3) that brings them within the scope of a doubly reactive navigation algorithm [9]. Novel obstacles are detected by LIDAR and assumed to be convex, thus falling within the scope of [9]. Formal guarantees are summarized in Theorems 8.1 and 8.2 of Section 8.3, and experimental settings are summarized in Fig. 8.7.

simulation package[1], and our ROS-based controller[2], in C++ and Python.

8.1 Problem Formulation and Approach

8.1.1 Problem Formulation

As in Chapters 6 - 7, we consider a robot with radius r, centered at $\mathbf{x} \in \mathbb{R}^2$, navigating a compact, polygonal, potentially non-convex workspace $\mathcal{W} \subset \mathbb{R}^2$, with known boundary $\partial\mathcal{W}$, towards a target $\mathbf{x}_d \in \mathcal{W}$. The robot is assumed to possess a sensor with fixed range R, for recognizing "familiar" objects and estimating distance to nearby obstacles[3]. We define the *enclosing workspace*, as the convex hull of the closure of the workspace $\mathcal{W}$, i.e., $\mathcal{W}_e := \left\{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \text{Conv}(\overline{\mathcal{W}})\right\}$.

The workspace is cluttered by a finite but unknown number of disjoint obstacles, denoted by $\tilde{\mathcal{O}} := \{\tilde{O}_1, \tilde{O}_2, \ldots\}$, which might also include non-convex "intrusions" of the boundary of the physical workspace $\mathcal{W}$ into $\mathcal{W}_e$. As in Chapter 7, we define the *freespace* $\mathcal{F}$ as the set of collision-free placements for the closed ball $\overline{\mathsf{B}(\mathbf{x}, r)}$ centered at $\mathbf{x}$ with radius r, and the

[1] `https://github.com/KodlabPenn/semnav_matlab`

[2] `https://github.com/KodlabPenn/semnav`

[3] As in Chapter 7, this idealized sensor is reduced to a combination of a LIDAR for distance measurements to obstacles and a monocular camera for object recognition and pose identification.

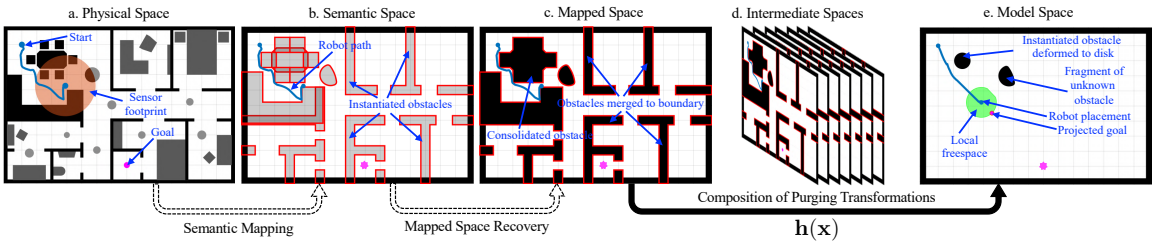

Figure 8.2: Snapshot Illustration of Key Ideas in Chapter 8, following Chapter 7: The robot moves in the physical space, in an environment with known exterior boundaries (walls), toward a goal (pink) discovering along the way (black) both familiar objects of known geometry but unknown location (dark grey) and unknown obstacles (light grey), with an onboard sensor of limited range (orange disk). As in Chapter 7, these obstacles are processed by the perceptual pipeline (Fig. 8.4) and stored permanently in the semantic space if they have familiar geometry, or temporarily, with just the corresponding sensed fragments, if they are unknown. The consolidated obstacles (formed by overlapping catalogued obstacles from the semantic space), along with the perceptually encountered components of the unknown obstacles, are again stored in the mapped space. A change of coordinates, $\mathbf{h}$, entailing an online computation greatly streamlined relative to its counterpart in Chapter 7 deforms the mapped space to yield a geometrically simple but topologically equivalent model space. This new change of coordinates defines a vector field on the model space, which is transformed in realtime through the diffeomorphism to generate the input in the physical space.

enclosing freespace, $\mathcal{F}_e$, as $\mathcal{F}_e := \left\{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \text{Conv}(\overline{\mathcal{F}})\right\}$.

Although none of the positions of any obstacles in $\tilde{\mathcal{O}}$ are à-priori known, a subset $\tilde{\mathcal{P}} := \{\tilde{P}_i\}_{i\in\mathcal{N}_\mathcal{P}} \subseteq \tilde{\mathcal{O}}$ of these obstacles, indexed by $\mathcal{N}_\mathcal{P} := \{1, \ldots, N_P\} \subset \mathbb{N}$, is assumed to be "familiar" in the sense of having a known, readily recognizable polygonal geometry, that the robot can instantly identify and localize. The remaining obstacles in $\tilde{\mathcal{C}} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{P}}$, indexed by $\mathcal{N}_\mathcal{C} := \{1, \ldots, N_C\} \subset \mathbb{N}$, are assumed to be strongly convex according to [9, Assumption 2], but are otherwise completely unknown to the robot.

To simplify the notation, we dilate each obstacle by r, and assume that the robot operates in the freespace $\mathcal{F}$. We denote the set of dilated obstacles derived from $\tilde{\mathcal{O}}, \tilde{\mathcal{P}}$ and $\tilde{\mathcal{C}}$, by $\mathcal{O}, \mathcal{P}$ and $\mathcal{C}$ respectively. Then, similarly to Chapters 6 - 7, we describe each polygonal obstacle $P_i \in \mathcal{P} \subseteq \mathcal{O}$ by an *obstacle function*, $\beta_i(\mathbf{x})$, a real-valued map providing an implicit representation of the form $P_i = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \beta_i(\mathbf{x}) \leq 0\}$ that the robot can construct online after it has localized P_i, following [176]. We also require the following separation assumptions.

Assumption 8.1. *1. Each obstacle $C_i \in \mathcal{C}$ has a positive clearance $d(C_i, C_j) > 0$ from any obstacle $C_j \in \mathcal{C}$, $j \neq i$. Also, $d(C_i, \partial\mathcal{F}) > 0$, $\forall C_i \in \mathcal{C}$.*

2. For each $P_i \in \mathcal{P}$, there exists $\varepsilon_i > 0$ such that the set $S_{\beta_i} := \{\mathbf{x} \,|\, \beta_i(\mathbf{x}) \leq \varepsilon_i\}$ has a positive clearance $d(S_{\beta_i}, C) > 0$ from any obstacle $C \in \mathcal{C}$.

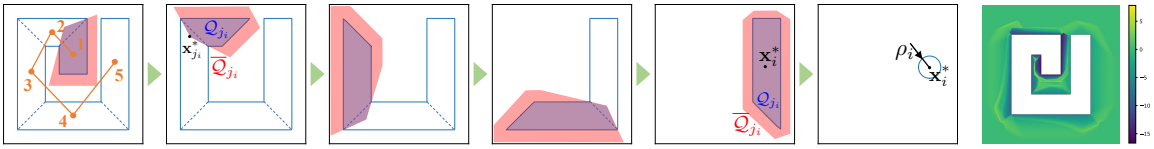

Figure 8.3: Diffeomorphism construction via direct convex decomposition: Any arbitrary convex decomposition (e.g., [68]) defines a tree $\mathcal{T}_{P_i} := (\mathcal{V}_{P_i}, \mathcal{E}_{P_i})$ (left), which induces the sequence of purging transformations that map the polygon's boundary and exterior to the boundary and exterior of an equivalent disk. The purging transformation for each convex piece $j_i \in \mathcal{V}_{P_i}$ is defined by a pair of convex polygons $\mathcal{Q}_{j_i}, \overline{\mathcal{Q}}_{j_i}$ that limit the effect of the diffeomorphism to a neighborhood of j_i. The final map is guaranteed to be smooth, as shown by a visualization of its determinant in logarithmic scale (right).

Based on these assumptions and considering first-order dynamics $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x})$, the problem consists of finding a Lipschitz continuous controller $\mathbf{u} : \mathcal{F} \to \mathbb{R}^2$, that leaves the path-connected freespace $\mathcal{F}$ positively invariant and steers the robot to the (possibly moving) goal $\mathbf{x}_d \in \mathcal{F}$.

8.1.2 Environment Representation and Technical Notation

The four distinct representations of the environment that we will refer to as *planning spaces* are shown in Fig. 8.2, and follow Section 7.2. The robot navigates the physical space and discovers obstacles, that are dilated by the robot radius r and stored in the semantic space. Potentially overlapping obstacles in the semantic space are subsequently consolidated in real time to form the mapped space. A change of coordinates from this space is then employed to construct a geometrically simplified (but topologically equivalent) model space, by merging familiar obstacles overlapping with the boundary of the enclosing freespace $\partial\mathcal{F}_e$ to $\partial\mathcal{F}_e$, deforming other familiar obstacles to disks, and leaving unknown obstacles intact.

8.2 Diffeomorphism Construction

Here, we describe our method of constructing the diffeomorphism, $\mathbf{h}^{\mathcal{I}}$, between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$. We assume that the robot has recognized and localized the $|\mathcal{J}^{\mathcal{I}}|$ obstacles in $\mathcal{P}^{\mathcal{I}}_{map}$, and has, therefore, identified obstacles to be merged to the boundary of the enclosing freespace $\partial\mathcal{F}_e$, stored in $\mathcal{B}^{\mathcal{I}}_{map}$, and obstacles to be deformed to disks, stored in $\mathcal{D}^{\mathcal{I}}_{map}$.

8.2.1 Obstacle Representation and Convex Decomposition

As a natural extension to doubly reactive algorithms for environments cluttered with convex obstacles [9, 147], we assume that the robot has access to the convex decomposition of each obstacle $P \in \mathcal{P}^{\mathcal{I}}_{map}$. For polygons without holes, we are interested in decompositions that do not introduce Steiner points (i.e., additional points except for the polygon vertices), as this guarantees the dual graph of the convex partition to be a tree. Here, we acquire this convex decomposition using Greene's method [68] and its C++ implementation in CGAL [188], operating in $\mathcal{O}(r^2n^2)$ time, with n the number of polygon vertices r the number of reflex vertices. Other algorithms [121] could be used as well, such as Keil's decomposition algorithm [98, 99], operating in $\mathcal{O}(r^2n^2 \log n)$ time.

As shown in Fig. 8.3, convex partioning results in a *tree of convex polygons* $\mathcal{T}_{P_i} := (\mathcal{V}_{P_i}, \mathcal{E}_{P_i})$ corresponding to P_i, with $\mathcal{V}_{P_i}$ a set of vertices identified with convex polygons (i.e., vertices of the dual of the formal partition) and $\mathcal{E}_{P_i}$ a set of edges encoding polygon adjacency. Therefore, we can pick any polygon as root and construct $\mathcal{T}_{P_i}$ based on the adjacency properties induced by the dual graph of the decomposition, as shown in Fig. 8.3. If $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, we pick as root the polygon with the largest surface area, whereas if $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, we pick as root any polygon adjacent to $\partial\mathcal{F}_e$.

8.2.2 The Map Between the Mapped and the Model Space

As shown in Fig. 8.3, the map $\mathbf{h}^{\mathcal{I}}$ between the mapped and the model space is constructed in several steps, involving the successive application of purging transformations by composition, during execution time, for all leaf polygons of all obstacles P in $\mathcal{B}^{\mathcal{I}}_{map}$ and $\mathcal{D}^{\mathcal{I}}_{map}$, in any order, until their root polygons are reached. We denote by $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ this final intermediate space, where all obstacles in $\mathcal{F}^{\mathcal{I}}_{map}$ have been deformed to their root polygons. We denote by $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ the intermediate spaces before and after the purging transformation of leaf polygon $j_i \in \mathcal{V}_{P_i}$ respectively.

We begin our exposition with a description of the purging transformation $\mathbf{h}^{\mathcal{I}}_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \rightarrow \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ that maps the boundary of a leaf polygon $j_i \in \mathcal{V}_{P_i}$ onto the boundary of its

parent, $p(j_i)$, and continue with a description of the map $\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$ that maps the boundaries of root polygons of obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ and $\mathcal{D}^{\mathcal{I}}_{map}$ to $\mathcal{F}_e$ and the corresponding disks in $\mathcal{F}^{\mathcal{I}}_{model}$ respectively.

The map between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$

We first find admissible centers $\mathbf{x}^*_{j_i}$, and polygonal collars $\overline{\mathcal{Q}}_{j_i}$, that encompass the actual polygon $\mathcal{Q}_{j_i}$, and limit the effect of the purging transformation in their interior, while keeping its value equal to the identity everywhere else (see Fig. 8.3).

Definition 8.1. *An admissible center for the purging transformation of the leaf polygon $j_i \in \mathcal{V}_{P_i}$, denoted by $\mathbf{x}^*_{j_i}$, is a point in $p(j_i)$ such that the polygon $\mathcal{Q}_{j_i}$ with vertices the original vertices of j_i and $\mathbf{x}^*_{j_i}$ is convex.*

Definition 8.2. *An admissible polygonal collar for the purging transformation of the leaf polygon j_i is a convex polygon $\overline{\mathcal{Q}}_{j_i}$ such that:*

1. *$\overline{\mathcal{Q}}_{j_i}$ does not intersect the interior of any polygon $k \in \mathcal{V}_P$ with $k \neq j_i, p(j_i)$, for all polygons P involved in the construction of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$, or any $C \in \mathcal{C}_{map}$.*
2. *$\mathcal{Q}_{j_i} \subset \overline{\mathcal{Q}}_{j_i}$, and $\overline{\mathcal{Q}}_{j_i} \backslash \mathcal{Q}_{j_i} \subset \mathcal{F}^{\mathcal{I}}_{map,j_i}$.*

Examples are shown in Fig. 8.3. As in Chapter 7, we also construct implicit functions $\gamma_{j_i}(\mathbf{x}), \delta_{j_i}(\mathbf{x})$ corresponding to the leaf polygon $j_i \in \mathcal{V}_{P_i}$ such that $\mathcal{Q}_{j_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \gamma_{j_i}(\mathbf{x}) \leq 0\}$ and $\overline{\mathcal{Q}}_{j_i} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \delta_{j_i}(\mathbf{x}) \geq 0\}$, using tools from [176].

Based on these definitions, we construct the C^∞ *switch of the purging transformation for the leaf polygon $j_i \in \mathcal{V}_{P_i}$* as a function $\sigma_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathbb{R}$, equal to 1 on the boundary of $\mathcal{Q}_{j_i}$, equal to 0 outside $\overline{\mathcal{Q}}_{j_i}$ and smoothly varying (except the polygon vertices) between 0 and 1 everywhere else (see (7.15)). Finally, we define the *deforming factors* as the functions $\nu_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathbb{R}$, responsible for mapping the boundary of the leaf polygon j_i onto the boundary of its parent $p(j_i)$ (see (7.16)). We can now construct the map between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ as in (7.18)

$$\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) := \sigma_{j_i}(\mathbf{x}) \left(\mathbf{x}^*_{j_i} + \nu_{j_i}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*_{j_i})\right) + (1 - \sigma_{j_i}(\mathbf{x}))\, \mathbf{x}$$

Proposition 8.1. *The map $\mathbf{h}^{\mathcal{I}}_{j_i}$ is a C^∞ diffeomorphism between $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ away from the polygon vertices of j_i, none of which lies in the interior of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$.*

Proof. Included in Appendix C.6. □

We denote by $\mathbf{g}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ the map between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$, arising from the composition of purging transformations $\mathbf{h}^{\mathcal{I}}_{j_i} : \mathcal{F}^{\mathcal{I}}_{map,j_i} \to \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$.

The Map Between $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$

Here, for each root polygon r_i, we define the polygonal collar and the C^∞ *switch of the transformation* $\sigma_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{map}$ as in Definition 8.2 and (7.23) respectively, and we distinguish between obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$ and in $\mathcal{D}^{\mathcal{I}}_{map}$ for the definition of the centers as follows (see Fig. 8.3).

Definition 8.3. *An admissible center for the transformation of:*

1. *the root polygon r_i, corresponding to $P_i \in \mathcal{D}^{\mathcal{I}}_{map}$, is a point $\mathbf{x}^*_i$ in the interior of r_i (here identified with $\mathcal{Q}_{r_i}$).*

2. *the root polygon r_i, corresponding to $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, is a point $\mathbf{x}^*_i \in \mathbb{R}^2 \backslash \mathcal{F}_e$, such that the polygon $\mathcal{Q}_{r_i}$ with vertices the original vertices of r_i and $\mathbf{x}^*_i$ is convex.*

Finally, we define the *deforming factors* $\nu_{r_i} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathbb{R}$ as in Section 8.2.2 for obstacles in $\mathcal{B}^{\mathcal{I}}_{map}$, and as the function $\nu_{r_i}(\mathbf{x}) := \frac{\rho_i}{||\mathbf{x}-\mathbf{x}^*_i||}$ for obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$ (see Fig. 8.3). We construct the map between $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ as

$$\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) := \sum_{i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{B}} \cup \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}} \sigma_{r_i}(\mathbf{x}) \left[\mathbf{x}^*_i + \nu_{r_i}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*_i)\right] + \sigma_d(\mathbf{x})\mathbf{x}$$

with $\sigma_d(\mathbf{x}) := 1 - \sum_{i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{B}} \cup \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}} \sigma_{r_i}(\mathbf{x})$. It should be noted that Definitions 8.2 and 8.3 guarantee that, at any point in the workspace, at most one switch σ_{r_i} will be greater than zero which, in turn, guarantees that the diffeomorphism computation is essentially "local", and allows scaling to multiple obstacles in the mapped space $\mathcal{F}^{\mathcal{I}}_{map}$.

We can similarly arrive at the following result.

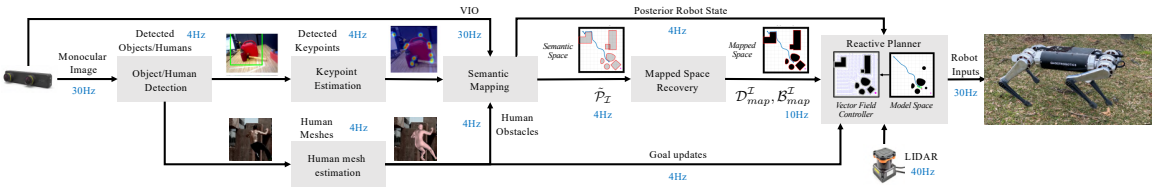

Figure 8.4: The online reactive planning architecture used in Chapter 8: Advancing beyond Chapter 7, camera output is run through a perceptual pipeline incorporating three separate neural networks (run onboard at 4Hz) whose function is to: (a) detect familiar obstacles and humans [158]; (b) localize corresponding semantic keypoints [148]; and (c) perform a 3D human mesh estimation [105]. Keypoint locations on the image, other detected geometric features, and an egomotion estimate provided by visual inertial odometry are used by the semantic mapping module [30] to give updated robot ($\mathbf{x}$) and obstacle poses ($\tilde{\mathcal{P}}_{\mathcal{I}}$). The reactive planner, now streamlined to run onboard at 3x the rate of the corresponding module in Chapter 7, merges consolidated obstacles in $\mathcal{D}^{\mathcal{I}}_{map}, \mathcal{B}^{\mathcal{I}}_{map}$ (recovered from $\tilde{\mathcal{P}}_{\mathcal{I}}$), along with LIDAR data for unknown obstacles, to provide the robot inputs and close the control loop. In this new architecture, the estimated human meshes are used to update the target's position in the reported human tracking experiments, detect a specific human gesture or pose related to the experiment's semantics, or (optionally) introduce additional obstacles in the semantic mapping module for some out-of-scope experiments.

Proposition 8.2. *The map $\hat{\mathbf{h}}^{\mathcal{I}}$ is a C^∞ diffeomorphism between $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ away from any sharp corners, none of which lie in the interior of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$.*

The Map Between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$

Based on the construction of $\mathbf{g}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\hat{\mathbf{h}}^{\mathcal{I}} : \hat{\mathcal{F}}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$, we can finally write the map between the mapped space and the model space as the function $\mathbf{h}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \to \mathcal{F}^{\mathcal{I}}_{model}$ given by $\mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \hat{\mathbf{h}}^{\mathcal{I}} \circ \mathbf{g}^{\mathcal{I}}(\mathbf{x})$. Since both $\mathbf{g}^{\mathcal{I}}$ and $\hat{\mathbf{h}}^{\mathcal{I}}$ are C^∞ diffeomorphisms away from sharp corners, it is straightforward to show that the map $\mathbf{h}^{\mathcal{I}}$ is a C^∞ diffeomorphism between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ away from any sharp corners, none of which lie in the interior of $\mathcal{F}^{\mathcal{I}}_{map}$.

8.3 Reactive Planning Algorithm

The analysis in Section 8.2 describes the diffeomorphism construction between $\mathcal{F}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ for a given index set $\mathcal{I}$ of instantiated familiar obstacles. However, the onboard sensor might incorporate new obstacles in the semantic map, updating $\mathcal{I}$. Therefore, as in Section 7.4, we give a hybrid systems description of our reactive controller, where each mode is defined by an index set $\mathcal{I} \in 2^{\mathcal{N}_{\mathcal{P}}}$ of familiar obstacles stored in the semantic map, the guards describe the sensor trigger events where a previously "unexplored" obstacle is discovered and incorporated in the semantic map (thereby changing $\mathcal{P}^{\mathcal{I}}_{map}$, and $\mathcal{D}^{\mathcal{I}}_{map}$, $\mathcal{B}^{\mathcal{I}}_{map}$),

and the resets describe transitions to new modes that are equal to the identity in the physical space, but might result in discrete "jumps" of the robot position in the model space. In this Section, this hybrid systems structure is not modified, and we just focus on each mode $\mathcal{I}$ separately.

For a fully actuated particle with dynamics $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}), \mathbf{u} \in \mathbb{R}^2$, the control law in each mode $\mathcal{I}$ is given as

$$\mathbf{u}^{\mathcal{I}}(\mathbf{x}) = k \left[D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}} \right]^{-1} \cdot \left(\mathbf{v}^{\mathcal{I}} \circ \mathbf{h}^{\mathcal{I}}(\mathbf{x}) \right) \tag{8.1}$$

with $D_{\mathbf{x}}$ denoting the derivative operator with respect to $\mathbf{x}$, and the control input in the model space given as [9]

$$\mathbf{v}^{\mathcal{I}}(\mathbf{y}) = - \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d) \right) \tag{8.2}$$

Here, $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x}) \in \mathcal{F}^{\mathcal{I}}_{model}$ and $\mathbf{y}_d = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$ denote the robot and goal position in the model space respectively, and $\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)$ denotes the projection onto the convex *local freespace* for $\mathbf{y}$, $\mathcal{LF}(\mathbf{y})$, defined as the Voronoi cell in (7.39), separating $\mathbf{y}$ from all the model space obstacles (see Fig. 8.2). We use the following definition to define a slowly moving, non-adversarial moving target.

Definition 8.4. *The smooth function $\mathbf{x}_d : \mathbb{R} \to \mathcal{F}^{\mathcal{I}}_{map}$ is a non-adversarial target if its model space velocity, given as $\dot{\mathbf{y}}_d := D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d) \cdot \dot{\mathbf{x}}_d$, always satisfies either $(\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))^{\top} \dot{\mathbf{y}}_d \geq 0$, or $||\dot{\mathbf{y}}_d|| \leq k \dfrac{||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \Pi_{\mathsf{B}\left(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), 0.5 d(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \partial \mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))||^2}{||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||}$.*

Intuitively, this Definition requires the moving target to slow down when the robot gets too close to obstacles (i.e., when $d(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \partial \mathcal{F}^{\mathcal{I}}_{model})$ becomes small) or the target itself (i.e., when $\Pi_{\mathsf{B}\left(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), 0.5 d(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \partial \mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)) = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$), proportionally to the control gain k, unless the target approaches the robot (i.e., $(\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))^{\top} \dot{\mathbf{y}}_d \geq 0$). We use Definition 8.4 to arrive at the following central result.

Theorem 8.1. *With $\mathcal{I}$ the terminal mode of the hybrid controller (see Definition 7.8), the reactive controller in* (8.1) *leaves the freespace $\mathcal{F}^{\mathcal{I}}_{map}$ positively invariant, and:*

1. *tracks $\mathbf{x}_d$ by not increasing $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$, if $\mathbf{x}_d$ is a non-adversarial target (see*

Definition 8.4).

2. *asymptotically reaches a constant* $\mathbf{x}_d$ *with its unique continuously differentiable flow, from almost any placement* $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map}$*, while strictly decreasing* $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$ *along the way.*

Proof. Included in Appendix C.6. □

In Chapter 6, we extend our algorithm to differential drive robots, by constructing a smooth diffeomorphism $\overline{\mathbf{h}}^{\mathcal{I}} : \mathcal{F}^{\mathcal{I}}_{map} \times S^1 \to \mathcal{F}^{\mathcal{I}}_{model} \times S^1$ away from sharp corners, as shown in (6.26). Based on this construction, we present our main result below, whose proof follows similar patterns to that of Theorem 8.1 and is omitted for brevity.

Theorem 8.2. *With* $\mathcal{I}$ *the terminal mode of the hybrid controller (see Definition 7.8), the reactive controller for differential drive robots* (6.26) *leaves the freespace* $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ *positively invariant, and:*

1. *tracks* $\mathbf{x}_d$ *by not increasing* $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$*, if* $\mathbf{x}_d$ *is a non-adversarial target (see Definition 8.4).*

2. *asymptotically reaches a constant* $\mathbf{x}_d$ *with its unique continuously differentiable flow, from almost any robot configuration in* $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$*, without increasing* $||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||$ *along the way.*

8.4 Numerical Studies

In this Section, we present numerical studies run in MATLAB using `ode45`, that illustrate our formal results. Our reactive controller is implemented in Python and communicates with MATLAB using the standard MATLAB-Python interface. For our numerical results, we assume perfect robot state estimation and localization of obstacles, using a fixed range sensor that can instantly identify and localize either the entirety of familiar obstacles that intersect its footprint, or the fragments of unknown obstacles within its range.

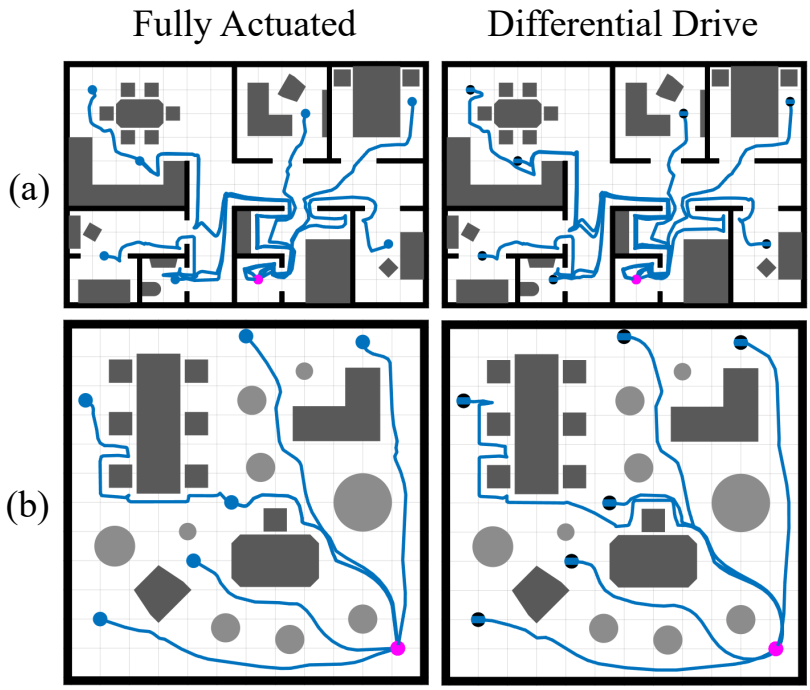

Figure 8.5: Top: Navigation in an indoor layout cluttered with multiple familiar obstacles and previously unknown pose. - Bottom: Navigation in a room cluttered with known non-convex (dark grey) and unknown convex (light grey) obstacles. Simulations are run from different initial conditions.

8.4.1 Illustrations of the Navigation Framework

We begin by illustrating the performance of our reactive planning framework in two different settings (Fig. 8.5), for both a fully actuated and a differential drive robot. In the first case (Fig. 8.5-a), the robot is tasked with moving to a predefined location in an environment resembling an apartment layout with known walls, cluttered with several familiar obstacles of unknown location and pose, from different initial conditions. In the second case (Fig. 8.5-b), the robot navigates a room cluttered with both familiar and unknown obstacles from several initial conditions. In both cases, the robot avoids all the obstacles and safely converges to the target. The robot radius used in our simulation studies is 0.2m.

8.4.2 Comparison with RRT^X [143]

In the second set of numerical results, we compare our reactive controller with a state-of-the-art path replanning algorithm, RRT^X [143]. We choose to compare against this specific algorithm instead of another sampling-based method for static environments (e.g.,

RRT* [96]), since both our reactive controller and RRT^X are dynamic in nature; they are capable of incorporating new information about the environment and modifying the robot's behavior appropriately. For our simulations, we assume that RRT^X possesses the same sensory apparatus with our algorithm; an "oracle" that can instantly identify and localize nearby obstacles. The computed paths are then reactively tracked using [8].

Fig. 8.6-a exemplifies the (well-known [118]) performance degradation of RRT^X in the presence of narrow passages: as the corridor narrows (while always larger than the robot's diameter), the minimum number of (offline-computed) samples needed for successful replanning and safe navigation increases in a nonlinear manner. In consequence of this dramatically growing time-to-completeness, the accompanying video of [205][4] demonstrates a potentially catastrophic failure of the associated replanner: in the presence of multiple narrow passages, it cycles repeatedly as it searches for possible alternative openings, before eventually (and only after increasingly protracted cycling) reporting failure (incorrectly) and halting. On the contrary, our algorithm always guarantees safe passage to the target through compliant environments – and Fig. 8.6-b illustrates its graceful failure for settings that violate Assumption 8.1. The non-compliant (novel but not convex) obstacle creates an attracting equilibrium state that traps a set of initial conditions whose area becomes arbitrarily large as its "shadow" (the associated basin of attraction) grows. However, the presence of a Lyapunov function precludes the possibility of any cycling behavior: failure to achieve the goal (and the diagnosis of a non-compliant environment) is readily identified.

8.5 Experiments

8.5.1 Experimental Setup

Our experimental layout is summarized in Fig. 8.4. Since the algorithms introduced in this Section take the form of first-order vector fields, we mainly use a quasi-static platform, the Turtlebot robot [194] for our physical experiments. We suggest the robustness of these feedback controllers by performing several experiments on the more dynamic Ghost Spirit

[4] `https://youtu.be/0ql1BaPcozc`

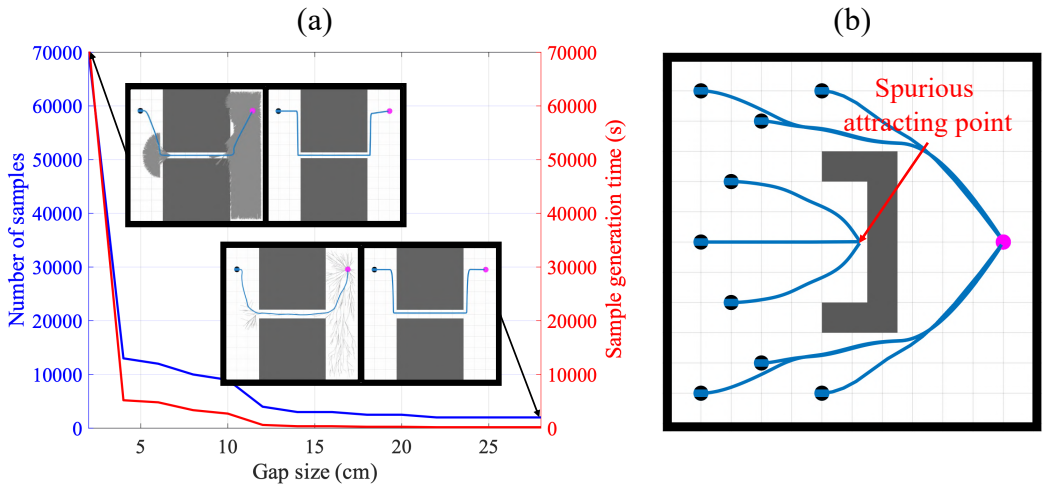

Figure 8.6: (a) Minimum number of (offline computed) samples needed for successful online implementation of RRT^X [143] in an unexplored environment with two familiar obstacles forming a narrow passage. The number becomes increasingly large as the gap becomes smaller. The robot diameter is 50cm. (b) Illustration of a graceful failure of our proposed algorithm. The sole non-convex but unknown encountered obstacle creates a spurious attracting equilibrium state that traps a subset of initial conditions. However, collision avoidance is always guaranteed by the onboard sensor.

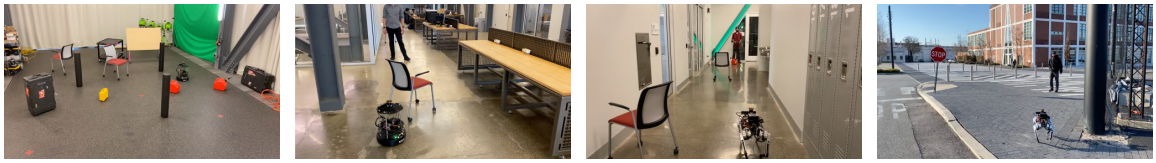

Figure 8.7: Types of environments used in our experiments. Visual context is included in the supplementary video[4].

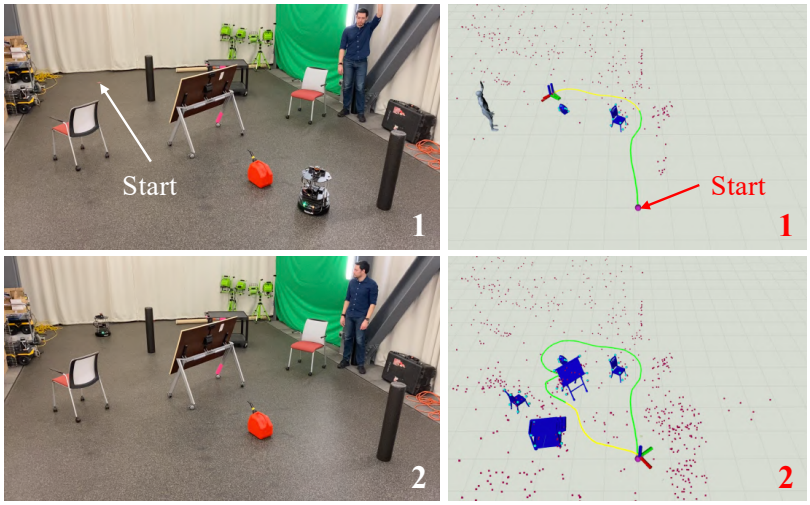

Figure 8.8: Top: Turtlebot reactively follows a human until a stop gesture is given and detected – Bottom: Turtlebot safely returns to its starting position.

legged robot [67], using a rough approximation to the quasi-static differential drive motion model as reported in Section 3.1.3. In both cases, the main computer is an Nvidia Jetson AGX Xavier GPU unit, responsible for running our perception and navigation algorithms, during execution time. This GPU unit communicates with a Hokuyo LIDAR, used to detect unknown obstacles, and a ZED Mini stereo camera, used for visual-inertial state estimation and for detecting humans and familiar obstacles.

Our perception pipeline, run onboard the Nvidia Jetson AGX Xavier at 4Hz, supports the detection and 3D pose estimation of objects and humans, who, for the purposes of this work, are used as moving targets. We use the YOLOv3 detector [158] to detect 2D bounding boxes on the image which are then processed based on the class of the detected object. If one of the specified object classes is detected, then we follow the semantic keypoints approach of [148] to estimate keypoints of the object on the image plane[5]. Similarly to Chapter 7, the familiar object classes (as defined in Section 8.1) used in our experiments are chair, table, ladder, cart, gascan and pelican case, although this dictionary can increase depending on the user's needs. The training data for the particular instances of interest are collected with a semi-automatic procedure, similarly to [148]. Given the bounding box and keypoint annotations for each image, the two networks are trained with their default configurations until convergence. On the other hand, if the bounding box corresponds to a person detection, then we use the approach of [105], that provides us with the 3D mesh of the person.

As also reported in Section 7.7, our semantic mapping infrastructure relies on the algorithm presented in [30], and is implemented in C++. This algorithm fuses inertial information (here provided by the position tracking implementation from StereoLabs on the ZED Mini stereo camera), geometric (i.e., geometric features on the 2D image), and semantic information (i.e., the detected keypoints and the associated object labels as described above) to give a posterior estimate for both the robot state and the associated poses of all tracked objects, by simultaneously solving the data association problem arising when several objects

[5]Note that while both the YOLOv3 detector [158] and the keypoint estimation algorithm [148] are empirically very robust (e.g., particularly against partial occlusions), they could be easily replaced with other state-of-the-art algorithms that provide reasonable robustness against partial occlusions.

of the same class exist in the map.

Finally, our reactive controller, running online and onboard the Nvidia Jetson AGX Xavier GPU unit at 30Hz, is also implemented in C++ using Boost Geometry [170] for the underlying polygon operations, and communicates with our perception pipelines using ROS, as shown in Fig. 8.4.

8.5.2 Empirical Results

As also reported in the supplementary video[4], we distinguish between two classes of physical experiments in several different environments shown in Fig. 8.7; tracking either a predefined static target or a moving human, and tracking a given semantic target (e.g., approach a desired object).

Geometric tracking of a (moving) target amidst obstacles

Fig. 8.1 shows Spirit tracking a human in a previously unexplored environment, cluttered with both catalogued obstacles (whose number and placement is unknown in advance) as well as completely unknown obstacles. The robot uses familiar obstacles to both localize itself against them [30] and reactively navigate around them. Fig. 8.7 summarizes the wide diversity of à-priori unexplored environments, with different lighting conditions, successfully navigated indoors (by Turtlebot and Spirit) and outdoors (by Spirit), while tracking humans[6] along thousands of body lengths.

Note that the formal results of Section 8.3 require that unknown obstacles be convex. However, here we clutter the environment with a mix of unknown obstacles – some convex, but others of more complicated non-convex shapes (e.g., unknown walls) – to establish empirical robustness in urban environments that are out of scope of the underlying theory. In such settings, that move beyond the formal assumptions outlined in Section 8.1, the robot might converge to undesired local minima behind non-convex obstacles from a subset of (unfavorable) initial conditions (see Fig. 8.6-b); however, collision avoidance is still guaranteed

[6]Collision avoidance when the robot gets close to the tracked human is guaranteed with the use of the onboard LIDAR; the human is treated as an unknown obstacle and the robot tries to keep separation and avoid collision (with formal guarantees assuming the conditions of Definition 8.4).

by the onboard LIDAR.

As anticipated, the few failures we recorded were associated with the inability of the SLAM algorithm to localize the robot in long, featureless environments. However, it should be noted that even when the robot or object localization process fails, collision avoidance is still guaranteed with the use of the onboard LIDAR. Nevertheless, collisions could result with obstacles that cannot be detected by the 2D horizontal LIDAR (e.g., the red gascan shown in Fig. 8.8). One could still think of extensions to the presented sensory infrastructure (e.g., the use of a 3D LIDAR) that could at least still guarantee safety under such circumstances.

Logical reaction using predefined semantics

In the second set of experimental runs, we exploit the new online semantic capabilities to introduce logic in our reactive tracking process. For example, Fig. 8.8 depicts a tracking task requiring the robot to respond to the human's stop signal (raised left or right hand) by returning to its starting position. The supplementary video[4] presents several other semantically specified tasks requiring autonomous reactions of both a logical as well as geometric nature (all involving negotiation of novel environments from the arbitrary geometric circumstances associated with different contexts of logical triggers).

Finally, apart from introducing semantics related to the estimated human pose, we also consider cases where the behavior of the robot changes on the fly based on the detected objects. In an experimental run reported in the accompanying video[4], the robot is tasked with moving to a predefined geometric target, unless it sees and localizes a cart; in that case, it is tasked with properly approaching and facing the cart with its camera, while avoiding all (previously unknown) obstacles along the way. Based on these results, we believe that our algorithm can be coupled with the hierarchical control scheme reported in Part II for accomplishing more sophisticated mobile manipulation tasks (e.g., using temporal logic [111]).

8.6 Discussion

Chapters 6 - 8 present a reactive navigation scheme for robots operating in planar workspaces, cluttered with obstacles of familiar geometry but à-priori unknown placement, and completely unknown, but strongly convex and well-separated obstacles. To the best of our knowledge, this is the first doubly reactive navigation framework (i.e., a scheme where not only the robot's trajectory but also the vector field that generates it are computed online at execution time) that can handle arbitrary polygonal shapes in real time without the need for specific separation assumptions between the familiar obstacles. The resulting algorithm combines state-of-the-art perception and object recognition techniques (based on neural network architectures) for familiar obstacles, with local range measurements (e.g., LIDAR) for the unknown obstacles, to yield provably correct navigation in geometrically complicated environments. We illustrate the practicability of this approach by reporting empirical results using modest computational hardware on a wheeled robot, and the intrinsic robustness of such reactive schemes by implementation on dynamic legged platforms, exhibiting imperfect fidelity to the differential drive model assumed in the formal results, while also provably safely semantically engage non-adversarial moving targets.

Part IV

Reactive Semantic Planning for Mobile Manipulation

Chapter 9

Reactive Planning for Mobile Manipulation Tasks in Unexplored Semantic Environments

Complex manipulation tasks, such as rearrangement planning of numerous objects, are combinatorially hard problems. Existing algorithms either do not scale well or assume a great deal of prior knowledge about the environment, and few offer any rigorous guarantees. In this Chapter, we propose a novel hybrid control architecture for achieving such tasks with mobile manipulators, based on the reactive controller presented in Chapter 8. On the discrete side, we enrich a temporal logic specification with mobile manipulation primitives such as moving to a point, and grasping or moving an object. Such specifications are translated to an automaton representation, which orchestrates the physical grounding of the task to mobility or manipulation controllers. The grounding from the discrete to the continuous reactive controller is online and can respond to the discovery of unknown obstacles or decide to push out of the way movable objects that prohibit task accomplishment. Despite the problem complexity, we prove that, under specific conditions, our architecture enjoys provable completeness on the discrete side, provable termination on the continuous side, and avoids all obstacles in the environment. Simulations illustrate the efficiency of our ar-

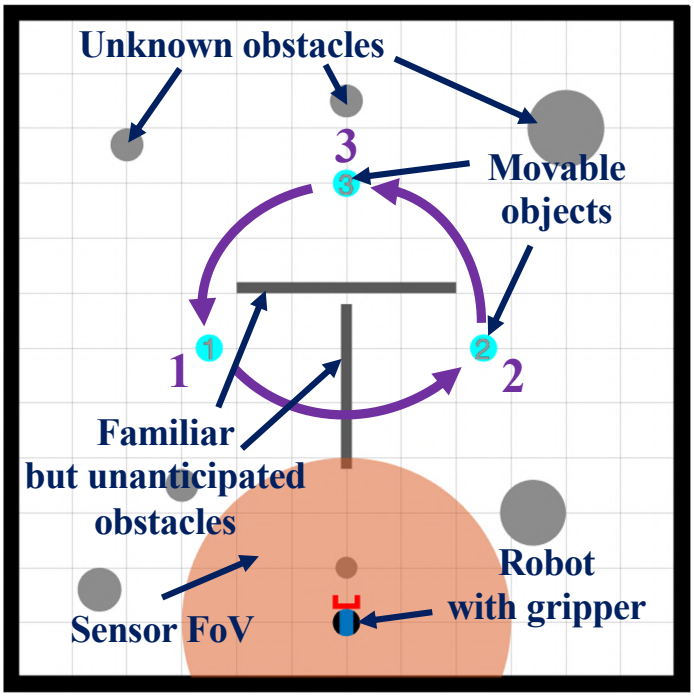

Figure 9.1: An example of a task considered in this Chapter, whose execution is depicted in Fig. 9.7. A differential drive robot, equipped with a gripper (red) and a limited range onboard sensor for localizing obstacles (orange), needs to accomplish a mobile manipulation task specified by a Linear Temporal Logic (LTL) formula, in a partially known environment (black), cluttered with both unanticipated (dark grey) and completely unknown (light grey) fixed obstacles. Here the task is to rearrange the movable objects counterclockwise, in the presence of the fixed obstacles. Objects' abstract locations (relative to abstract, named regions of the workspace) are known by the symbolic controller both à-priori and during the entire task sequence. Geometrically complicated obstacles are assumed to be familiar but unanticipated in the sense that neither their number nor placement are known in advance. Completely unknown obstacles are presumed to be convex. All obstacles and disconnected configurations caused by the movable objects are handled by the reactive vector field motion planner (Fig. 9.2) and never reported to the symbolic controller.

chitecture that can handle tasks of increased complexity while also responding to unknown obstacles or unanticipated adverse configurations.

The Chapter is organized as follows. After formulating the problem in Section 9.1, Section 9.2 presents a discrete controller which given an LTL specification generates on-the-fly high-level manipulation primitives, translated to point navigation commands through an interface layer outlined in Section 9.3. Using this interface, Section 9.4 continues with the reactive implementation of our symbolic actions and the employed algorithm for connecting disconnected freespace components blocked by movable objects. Finally, Section 9.5 discusses our numerical results.

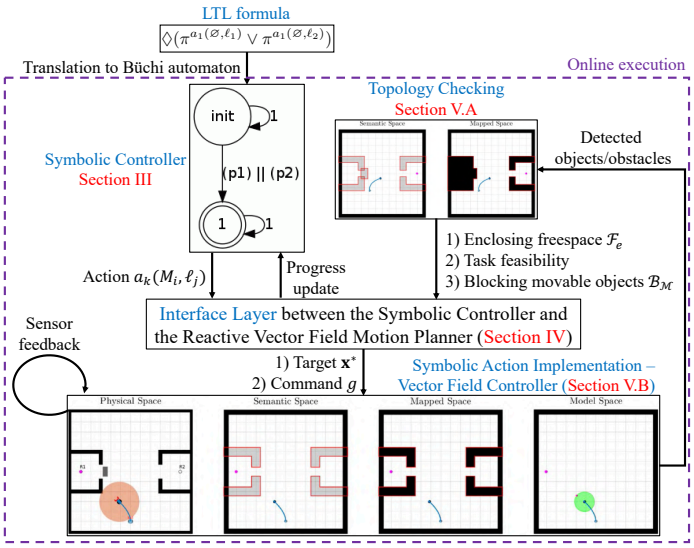

Figure 9.2: System architecture, following Fig. 1.1, without the (platform-specific) gait layer. The task is encoded in an LTL formula, translated offline to a Büchi automaton (symbolic controller - Section 9.2). Then, during execution time in a previously unexplored semantic environment, each individual sub-task provided by the Büchi automaton is translated to a point navigation task toward a target $\mathbf{x}_d$ and a gripper command g, through an interface layer (Section 9.3). This task is executed online by realizing each symbolic action (Section 9.4.3) using a reactive, vector field motion planner (continuous-time controller, Chapter 8) implementing closed-loop navigation using sensor feedback and working closely with a topology checking module (Section 9.4.2), responsible for detecting freespace disconnections. The reactive controller guarantees collision avoidance and target convergence when both the initial and the target configuration lie in the same freespace component. On the other hand, if the topology checking module determines that the target is not reachable, the reactive controller either attempts to connect the disconnected configuration space by switching to a *Fix mode* and interacting with the environment in order to rearrange blocking movable objects, or the interface layer reports failure to the symbolic controller when this is impossible and requests an alternative action.

9.1 Problem Description

9.1.1 Model of the Robot and the Environment

We consider a first-order, nonholonomically-constrained, disk-shaped robot, centered at $\mathbf{x} \in \mathbb{R}^2$ with radius $r \in \mathbb{R}_{>0}$ and orientation $\psi \in S^1$; its rigid placement is denoted by $\overline{\mathbf{x}} := (\mathbf{x}, \psi) \in \mathbb{R}^2 \times S^1$ and its input vector $\overline{\mathbf{u}} := (v, \omega)$ consists of a fore-aft and an angular velocity command. The robot uses a gripper to move disk-shaped *movable objects* of known location, denoted by $\tilde{\mathcal{M}} := \{\tilde{M}_i\}_{i \in \{1,\ldots,N_M\}}$, with a vector of radii $(\rho_1, \ldots, \rho_{N_M}) \in \mathbb{R}^{N_M}$, in a closed, compact, polygonal, typically non-convex workspace $\mathcal{W} \subset \mathbb{R}^2$. The robot's gripper g can either be engaged ($g = 1$) or disengaged ($g = 0$). Moreover, we adopt the perceptual model of Chapter 8 whereby a sensor of range R recognizes and instantaneously localizes

any fixed "familiar" or "unfamiliar" obstacles; see also Fig. 9.1.

The workspace is cluttered by a finite collection of disjoint obstacles of unknown number and placement, denoted by $\tilde{\mathcal{O}}$. This set might also include non-convex "intrusions" of the boundary of the physical workspace $\mathcal{W}$ into the convex hull of the closure of the workspace $\mathcal{W}$, defined as the *enclosing workspace*. As in Chapters 7 - 8, we define the *freespace* $\mathcal{F}$ as the set of collision-free placements for the closed ball $\overline{\mathsf{B}(\mathbf{x}, r)}$ centered at $\mathbf{x}$ with radius r, and the *enclosing freespace*, $\mathcal{F}_e$, as $\mathcal{F}_e := \left\{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \mathrm{Conv}(\overline{\mathcal{F}})\right\}$.

Although none of the positions of any obstacles in $\tilde{\mathcal{O}}$ are à-priori known, a subset $\tilde{\mathcal{P}} \subseteq \tilde{\mathcal{O}}$ of these obstacles is assumed to be "familiar" in the sense of having a recognizable polygonal geometry, that the robot can instantly identify and localize (see Chapter 8). Similarly to Chapters 7 - 8, the remaining obstacles in $\tilde{\mathcal{C}} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{P}}$ are assumed to be strongly convex according to Assumption 7.1, but are otherwise completely unknown.

To simplify the notation, we dilate each obstacle and movable object by r (or $r + \rho_i$ when the robot carries an object i), and assume that the robot operates in the freespace $\mathcal{F}$. We denote the set of dilated objects and obstacles derived from $\tilde{\mathcal{M}}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}}$ and $\tilde{\mathcal{C}}$, by $\mathcal{M}, \mathcal{O}, \mathcal{P}$ and $\mathcal{C}$ respectively. For our formal results, we assume that each obstacle in $\mathcal{C}$ is always well-separated from all other obstacles in both $\mathcal{C}$ and $\mathcal{P}$, as outlined in Assumption 7.2; in practice, the surrounding environment often violates our separation assumptions, without precluding successful task completion.

9.1.2 Specifying Complex Manipulation Tasks

The robot needs to accomplish a mobile manipulation task, by visiting known regions of interest $\ell_j \subseteq \mathcal{W}$, where $j \in \{1, \ldots, L\}$, for some $L > 0$, and applying one of the following three manipulation actions $a_k(M_i, \ell_j) \in \mathcal{A}$, with $M_i \in \mathcal{M}$ referring to a movable object, defined as follows:

- MOVE(ℓ_j) instructing the robot to move to region ℓ_j, labeled as $a_1(\varnothing, \ell_j)$, where $\varnothing$ means that this action does not logically entail interaction with any specific movable object[1].

[1] Although, as will be detailed in Section 9.4, the hybrid reactive controller may actually need to move

- GRASPOBJECT(M_i) instructing the robot to grasp the movable object M_i, labeled as $a_2(M_i, \varnothing)$, with $\varnothing$ denoting that no region is associated with this action.

- RELEASEOBJECT(M_i, ℓ_j) instructing the robot to push the (assumed already grasped) object M_i toward its designated goal position, ℓ_j, labeled as $a_3(M_i, \ell_j)$.

For instance, consider a rearrangement planning scenario where the locations of three objects of interest need to be rearranged, as in Fig. 9.1. We capture such complex manipulation tasks via Linear Temporal Logic (LTL) specifications. Specifically, we use atomic predicates of the form $\pi^{a_k(M_i,\ell_j)}$, which are true when the robot applies the action $a_k(M_i, \ell_j)$ and false until the robot achieves that action. Note that these atomic predicates allow us to specify temporal logic specifications defined over manipulation primitives and, unlike related works [74, 179], are entirely agnostic to the geometry of the environment. We define LTL formulas by collecting such predicates in a set $\mathcal{AP}$ of atomic propositions. For example, the rearrangement planning scenario with three movable objects initially located in regions ℓ_1, ℓ_2, and ℓ_3, as shown in Fig. 9.1, can be described as a sequencing task [56] by the following LTL formula:

$$\begin{aligned}\phi = &\Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond(\pi^{a_3(M_1,\ell_2)} \wedge \\ &\Diamond(\pi^{a_2(M_2,\varnothing)} \wedge \Diamond\pi^{a_3(M_2,\ell_3)} \wedge \\ &\Diamond(\pi^{a_2(M_3,\varnothing)} \wedge \Diamond\pi^{a_3(M_3,\ell_1)}))))\end{aligned} \tag{9.1}$$

where $\Diamond$ and $\wedge$ refer to the 'eventually' and 'AND' operator. In particular, this task requires the robot to perform the following steps in this order (i) grasp object M_1 and release it in location ℓ_2 (first line in (9.1)); (ii) then grasp object M_2 and release it in location ℓ_3 (second line in (9.1)); (iii) grasp object M_3 and release it in location ℓ_1 (third line in (9.1)). LTL formulas are satisfied over an infinite sequence of states [17]. Unlike related works where a state is defined to be the robot position, e.g., [111], here a state is defined by the

objects out of the way, rearranging the topology of the workspace in a manner hidden from the logical task controller.

manipulation action $a_k(M_i, \ell_j)$ that the robot applies. In other words, an LTL formula defined over manipulation-based predicates $\pi^{a_k(M_i,\ell_j)}$ is satisfied by an infinite sequence of actions $p = p_0, p_1, \ldots, p_n, \ldots$, where $p_n \in \mathcal{A}$, for all $n \geq 0$ [17]. Given a sequence p, the syntax and semantics of LTL can be found in [17]; hereafter, we exclude the 'next' operator from the syntax, since it is not meaningful for practical robotics applications [103], as well as the negation operator[2].

9.1.3 Problem Statement

Given a task specification captured by an LTL formula ϕ, our goal is to (i) generate online, as the robot discovers the environment via sensor feedback, appropriate actions using the (discrete) symbolic controller, (ii) translate them to point navigation tasks, (iii) execute these navigation tasks and apply the desired manipulation actions with a (continuous-time) vector field controller, while avoiding unknown and familiar obstacles, (iv) be able to online detect freespace disconnections that prohibit successful action completion, and (v) either locally amend the provided plan by disassembling blocking movable objects, or report failure to the symbolic controller and request an alternative action.

9.2 Symbolic Controller

In this Section, we design a discrete controller that generates manipulation commands online in the form of the actions defined in Section 9.1 (e.g., 'release the movable object M_i at a region ℓ_j'). A summary of the overall architecture is given in Fig. 9.2, and we now proceed to outline the manner in which the symbolic controller depicted there extends prior work [92] to account for the manipulation-based atomic predicates defined in Section 9.1 and adapted to the single-agent setting. To accomplish this, first in Section 9.2.1 we translate the LTL formula into a Non-deterministic Büchi Automaton (NBA) and we provide a formal definition of its accepting condition. Then, in Section 9.2.2, we provide a detailed description for the construction of the distance metric over this automaton state space. Section 9.2.3

[2]Since the negation operator is excluded, safety requirements, such as obstacle avoidance, cannot be captured by the LTL formula; nevertheless, the proposed method can still handle safety constraints by construction of the (continuous-time) reactive, vector field controller in Section 9.4.

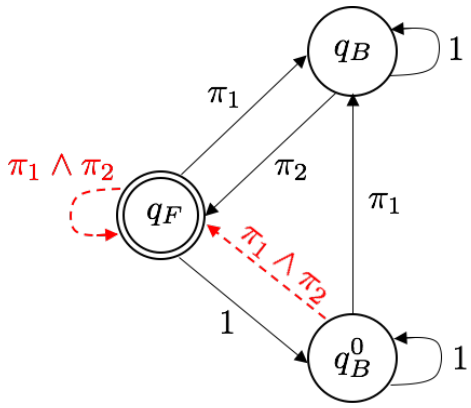

Figure 9.3: Graphical illustration of the NBA corresponding to the LTL formula $\phi = \square\lozenge(\pi_1) \wedge \square\lozenge(\pi_2)$ where for simplicity of notation $\pi_1 = \pi^{a_1(\varnothing,\ell_1)}$ and $\pi_2 = \pi^{a_1(\varnothing,\ell_2)}$. The automaton has been generated using the tool in [64]. In words, this LTL formula requires the robot to visit infinitely often and in any order the regions ℓ_1 and ℓ_2. The initial state of the automaton is denoted by q_B^0 while the final state is denoted by q_F. When the robot is in an NBA state and the Boolean formula associated with an outgoing transition from this NBA state is satisfied, then this transition can be enabled. For instance, when the robot is in the initial state q_B^0 and satisfies the atomic predicate π_1, the transition from q_B^0 to q_B can be enabled, i.e., $q_B \in \delta_B(q_B^0, \pi_1)$. The LTL formula is satisfied if starting from q_B^0, the robot generates an infinite sequence of observations (i.e., atomic predicates that become true) that yields an infinite sequence of transitions so that the final state q_F is visited infinitely often. The red dashed lines correspond to infeasible NBA transitions as they are enabled only if the Boolean formula $\pi_1 \wedge \pi_2$ is satisfied, i.e., only if the robot is in more than one region simultaneously; such edges are removed yielding the pruned NBA.

describes our method for generating symbolic actions online, and Section 9.2.4 includes our completeness result. The proposed method is also illustrated in Figs. 9.3 - 9.4.

9.2.1 Construction of the Symbolic Controller

First, we translate the specification ϕ, constructed using a set of atomic predicates $\mathcal{AP}$, into a Non-deterministic Büchi Automaton (NBA) with state-space and transitions among states that can be used to measure how much progress the robot has made in terms of accomplishing the assigned mission, defined as follows.

Definition 9.1 (NBA). *A Non-deterministic Büchi Automaton (NBA) B over $\Sigma = 2^{\mathcal{AP}}$ is defined as a tuple $B = \left(\mathcal{Q}_B, \mathcal{Q}_B^0, \delta_B, \mathcal{Q}_F\right)$, where (i) $\mathcal{Q}_B$ is the set of states; (ii) $\mathcal{Q}_B^0 \subseteq \mathcal{Q}_B$ is a set of initial states; (iii) $\delta_B : \mathcal{Q}_B \times \Sigma \to 2^{\mathcal{Q}_B}$ is a non-deterministic transition relation, and $\mathcal{Q}_F \subseteq \mathcal{Q}_B$ is a set of accepting/final states.*

To interpret a temporal logic formula over the trajectories of the robot system, we use a labeling function $L : \mathcal{A} \to 2^{\mathcal{AP}}$ that determines which atomic propositions are true given the

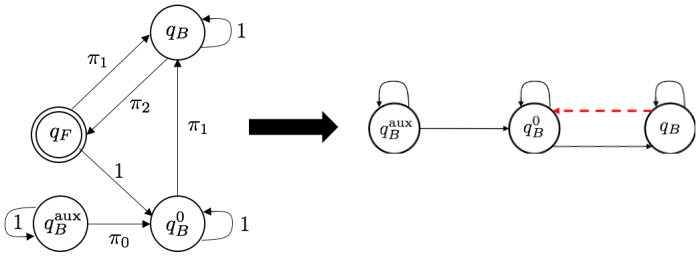

Figure 9.4: Graphical illustration of the graph $\mathcal{G}$ construction for the NBA shown in Fig. 9.3. The left figure corresponds to the pruned automaton after augmenting its state space with the state q_B^{aux}, where π_0 corresponds to the atomic predicate that the robot satisfies initially at $t = 0$. If no atomic predicates are satisfied initially, then π_0 corresponds to the empty symbol [17]. Observe in the left figure that $\mathcal{D}_{q_B^{\text{aux}}} = \{q_B^{\text{aux}}, q_B^0, q_B\}$. The right figure illustrates the graph $\mathcal{G}$ corresponding to this automaton. The red dashed line corresponds to an accepting edge. Also, we have that $\mathcal{V}_F = \{q_B\}$, $d_F(q_B^{\text{aux}}, \mathcal{V}_F) = 2$, $d_F(q_B^0, \mathcal{V}_F) = 1$, and $d_F(q_B, \mathcal{V}_F) = 0$. For instance, every time the robot reaches the state q_B^0 with $d_F(q_B^0, \mathcal{V}_F) = 1$, it generates a symbol to reach the state q_B since reaching this state decreases the distance to the set of accepting edges (since $d_F(q_B, \mathcal{V}_F) = 0$). The symbol that can enable this transition is the symbol that satisfies the Boolean formula $b^{q_B^0, q_B} = \pi_1$; this formula is trivially satisfied by the symbol $\pi_1 = \pi^{a_1(\varnothing, \ell_1)}$. As a result the command send to the continuous time controller is 'Move to Region ℓ_1'.

current robot action $a_k(M_i, \ell_j)$; note that, by definition, these actions also encapsulate the position of the robot in the environment. An infinite sequence $p = p(0)p(1)\dots p(k)\dots$ of actions $p(k) \in \mathcal{A}$, satisfies ϕ if the word $\sigma = L(p(0))L(p(1))\dots$ yields an accepting NBA run defined as follows [17]. First, a run ρ_B of B over an infinite word $\sigma = \sigma(1)\sigma(2)\dots\sigma(k)\cdots \in (2^{\mathcal{AP}})^\omega$, is a sequence $\rho_B = q_B^0 q_B^1 q_B^2 \dots, q_B^k, \dots$, where $q_B^0 \in \mathcal{Q}_B^0$ and $q_B^{k+1} \in \delta_B(q_B^k, \sigma(k))$, $\forall k \in \mathbb{N}$. A run ρ_B is called *accepting* if at least one final state appears infinitely often in it. In words, an infinite-length discrete plan τ satisfies an LTL formula ϕ if it can generate at least one accepting NBA run.

9.2.2 Distance Metric Over the NBA

In this Section, given a graph constructed using the NBA, we define a function to compute how far an NBA state is from the set of final states. Following a similar analysis as in [91, 92], we first prune the NBA by removing infeasible transitions that can never be enabled as they require the robot to be in more than one region and/or take more that one action simultaneously. Specifically, a symbol $\sigma \in \Sigma := 2^{\mathcal{AP}}$ is *feasible* if and only if $\sigma \not\models b^{\text{inf}}$, where b^{inf} is a Boolean formula defined as

$$b^{\text{inf}} = [(\vee_{\forall k,r,j,e\neq j}(\pi^{a_k(\cdot,\ell_e)} \wedge \pi^{a_r(\cdot,\ell_j)}))] \bigvee [(\vee_{\forall j,k,r\neq k}(\pi^{a_k(\cdot,\ell_j)} \wedge \pi^{a_r(\cdot,\ell_j)}))] \tag{9.2}$$

In words, b^{inf} requires the robot to be either present simultaneously in more than one region *or* take more than one action in a given region at the same time. Specifically, the first line requires the robot to be present in locations ℓ_j and ℓ_e, $e \neq j$ and apply the actions $a_k, a_r \in \mathcal{A}$ while the second line requires the robot to take two distinct actions $a_k(\cdot, \ell_j)$ and $a_r(\cdot, \ell_j)$ at the same region ℓ_j, simultaneously.

Next, we define the sets that collect all feasible symbols that enable a transition from an NBA state q_B to another, not necessarily different, NBA state q'_B. This definition relies on the fact that transition from a state q_B to a state q'_B is enabled if a Boolean formula, denoted by b^{q_B,q'_B} and defined over the set of atomic predicates $\mathcal{AP}$, is satisfied. In other words, $q'_B \in \delta_B(q_B, \sigma)$, i.e., q'_B can be reached from the NBA state q_B under the symbol σ, if σ satisfies b^{q_B,q'_B}. An NBA transition from q_B to q'_B is infeasible if there are no feasible symbols that satisfy b^{q_B,q'_B}. All infeasible NBA transitions are removed yielding a pruned NBA automaton. All feasible symbols that satisfy b^{q_B,q'_B} are collected in the set Σ^{q_B,q'_B}.

To take into account the initial robot state in the construction of the distance metric, in the pruned automaton we introduce an auxiliary state q_B^{aux} and transitions from q_B^{aux} to all initial states $q_B^0 \in \mathcal{Q}_B^0$ so that $b^{q_B^{\text{aux}},q_B^{\text{aux}}} = 1$ and $b^{q_B^{\text{aux}},q_B^0} = \pi_0$, i.e., transition from q_B^{aux} to q_B^0 can always be enabled based on the atomic predicate that is initially satisfied denoted by π_0; note that if no predicates are satisfied initially, then π_0 corresponds to the empty symbol [17]. Hereafter, the auxiliary state q_B^{aux} is considered to be the initial state of the resulting NBA; see also Fig. 9.4.

Next, we collect all NBA states that can be reached from q_B^{aux} in a possibly multi-hop fashion, using a finite sequence of feasible symbols, so that once these states are reached, the robot can always remain in them as long as needed using the same symbol that allowed it to reach this state. Formally, let $\mathcal{D}_{q_B^{\text{aux}}}$ be a set that collects all NBA states q_B (i) that have a feasible self-loop, i.e., $\Sigma^{q_B,q_B} \neq \emptyset$ and (ii) for which there exists a finite and feasible word w, i.e., a finite sequence of feasible symbols, so that starting from q_B^{aux} a finite NBA run ρ_w (i.e., a finite sequence of NBA states) is incurred that ends in q_B and activates the

self-loop of q_B. In math, $\mathcal{D}_{q_B^{\text{aux}}}$ is defined as:

$$\mathcal{D}_{q_B^{\text{aux}}} = \{q_B \in \mathcal{Q}_B | (\Sigma^{q_B,q_B} \neq \emptyset) \wedge (\exists w \text{ s.t. } \rho_w = q_B^{\text{aux}} \dots \bar{q}_B q_B q_B)\}. \tag{9.3}$$

By definition of q_B^{aux}, we have that $q_B^{\text{aux}} \in \mathcal{D}_{q_B^{\text{aux}}}$.

Among all possible pairs of states in $\mathcal{D}_{q_B^{\text{aux}}}$, we examine which transitions, possibly multi-hop, can be enabled using feasible symbols, so that, once these states are reached, the robot can always remain in them forever using the same symbol that allowed it to reach this state. Formally, consider any two states $q_B, q_B' \in \mathcal{D}_{q_B^{\text{aux}}}$ (i) that are connected through a - possibly multi-hop - path in the NBA, and (ii) for which there exists a symbol, denoted by $\sigma^{q_B,q_B'}$, so that if it is repeated a finite number of times starting from q_B, the following finite run can be generated:

$$\rho = q_B q_B^1 \dots q_B^{K-1} q_B^K q_B^K, \tag{9.4}$$

where $q_B' = q_B^K$, for some finite $K > 0$. In (9.4), the run is defined so that (i) $q_B^k \neq q_B^{k+1}$, for all $k \in \{1, K-1\}$; (ii) $q_B^k \in \delta_B(q_B^k, \sigma^{q_B,q_B'})$ is not valid for all $\forall k \in \{1, \dots, K-1\}$, i.e., the robot cannot remain in any of the intermediate states (if any) that connect q_B to q_B' either because a feasible self-loop does not exist or because $\sigma^{q_B,q_B'}$ cannot activate this self-loop; and (iii) $q_B' \in \delta_B(q_B', \sigma^{q_B,q_B'})$ i.e., there is a feasible loop associated with q_B' that is activated by $\sigma^{q_B,q_B'}$. Due to (iii), the robot can remain in q_B' as long as $\sigma^{q_B,q_B'}$ is generated. The fact that the finite repetition of a *single* symbol needs to generate the run (9.4) precludes multi-hop transitions from q_B to q_B' that require the robot to jump from one region of interest to another one instantaneously as such transitions are not meaningful as discussed in Section 9.1; see also Fig. 9.4. Hereafter, we denote the - potentially multi-hop - transition incurred due to the run (9.4) by $q_B' \in \delta_B^m(q_B, \cdot)$.

Then, we construct the directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V} \subseteq \mathcal{Q}_B$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The set of nodes is defined so that $\mathcal{V} = \mathcal{D}_{q_B^{\text{aux}}}$ and the set of edges is defined so that $(q_B, q_B') \in \mathcal{E}$ if there exists a feasible symbol that incurs the run ρ_w defined in (9.4); see also Fig. 9.4.

Given the graph $\mathcal{G}$, we define the following distance metric.

Definition 9.2 (Distance Metric)**.** *Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be the directed graph that corresponds to NBA B. Then, we define the distance function $d : \mathcal{V} \times \mathcal{V} \to \mathbb{N}$ as follows*

$$d(q_B, q'_B) = \begin{cases} |SP_{q_B,q'_B}|, \text{ if } SP_{q_B,q'_B} \text{ exists,} \\ \infty, \qquad \text{otherwise,} \end{cases} \tag{9.5}$$

where SP_{q_B,q'_B} denotes the shortest path (in terms of hops) in $\mathcal{G}$ from q_B to q'_B and $|SP_{q_B,q'_B}|$ stands for its cost (number of hops).

In words, $d : \mathcal{V} \times \mathcal{V} \to \mathbb{N}$ returns the minimum number of edges in the graph $\mathcal{G}$ that are required to reach a state $q'_B \in \mathcal{V}$ starting from a state $q_B \in \mathcal{V}$. This metric can be computed using available shortest path algorithms, such the Dijkstra method with worst-case complexity $O(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$.

Next, we define the final/accepting edges in $\mathcal{G}$ as follows.

Definition 9.3 (Final/Accepting Edges)**.** *An edge $(q_B, q'_B) \in \mathcal{E}$ is called final or accepting if the corresponding multi-hop NBA transition $q'_B \in \delta_B^m(q_B, \cdot)$ includes at least one final state $q_F \in \mathcal{Q}_F$.*

Based on the definition of accepting edges, we define the set $\mathcal{V}_F \subseteq \mathcal{V}$ that collects all states $q_B \in \mathcal{V}$ from which an accepting edge originates, i.e.,

$$\mathcal{V}_F = \{q_B \in \mathcal{V} \mid \exists \text{ accepting edge } (q_B, q'_B) \in \mathcal{E}\}. \tag{9.6}$$

By definition of the accepting condition of the NBA, we have that if at least one of the accepting edges is traversed infinitely often, then the corresponding LTL formula is satisfied.

Similar to [25], we define the distance of any state $q_B \in \mathcal{V}$ to the set $\mathcal{V}_F \subseteq \mathcal{V}$ as

$$d_F(q_B, \mathcal{V}_F) = \min_{q'_B \in \mathcal{V}_F} d(q_B, q'_B), \tag{9.7}$$

where $d(q_B, q'_B)$ is defined in (9.5) and $\mathcal{V}_F$ is defined in (9.6); see also Fig. 9.4.

9.2.3 Online Symbolic Controller

In this Section, we present how manipulation commands are generated online. The main idea can be summarized as follows. Once the continuous-time controller accomplishes the assigned sub-task, a new target automaton state is selected and a new manipulation command is generated. In case the continuous-time controller fails to accomplish the assigned sub-task (because e.g., a target region of interest is surrounded by fixed obstacles), the symbolic controller checks if there exists an alternative command that ensures reachability of the target automaton state (e.g., consider a case where a given target NBA state can be reached if the robot goes to either region ℓ_1 or ℓ_2). If there are no alternative commands to reach the desired automaton state, then a new target automaton state is selected that also decreases the distance to satisfying the accepting NBA condition. If there are no other automaton states that can decrease this distance, a message is returned stating that the robot cannot accomplish the assigned mission.

More specifically, the proposed controller requires as an input the graph $\mathcal{G}$ defined in Section 9.2.2, and selects NBA states that the robot should visit next so that the distance to the final states, as per (9.7), decreases over time. Namely, let $q_B(t) \in \mathcal{V}$ be the NBA state that the robot has reached after navigating the unknown environment for t time units. At time $t = 0$, $q_B(t)$ is selected to be the initial NBA state. Given the current NBA state $q_B(t)$, the robot selects a new NBA state, denoted by $q_B^{\text{next}} \in \mathcal{V}$ that it should reach next to make progress towards accomplishing their task. This state is selected among the neighbors of $q_B(t)$ in the graph $\mathcal{G}$ based on the following two cases. If $q_B(t) \notin \mathcal{V}_F$, where $\mathcal{V}_F$ is defined in (9.6), then among all neighboring nodes, we select one that satisfies

$$d_F(q_B^{\text{next}}, \mathcal{V}_F) = d_F(q_B(t), \mathcal{V}_F) - 1, \tag{9.8}$$

i.e., a state that is one hop closer to the set $\mathcal{V}_F$ than $q_B(t)$ is where d_F is defined in (9.7). Under this policy of selecting q_B^{next}, we have that eventually $q_B(t) \in \mathcal{V}_F$; controlling the robot to ensure this property is discussed in Section 9.4. If $q_B(t) \in \mathcal{V}_F$, then the state q_B^{next}

is selected so that $(q_B(t), q_B^{\text{next}})$ is an accepting edge as per Definition 9.3. This way we ensure that accepting edges are traversed infinitely often and, therefore, the assigned LTL task is satisfied.

Given the selected state q_B^{next}, a feasible symbol is selected that can enable the transition from $q_B(t)$ to q_B^{next}, i.e., can incur the run (9.4). By definition of the run in (9.4), it suffices to select a symbol that satisfies the following Boolean formula:

$$b^{q_B,q'_B} = b^{q_B,q_B^1} \wedge b^{q_B^2,q_B^3} \wedge \dots b^{q_B^{K-1},q_B^K} \wedge b^{q_B^K,q_B^K}, \tag{9.9}$$

where $q_B^K = q_B^{\text{next}}$. In words, the Boolean formula in (9.9) is the conjunction of all Boolean formulas $b^{q_B^{k-1},q_B^k}$ that need to be satisfied simultaneously to reach $q_B^{\text{next}} = q_B^K$ through a multi-hop path. Once such a symbol is generated, a point-to-point navigation and manipulation command is accordingly generated. For instance, if this symbol is $\pi^{a_k(M_i,\ell_j)}$ then the robot has to apply the action $a_k(M_i, \ell_j)$, i.e., go to a known region of interest ℓ_j and apply action a_k to the movable object M_i. The online implementation of such action is discussed in Section 9.4.

9.2.4 Completeness of the Symbolic Controller

Here, we provide conditions under which the proposed discrete controller is complete.

Proposition 9.1 (Completeness)**.** *Assume that there exists at least one infinite sequence of manipulation actions in the set $\mathcal{A}$ that satisfies ϕ. If the environmental structure and the continuous-time controller always ensure that at least one of the candidate next NBA states can be reached, then the proposed discrete algorithm is complete, i.e., a feasible solution will be found.*[3]

Proof. Included in Appendix C.7. □

[3]Given the current NBA state, denoted by $q_B(t)$, the symbolic controller selects as the next NBA state, a state that is reachable from $q_B(t)$ and closer to the final states as per the proposed distance metric. All NBA states that satisfy this condition are called candidate next NBA states. Also, reaching an NBA state means that at least one of the manipulation actions required to enable the transition from $q_B(t)$ to the next NBA state is feasible.

Note that the graph $\mathcal{G}$ is agnostic to the structure of the environment, meaning that an edge in $\mathcal{G}$ may not be able to be traversed. For instance, consider an edge in this graph that is enabled only if the robot applies a certain action to a movable object that is in a region blocked by fixed obstacles; in this case the continuous-time controller will not be able to execute this action due to the environmental structure. Satisfaction of the second assumption in Proposition 9.1 implies that if such scenarios never happen, (e.g., all regions and objects that the robot needs to interact with are accessible and the continuous-time controller allows the robot to reach them) then the proposed hybrid control method will satisfy the assigned LTL task if this formula is feasible. However, if the second assumption does not hold, there may be an alternative sequence of automaton states to follow in order to satisfy the LTL formula that the proposed algorithm failed to find due to the à-priori unknown structure of the environment.

9.3 Interface Layer Between the Symbolic and the Reactive Controller

We assume that the robot is nominally in an *LTL mode*, where it executes sequentially the commands provided by the symbolic controller described in Section 9.2. We use an *interface layer* between the symbolic controller and the reactive motion planner, as shown in Fig. 9.2, to translate each action to an appropriate gripper command ($g = 0$ for MOVE and GRASPOBJECT, and $g = 1$ for RELEASEOBJECT), and a navigation command toward a target $\mathbf{x}_d$. If the provided action is MOVE(ℓ_j) or RELEASEOBJECT(M_i, ℓ_j), we pick as $\mathbf{x}_d$ the centroid of region ℓ_j. If the action is GRASPOBJECT(M_i), we pick as $\mathbf{x}_d$ a collision-free location on the boundary of object M_i, contained in the freespace $\mathcal{F}$.

Consider again the example shown in Fig. 9.1. The first step of the assembly requires the robot to move object M_1 to ℓ_2 which, however, is occupied by the object M_2. In this case, instead of reporting that the assigned LTL formula cannot be satisfied, we allow the robot to temporarily pause the command execution from the symbolic controller and switch to a *Fix mode* and push object M_2 away from ℓ_1, before resuming the execution of the action

instructed by the symbolic controller. For plan fixing purposes, we introduce a fourth action, DisassembleObject($M_i, \mathbf{x}^*$), invisible to the symbolic controller, instructing the robot to push the object M_i (after it has been grasped using GraspObject) toward a position $\mathbf{x}^*$ on the boundary of the freespace until specific separation conditions are satisfied. Hence, an additional responsibility of the interface layer (when in Fix mode) is to pick the next object to be grasped and disassembled from a stack of blocking movable objects $\mathcal{B}_{\mathcal{M}}$, as well as the target $\mathbf{x}_d$ of each DisassembleObject action, until the stack $\mathcal{B}_{\mathcal{M}}$ becomes empty[4]; see Section 9.4.3.

Finally, the interface layer (a) requests a new action from the symbolic controller, if the robot successfully converges to $\mathbf{x}_d$ to complete the current action execution, or (b) reports that the currently executed action $a_k(M_i, \ell_j)$ (associated with a movable object M_i and a region of interest ℓ_j) is infeasible and requests an alternative action, if the topology checking module outlined in Section 9.4.2 determines that the goal $\mathbf{x}_d$ is surrounded by fixed obstacles.

9.4 Symbolic Action Implementation

In this Section, we describe the online implementation of our symbolic actions, assuming that the robot has already picked a target $\mathbf{x}_d$ using the interface layer from Section 9.3. As reported above, in the LTL mode, the robot executes commands from the symbolic controller, using one of the actions Move, GraspObject and ReleaseObject. The robot exits the LTL mode and enters the Fix mode when one or more movable objects block the target destination $\mathbf{x}_d$; in this mode, it attempts to rearrange blocking movable objects using a sequence of the actions GraspObject and DisassembleObject, before returning to the LTL mode.

The "backbone" of the symbolic action implementation is the reactive, vector field motion planner from Chapter 8, allowing either a fully actuated or a differential-drive robot to provably converge to a designated fixed target while avoiding all obstacles in the environment. When the robot is gripping an object i, we use the method from Chapter 4 for generating

[4]The exclusion of the negation operator from the LTL syntax, as assumed in Section 9.1, guarantees that each DisassembleObject action will *not* interfere with the satisfaction of the LTL formula ϕ, e.g., the robot will not disassemble an object that should not be grasped or moved.

virtual commands for the center $\mathbf{x}_{i,c}$ of the circumscribed disk with radius $(\rho_i + r)$, enclosing the robot and the object. Namely, we assume that the robot-object pair is a fully actuated particle with dynamics $\dot{\mathbf{x}}_{i,c} = \mathbf{u}_{i,c}(\mathbf{x}_{i,c})$, design our control policy $\mathbf{u}_{i,c}$ using the same vector field controller, and translate to commands $\overline{\mathbf{u}} = (v, \omega)$ for our differential drive robot as $\overline{\mathbf{u}} := \mathbf{T}_{i,c}(\psi)^{-1}\, \mathbf{u}_{i,c}$, with $\mathbf{T}_{i,c}(\psi)$ the Jacobian of the gripping contact, i.e., $\dot{\mathbf{x}}_{i,c} = \mathbf{T}_{i,c}(\psi)\, \overline{\mathbf{u}}$.

This reactive controller assumes that a path to the goal always exists (i.e., the robot's freespace is path-connected), and does not consider cases where the target is blocked either by a fixed obstacle or a movable object[5]. Hence, here, after including a brief overview of the reactive, vector field motion planner from Chapter 8 (Section 9.4.1), we extend the algorithm's capabilities by providing a topology checking algorithm (Section 9.4.2) that detects blocking movable objects or fixed obstacles, as outlined in Fig. 9.2. Based on these capabilities, we finally describe our symbolic action implementations (Section 9.4.3).

9.4.1 Reactive Controller Overview

As described in Chapter 8 and shown in Fig. 9.5, the robot navigates the physical space and discovers obstacles (e.g., using the semantic mapping engine in [30]), which are dilated by the robot radius and stored in the semantic space. Potentially overlapping obstacles in the semantic space are subsequently consolidated in real time to form the mapped space. A change of coordinates $\mathbf{h}$ from this space is then employed to construct a geometrically simplified (but topologically equivalent) model space, by merging familiar obstacles overlapping with the boundary of the enclosing freespace to this boundary, deforming other familiar obstacles to disks, and leaving unknown obstacles intact. As shown in Chapter 8, the constructed change of coordinates $\mathbf{h}^{\mathcal{I}}$ between the mapped and the model space, for a given index set $\mathcal{I}$ of instantiated familiar obstacles, is a C^∞ diffeomorphism away from sharp corners. Using the diffeomorphism $\mathbf{h}^{\mathcal{I}}$, we construct a hybrid vector field controller (with the modes indexed by $\mathcal{I}$, i.e., depending on external perceptual updates), that guarantees simultaneous obstacle avoidance and target convergence, while respecting input command

[5]The possibility of an entirely unknown blocking convex obstacle is precluded by our separation assumptions in Section 9.1.

limits, in unexplored semantic environments (see (8.1)).

9.4.2 Topology Checking Algorithm

The topology checking algorithm is used to detect freespace disconnections, update the robot's enclosing freespace $\mathcal{F}_e$, and modify its action by switching to the Fix mode, if necessary. In summary, the algorithm's input is the initially assumed polygonal enclosing freespace $\mathcal{F}_e$ for either the robot or the robot-object pair, along with all known dilated movable objects in $\mathcal{M}$ and fixed obstacles in $\mathcal{P}_\mathcal{I}$ (corresponding to the index set $\mathcal{I}$ of localized familiar obstacles). The algorithm's output is the detected enclosing freespace $\mathcal{F}_e$, used for the diffeomorphism construction in the reactive controller (Chapter 8), along with a stack of *blocking movable objects* $\mathcal{B}_\mathcal{M}$ and a Boolean indication of whether the current symbolic action is feasible. Based on this output, the robot switches to the Fix mode when the stack $\mathcal{B}_\mathcal{M}$ becomes non-empty, and resumes execution from the symbolic controller once all movable objects in $\mathcal{B}_\mathcal{M}$ are disassembled.

More specifically, this algorithm works as follows (see Algorithm 9.1). Starting with the initially assumed polygonal enclosing freespace $\mathcal{F}_e$ for either the robot or the robot-object pair, we subtract the union of all known dilated movable objects in $\mathcal{M}$ and fixed obstacles in $\mathcal{P}_\mathcal{I}$ (corresponding to the index set $\mathcal{I}$ of localized familiar obstacles), using standard logic operations with polygons (see e.g., [41, 53, 55]). This operation results in a list of freespace components, which we denote by $\mathcal{L}_\mathcal{F} := (\mathcal{F}_1, \mathcal{F}_2, \ldots)$. From this list, we identify the freespace $\mathcal{F}$ as the freespace component $\mathcal{F}_k$ that contains the robot position $\mathbf{x}$ (or the robot-object pair center $\mathbf{x}_{i,c}$) and re-define the enclosing freespace as its convex hull, i.e., $\mathcal{F}_e := \text{Conv}\left(\overline{\mathcal{F}}_k\right)$.

If the goal $\mathbf{x}_d$ is contained in $\mathcal{F}_e$, the reactive controller proceeds as usual, using $\mathcal{F}_e$ for the diffeomorphism construction (see Section 9.4.1), and treating all other freespace components as obstacles. Otherwise, we need to check whether movable objects or fixed obstacles cause a freespace disconnection that does not allow for successful action completion. Namely, we need to check whether both the robot position $\mathbf{x}$ (or the robot-object pair center $\mathbf{x}_{i,c}$) and the target $\mathbf{x}_d$ are included in the same connected component of the set

$\mathcal{L}_{\mathcal{F}+\mathcal{M}} := (\bigcup_i \mathcal{F}_i) \cup \left(\bigcup_j M_j\right)$, i.e., the union of all freespace components in $\mathcal{L}_{\mathcal{F}}$ with all dilated movable objects in $\mathcal{M}$. This would imply that a subset of movable objects in $\mathcal{M}$ blocks the target configuration. In that case, the robot switches to the Fix mode to rearrange these objects; otherwise, the interface layer reports to the symbolic controller that the current action is infeasible.

In the former case, we proceed one step further to identify the blocking movable objects in order to reconfigure them on-the-fly. First, we isolate the connected components of the union of all movable objects in $\mathcal{M}$ into a list $\mathcal{L}_{\mathcal{M}} := (\mathcal{M}_1, \mathcal{M}_2, \ldots)$; we refer to the elements of that list as the *movable object clusters*. Assuming that each movable object cluster is connected to at most two freespace components from $\mathcal{L}_{\mathcal{F}}$, we build a connectivity tree rooted at the robot's (or the robot-object pair's) freespace $\mathcal{F}$, by checking whether the closures of two individual regions overlap; the tree's vertices are geometric regions (freespace components in $\mathcal{L}_{\mathcal{F}}$ and movable object clusters in $\mathcal{L}_{\mathcal{M}}$) and edges denote adjacency. We then backtrack from the vertex of the tree that contains the goal $\mathbf{x}_d$ until we reach the root, saving the encountered movable object clusters along the way. Any movable object intersecting any of these clusters is pushed to a stack of *blocking movable objects* $\mathcal{B}_{\mathcal{M}}$, that the robot needs to disassemble. An algorithmic overview of the method is included in Algorithm 9.1.

9.4.3 Action Implementation

We are now ready to describe the used symbolic actions. The symbolic action Move(ℓ_j) simply uses the reactive controller to navigate to the selected target $\mathbf{x}_d$, as described in Section 9.4.1. Similarly, the symbolic action GraspObject(M_i) uses the reactive controller to navigate to a collision-free location on the boundary of object M_i, and then aligns the robot so that its gripper faces the object, in order to get around Brockett's condition [34]. ReleaseObject(M_i, ℓ_j) uses the reactive controller to design inputs for the robot-object center $\mathbf{x}_{i,c}$ and translates them to differential drive commands through the center's Jacobian $\mathbf{T}_{i,c}(\psi)$, in order to converge to the goal $\mathbf{x}_d$.

Finally, the action DisassembleObject($M_i, \mathbf{x}_d$) is identical to ReleaseObject, with two important differences. First, we heuristically select as target $\mathbf{x}_d$ the middle point of the

Algorithm 9.1 Topology Checking Algorithm.

function TopologyChecking($\mathbf{x}$,$\mathbf{x}_d$,$\mathcal{F}_e$,$\mathcal{M}$,$\mathcal{P}_\mathcal{I}$)
  $\mathcal{L}_\mathcal{F} \leftarrow$ `Subtract`$(\mathcal{F}_e,$ `Union`$(\mathcal{M}, \mathcal{P}_\mathcal{I}))$
  for $\mathcal{F}_k \in \mathcal{L}_\mathcal{F}$ **do**
    if $\mathbf{x} \in \mathcal{F}_k$ **then**
      $\mathcal{F} \leftarrow \mathcal{F}_k$
      $\mathcal{F}_e \leftarrow \text{Conv}(\overline{\mathcal{F}}_k)$
      break
    end if
  end for
  if $\mathbf{x}_d \in \mathcal{F}$ **then**
    $\mathcal{B}_\mathcal{M} \leftarrow \varnothing$ ▷ No blocking objects or obstacles
    `IsFeasible` $\leftarrow$ `True` ▷ Task feasible
  else
    $\mathcal{L}_{\mathcal{F}+\mathcal{M}} \leftarrow (\bigcup_i \mathcal{F}_i) \cup (\bigcup_j M_j), \mathcal{F}_i \in \mathcal{L}_\mathcal{F}, M_j \in \mathcal{M}$
    for $\mathcal{F}_k \in \mathcal{L}_{\mathcal{F}+\mathcal{M}}$ **do**
      if $\mathbf{x} \in \mathcal{F}_k$ **then**
        if $\mathbf{x}_d \in \mathcal{F}_k$ **then**
          $\mathcal{L}_\mathcal{M} \leftarrow \bigcup_j M_j, M_j \in \mathcal{M}$
          $(\mathcal{V}_\mathcal{L}, \mathcal{E}_\mathcal{L}) \leftarrow$ `ConnectTree`$(\mathcal{L}_\mathcal{F}, \mathcal{L}_\mathcal{M})$
          for $V \in \mathcal{V}_\mathcal{L}$ **do**
            if $\mathbf{x}_d \in V$ **then**
              $\mathcal{B}_\mathcal{M} \leftarrow$ `BacktrackFrom`(V)
              break
            end if
          end for
          `IsFeasible` $\leftarrow$ `True`
        else
          $\mathcal{B}_\mathcal{M} \leftarrow \varnothing$ ▷ Blocked by fixed obstacles
          `IsFeasible` $\leftarrow$ `False`
        end if
        break
      end if
    end for
  end if
  return $\mathcal{F}_e$, `IsFeasible`, $\mathcal{B}_\mathcal{M}$
end function

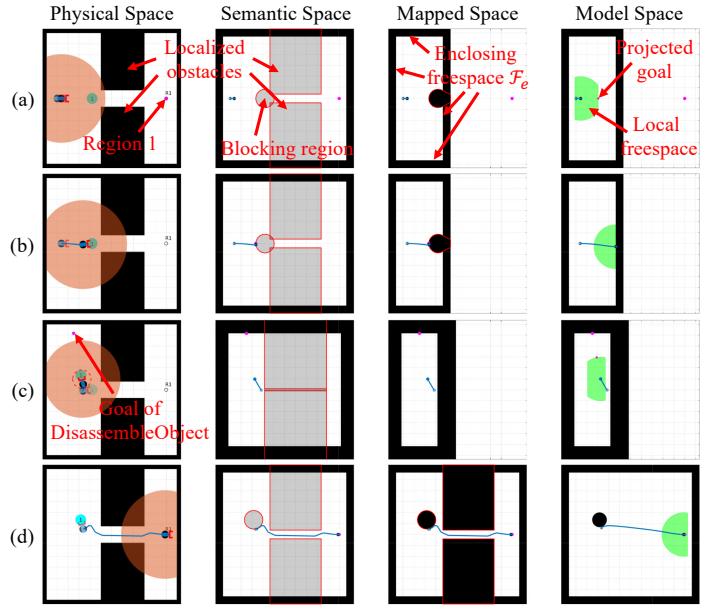

Figure 9.5: Demonstration of local LTL plan fixing, where the task is to navigate to region 1, captured by the LTL formula $\phi = \Diamond \pi^{a_1(\varnothing,\ell_1)}$ where ℓ_1 refers to region 1 in the figure. (a) The robot starts navigating to its target, until it localizes the two rectangular obstacles and recognizes that the only path to the goal is blocked by a movable object. (b) The robot switches to the Fix mode, grips the object, and (c) moves it away from the blocking region, until the separation assumptions outlined in Section 9.4.3 are satisfied. (d) It then proceeds to complete the task.

edge of the polygonal freespace $\mathcal{F}$ that maximizes the distance to all other movable objects (except M_i) and all regions of interest ℓ_j. Second, in order to accelerate performance and shorten the resulting trajectories, we stop the action's execution if the robot-object pair, centered at $\mathbf{x}_{i,c}$ does not intersect any region of interest and the distance of $\mathbf{x}_{i,c}$ from all other objects in the workspace is at least $2(r + \max_{k \in \mathcal{B}_\mathcal{M}} \rho_k)$, as this would imply that dropping the object in its current location would not block a next step of the disassembly process. Even though we do not yet report on formal results pertaining to the task sequence in the Fix mode, the DisassembleObject action maintains formal results of obstacle avoidance and target convergence to a feasible $\mathbf{x}_d$, using our reactive, vector field controller.

9.5 Illustrative Simulations

In this Section, we implement simulated examples of different tasks in various environments using our architecture, shown in Fig. 9.2. All simulations were run in MATLAB using

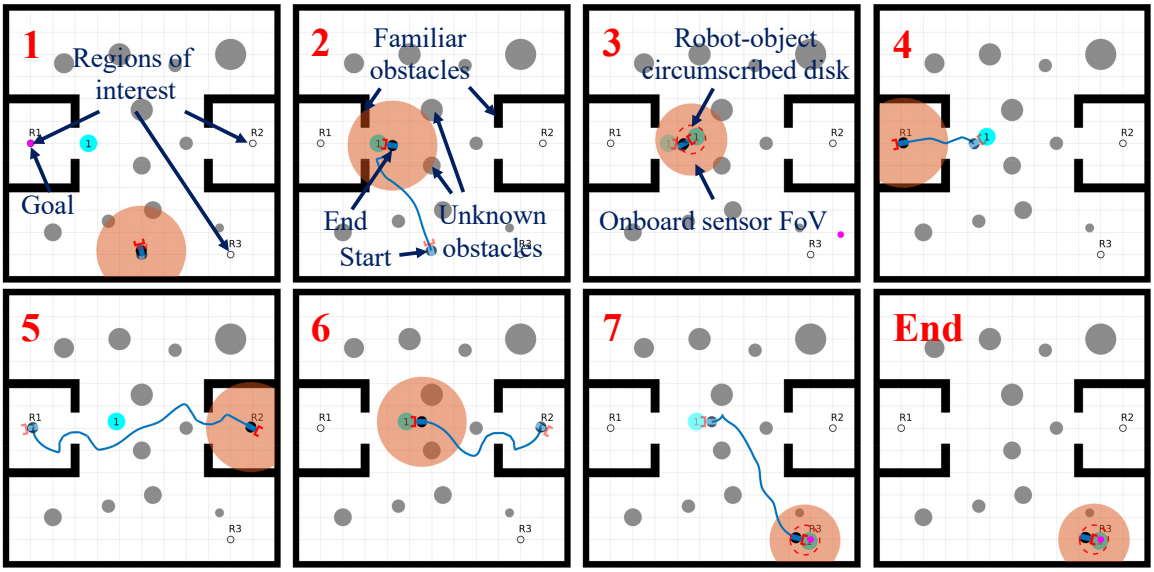

Figure 9.6: Executing the LTL formula $\phi = \Diamond(\pi^{a_1(\varnothing,\ell_1)} \wedge \Diamond(\pi^{a_1(\varnothing,\ell_2)} \wedge \Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond\pi^{a_3(M_1,\ell_3)})))$ in an environment cluttered with known walls (black) and unknown convex obstacles (grey).

`ode45`, leveraging and enhancing the presentation infrastructure from Chapters 7 - 8[6]. The discrete controller and the interface layer are implemented in MATLAB, whereas the reactive controller is implemented in Python and communicates with MATLAB using the standard MATLAB-Python interface. For our numerical results, we assume perfect robot state estimation and localization of obstacles using the onboard sensor, which can instantly identify and localize either the entirety of familiar obstacles or fragments of unknown obstacles within its range. The reader is referred to the accompanying video submission of [206][7] for visual context and additional simulations.

9.5.1 Demonstration of Local LTL Plan Fixing

Fig. 9.5 includes a demonstration of a simple task, encoded in the LTL formula $\phi = \Diamond\pi^{a_1(\varnothing,\ell_1)}$, i.e., eventually execute the action MOVE to navigate to region 1, demonstrating how the Fix mode for local rearrangement of blocking movable objects works.

[6] See `https://github.com/KodlabPenn/semnav_matlab`.
[7] `https://youtu.be/grypNPM1zo4`

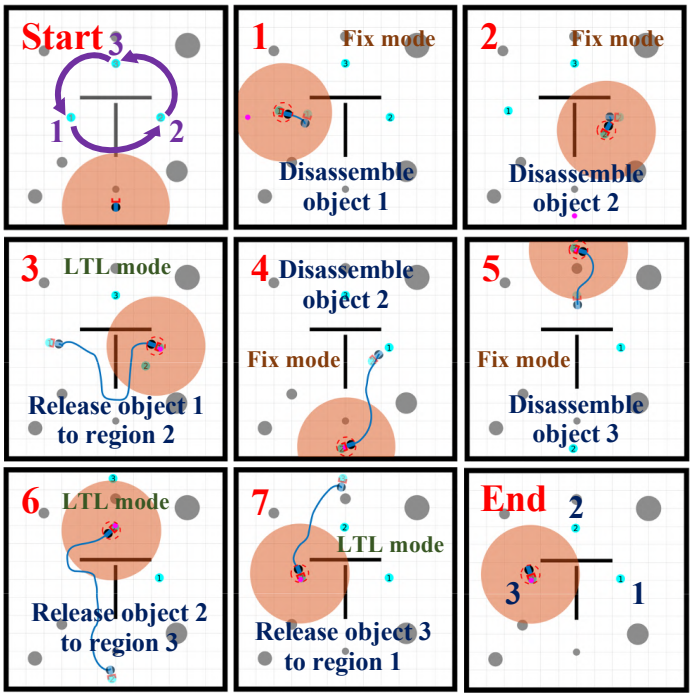

Figure 9.7: An illustrative execution of the problem depicted in Fig. 9.1. The task is specified by the LTL formula (9.1) requires the counterclockwise rearrangement of 3 objects in an environment cluttered with some unanticipated familiar (initially dark grey and then black upon localization) and some completely unknown (light grey) fixed obstacles.

9.5.2 Executing More Complex LTL Tasks

Fig. 9.6 includes successive snapshots of a more complicated LTL task, captured by the formula

$$\phi = \Diamond(\pi^{a_1(\varnothing,\ell_1)} \wedge \Diamond(\pi^{a_1(\varnothing,\ell_2)} \wedge \Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond\pi^{a_3(M_1,\ell_3)})))$$

which instructs the robot to first navigate to region 1, then navigate to region 2, and finally grasp object 1 and move it to region 3, in an environment cluttered with both familiar non-convex and completely unknown convex obstacles. Before navigating to region 1, the robot correctly identifies that the movable object disconnects its freespace and proceeds to disassemble it. After visiting region 2, it then revisits the movable object, grasps it and moves it to the designated location to complete the required task. The reader is referred to the video submission of [206][7] for visual context regarding the evolution of all planning spaces (semantic, mapped and model space) during the execution of this task, as well as several other simulations with more movable objects, including (among others) a task where

the robot needs to patrol between some predefined regions of interest in an environment cluttered with obstacles by visiting each one of them infinitely often.

9.5.3 Execution of Rearrangement Tasks

Finally, a promising application of our reactive architecture concerns rearrangement planning with multiple movable pieces. Traditionally, such tasks are executed using sampling-based planners, whose offline search times can blow up exponentially with the number of movable pieces in the environment (see, e.g., [209, Table I]). Instead, as shown in Fig. 9.7, the persistent nature of our reactive architecture succeeds in achieving the given task online in an environment with multiple obstacles, even though our approach might require more steps and longer trajectories in the overall assembly process than other optimal algorithms [210]. Moreover, the LTL formulas for encoding such tasks are quite simple to write (see (9.1) for the example in Fig. 9.7), instructing the robot to grasp and release each object in sequence; the reactive controller is capable of handling obstacles and blocking objects during execution time. The accompanying video submission of [206][7] includes a rearrangement example with 4 movable objects, requiring more steps in the assembly process.

Chapter 10

Conclusion and Ideas for Future Work

This concluding Chapter presents a summary of observations about the reported work and discusses possible future directions.

10.1 Conclusion

This research suggests with formal arguments and empirical demonstration the effectiveness of a hierarchical control structure for highly dynamic physical systems in mobile manipulation settings, shown in Fig. 1.1. We believe this is the first provably correct deliberative/reactive planner to engage an unmodified general purpose mobile manipulator in physical rearrangements of its environment, by moving objects with size comparable to the robot's size among unanticipated conditions and obstacles (Chapters 4 - 5). To this end, we have developed the mobile manipulation maneuvers to accomplish each task at hand (Chapter 5), successfully anchored the useful kinematic unicycle template to control the highly dynamic Minitaur robot (Chapter 3) and integrated perceptual feedback with low-level control to coordinate the robot's movement (Chapter 5).

At the same time, this research exploits recent developments in semantic SLAM [30] and object pose and triangular mesh extraction using convolutional neural net architectures [93, 106, 148] to provide an avenue for incorporating partial prior knowledge within a deterministic framework well suited to existing vector field planning methods [7]. The developed

algorithms guarantee collision avoidance and convergence to the designated goal for both a differential drive robot and a differential drive robot gripping and manipulating objects, in a workspace cluttered with completely unknown convex obstacles (Chapters 3 - 4), completely unknown non-convex obstacles (Chapter 5) that obey specific "length-scale" geometric assumptions [152], or "familiar", online recognizable non-convex obstacles (Chapters 6 - 8). Based on these capabilities, we build an interface between the developed reactive schemes and an abstract temporal logic engine [92] for addressing logically complex tasks, and reduce the overall offline deliberative planning time by greedily rearranging the workspace during execution time when a given sub-task is not feasible (Chapter 9).

10.2 Proposed Future Work

In the following, we present research currently underway or propose ideas for future work, that could significantly enhance the hierarchical architecture of Fig. 1.1.

10.2.1 Deliberative Layer

Ongoing research seeks to expand the mobile manipulation work with Minitaur, presented in Part II, and address more complex tasks in 2.5D environments (planar workspaces cluttered with "platforms" of discrete height values that the robot can exploit). As a particular example, we consider tasks where a quadrupedal robot (such as Minitaur) is trapped in a cluttered area and, in order to escape its confines, it must rearrange the environment so that it can execute a sequence of highly dynamic jumping maneuvers, such as climbing the surrounding "clutter" or jumping across gaps. Finding a reasonable plan in this domain is quite challenging (even when assuming a deterministic robot in a fully observable world) since the robot must consider the pose of each movable object, and the ways in which that pose can affect the problem solution. That problem is exacerbated when we abandon the planar workspaces considered in this thesis and instead focus on 2.5D environments, as this introduces additional challenges related to the description of implicit geometric constraints used for collision detection during the deliberative search, and makes the problem combinatorially hard by significantly increasing the number of required samples. However, following

the architecture outlined in Fig. 1.1, we intend to show that such problems can be solved efficiently, by letting the deliberative planner focus on just the high-level task planning problem, since the reactive planner can guarantee target convergence to any point in the robot's connected component of the configuration space, using the reactive layer for local obstacle avoidance with sensor feedback, and extending the gait layer to accommodate more complex pedipulation maneuvers [191].

Initial results demonstrating a simulation example of Minitaur successfully executing such a complex task are included in Fig. 10.1. The robot is tasked with moving to a predefined position on top of a table, which is initially unreachable; to reach its target, the robot needs to first grab and move the circular table to a suitable position and then perform a sequence of dynamic jumping maneuvers that also exploit other immovable objects in the environment. Following Fig. 1.1, the deliberative layer reasons about the known movable objects in the environment (round table, rectangular tables, boxes), and the sequence of actions that achieve the desired task and connect the disconnected configuration space. The reactive layer uses sensory feedback from the onboard camera and LIDAR to make sure that the robot avoids unexpected obstacles in the workspace during the execution of each action, and carefully aligns the robot with the objects before the execution of each dynamic maneuver using AprilTags [214]. Finally, the gait layer receives the commands from the reactive layer (such as distance and bearing to objects that the robot needs to grab, or desired forward velocity and yaw rate as outlined in Chapter 3) and executes the corresponding steady-state (Walk or Push-Walk, following Chapter 5) and dynamic maneuvers (Mount, Dismount, Jump [191]), using low-level feedback from the robot joints and IMUs.

10.2.2 Interface Layer

In Chapter 9, we propose a novel hybrid control architecture for achieving complex tasks with mobile manipulators in the presence of unanticipated obstacles and conditions, using an interface between an abstract temporal logic engine and a reactive controller for target convergence and collision avoidance. However, we have not yet provided any formal claims on

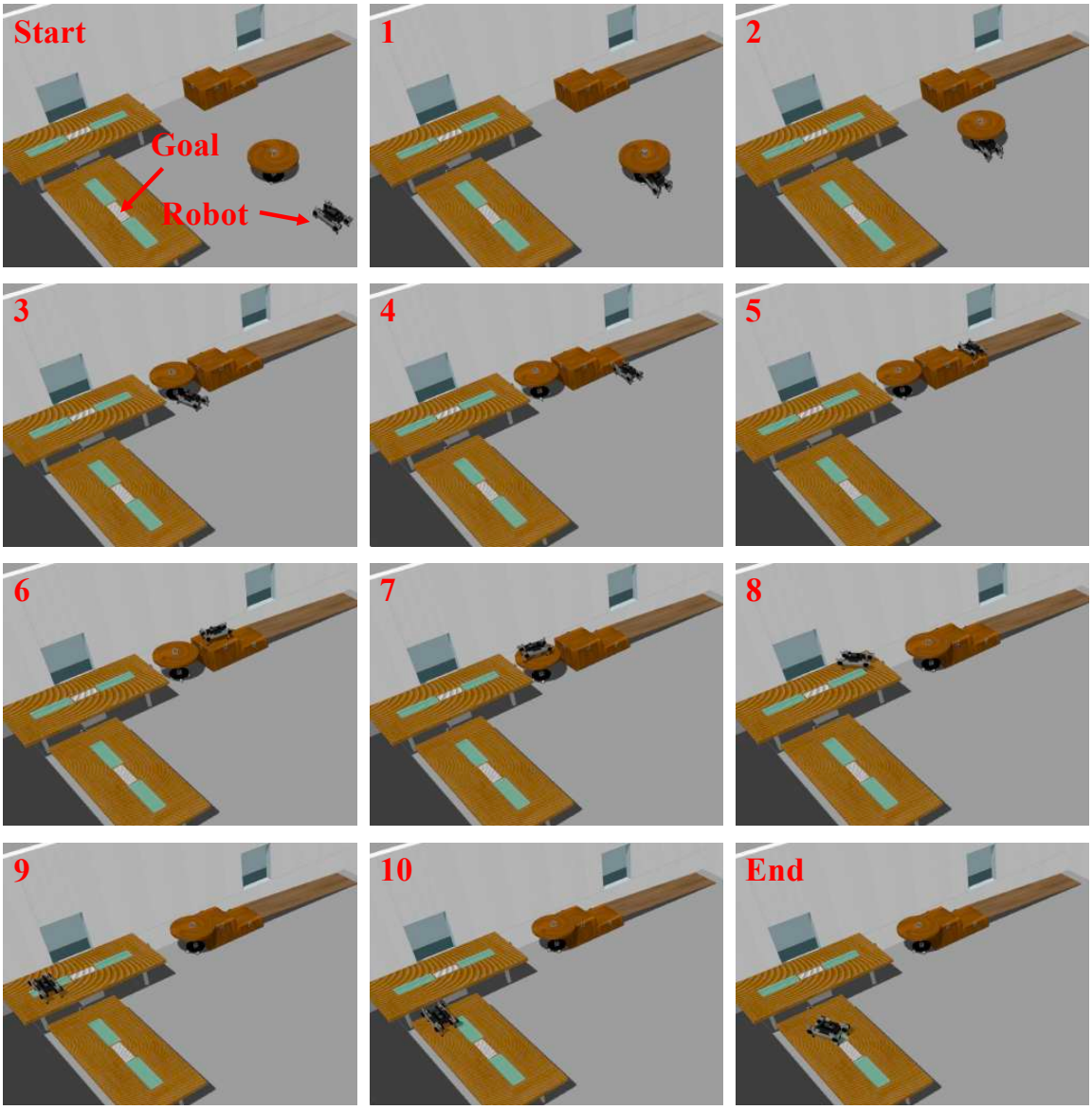

Figure 10.1: Simulation example in Gazebo, with Minitaur successfully manipulating and exploiting its environment with dynamic jumping and other pedipulation maneuvers [191] to reach its target.

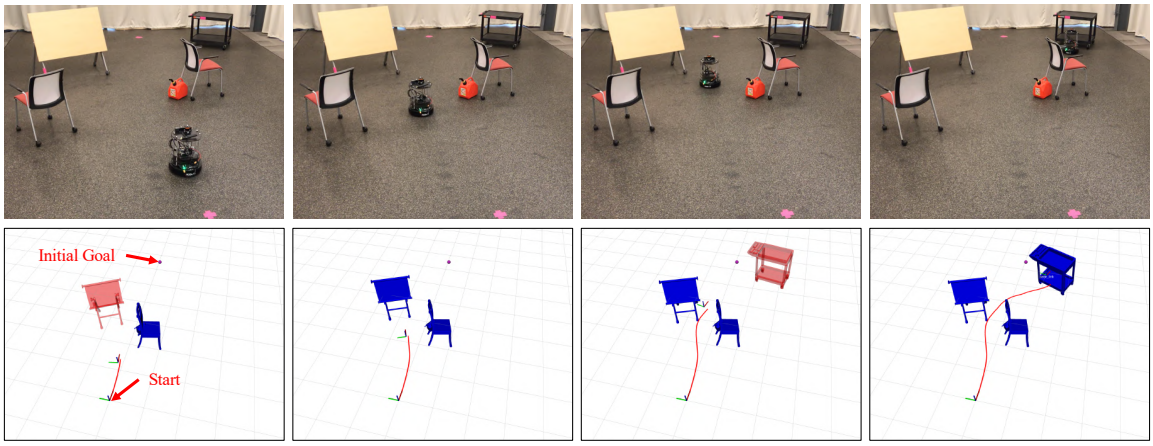

Figure 10.2: Navigation toward a semantic target with Turtlebot. The robot is initially tasked with moving to a predefined location, unless it detects and localizes a cart; in that case it has to approach and face the cart. The last column (Top: snapshot of the physical workspace, Bottom: illustration of the recorded trajectory in RViz) shows that the robot successfully executes the task.

that interface, and the local plan fixing, mobile manipulation vector field does not necessarily guarantee task completion (e.g., in tightly packed workspaces). Future work could focus on providing end-to-end correctness guarantees for the architecture shown in Fig. 9.2, using a more elaborate intermediate goal selection scheme for plan fixing purposes (see e.g., [10]) that assures task completion under some conservative assumptions about the environment, as well as extensions to multiple robots for collaborative manipulation tasks.

Figs. 10.2 and 10.3 present snapshots from experiments in settings falling outside the scope of the formal results in Chapter 7 that illustrate how we can use the perceptual infrastructure developed in Chapters 7 - 8 to further enhance the interface layer and make the robot capable of reasoning about its environment. In Fig. 10.2, we command the Turtlebot robot to move to a geometrically predefined target, unless it sees and localizes a cart; in that case, it is tasked with approaching and facing the cart with its camera. As shown in the bottom row of Fig. 10.2, the robot avoids familiar obstacles, localizes the cart and proceeds to properly approach it, with the right orientation. We take this approach one step further with the example shown in Fig. 10.3, using the Minitaur platform. Using the mobile manipulation primitives developed in [191], we task the robot by not only localizing and approaching the cart, but also jumping to grab and mount it. Therefore, future work

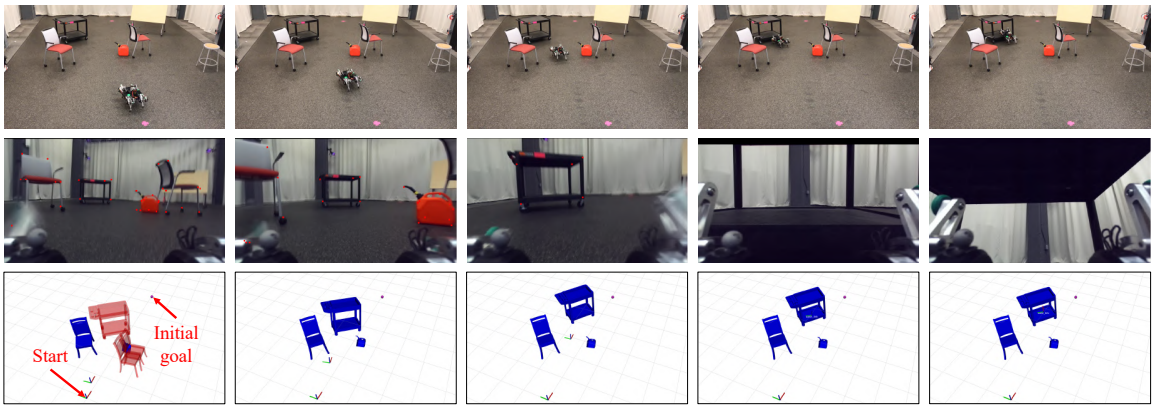

Figure 10.3: Using reactive navigation with mobile manipulation primitives on Minitaur. Similarly to Fig. 10.2, the robot is tasked with moving to a predefined location, unless it detects and localizes a cart; in that case it has to approach and jump to mount the cart, using a maneuver from [191]. Top: Recorded snapshots of the physical workspace, Middle: First-person view with semantic keypoints of familiar obstacles shown as red dots, Bottom: RViz illustration of the recorded semantic map.

could seek to develop a more elaborate interface layer that uses such perceptual schemes to reason about objects in the workspace. For example, one could develop a visuotactile algorithm that lets the robot not only observe a particular object that blocks a passage to its goal, but also use its limbs to feel it and decide whether it is movable; the robot could then deduce whether to declare failure to the task planner and re-plan, or simply grasp the object and push it to clear the path. Finally, we could consider scenarios where the interface layer decides whether the robot should switch from navigation to exploration and vice versa (using, e.g., motivation dynamics [161]), while preserving formal guarantees of target convergence and obstacle avoidance.

10.2.3 Reactive Layer

We believe the methods developed in Chapters 6 - 8 for generating in real time simple, topologically equivalent model spaces and pulling back the model controller through the corresponding diffeomorphism can be applied to diverse, philosophically alternative approaches to our purely reactive formulation of motion planning. For example, sampling-based (probabilistically complete) offline planners have been shown to benefit from integration with even geometrically naive locally reactive methods [11] that can mitigate difficulties such as finding paths through narrow passages. We imagine that even greater simplification of the

steering and collision-checking issues arising from sampling-based methods in partially "familiar" geometrically complicated environments [24, 119] might be achieved by shifting the problem of finding a feasible path to a topologically equivalent, metrically simple abstracted model wherein planning might be significantly faster. The robot could then be tasked to follow a generated path in the abstract space (e.g., along the lines of [8]) and the associated commands can be pulled back to the physical space through the diffeomorphism. Careful future inquiry will be needed to explore such deliberative-reactive hybrid uses for the online topological abstraction of familiar geometry developed here.

Moreover, future work could relax the required degree of partial knowledge and the separation assumptions needed for our formal results, by merging the "implicit representation trees" (e.g. see Fig. A.1) online, when needed. As a particular example, Chapters 7 - 8 use only an RGB camera to detect and reconstruct familiar obstacles in the environment, using deep learning. We could, however, imagine architectures that use more elaborate sensory schemes with 3D LIDARs; in this way, the robot would explore its workspace and use its 3D LIDAR to incrementally build an implicit representation of the surrounding (geometric) environment that would be deformed in real time to its topological model space, used for navigation during execution time.

In the longer term, we believe that concepts from the literature on convex decomposition of polyhedra [122] may afford a generalization beyond our present restriction to 2D workspaces toward the challenge of navigating partially known environments in higher dimension. Even though the currently presented algorithms would be restricted to shapes with genus zero (no holes), one could develop algorithms that "patch" the holes of shapes with non-zero genus when they are not important, and use the same principles for navigation. Separate research, but related to the problem of reactive planning for more interesting configuration spaces, could focus on robots with more complex and/or differentially constrained motion models, such as autonomous winged aircrafts.

Finally, research currently underway considers extensions of our reactive planning algorithms to (mildly) adversarial environments with moving obstacles and/or aggressively

moving targets, to allow for smoother integration of mobile robots in human-crowded environments. Although the objective of obstacle avoidance in such scenarios remains the same, we might need to depart from the notion of "target convergence" in the Lyapunov sense and come up with a different criterion, that allows the robot to simply track a given target in adversarial conditions, while avoiding getting trapped in unfavorable configurations.

10.2.4 Gait Layer

Throughout this work, we use the kinematic unicycle model as a well-behaved, steady-state template for legged locomotion on the horizontal plane (see Chapter 3), even though we have just offered an empirical anchoring algorithm for the Minitaur and Spirit robots. Future work could aim to make formal arguments about the anchoring of the kinematic unicycle model on different gaits or, alternatively, propose a different navigation template for legged robots (perhaps by fusing traditional templates, such as the first or second order differential drive model, with footstep planning) and accordingly modify the presented reactive navigation strategies for more efficient application in outdoor settings with challenging terrains. Future work could similarly focus on transitional maneuvers [191] and address the problem of more closely integrating the reactive and deliberative planners, while maintaining provable properties such as assured successful maneuver execution and tracking of template commands (e.g., desired speed, jump height) provided by the reactive layer.

Appendices

Appendix A

Computational Geometry Methods

A.1 Implicit Representation of Obstacles with R-functions

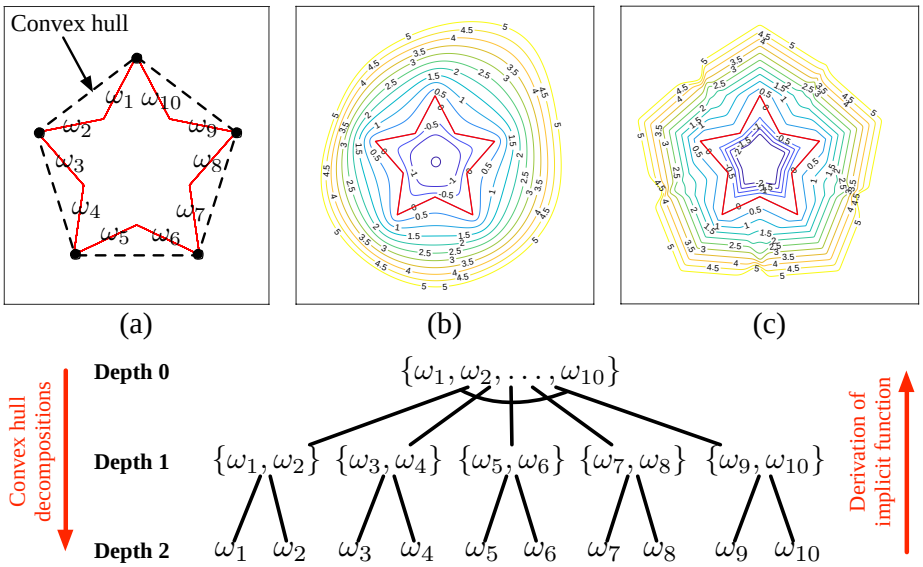

Figure A.1: Top: (a) An example of a polygonal obstacle and the corresponding ω_j functions, (b) Level curves of the corresponding implicit function β for $p = 2$, (c) Level curves of the corresponding implicit function β for $p = 20$, Bottom: The AND-OR tree, constructed by the algorithm described in Appendix A.1.2 to represent this polygon. The polygon is split at the vertices of the convex hull to generate five subchains at depth 1. Each of these subchains is then split into two subchains at depth 2. The subchains at depth 2 (1) are combined via disjunction (conjunction), since they meet at non-convex (convex) vertices of the original polygon. In this way, we get our implicit function $\beta = \neg\left((\omega_1 \vee \omega_2) \wedge (\omega_3 \vee \omega_4) \wedge (\omega_5 \vee \omega_6) \wedge (\omega_7 \vee \omega_8) \wedge (\omega_9 \vee \omega_{10})\right)$.

In this Appendix, we describe our method for implicit function representation of our memorized catalogue elements using R-function compositions [168], explored by Rimon [163] and explicated within the field of constructive solid geometry by Shapiro [176]. This modular representation of shape helps with the instantiation of the posited mapping oracle for obstacles with known geometry, whose mesh can be identified in real time using state-of-the-art techniques [93, 106, 148] in order to extract implicit function representations for polygonal obstacles.

A.1.1 Preliminary Definitions

We begin by providing a definition of an R-function [176].

Definition A.1. *A function $\gamma_\Phi : \mathbb{R}^n \to \mathbb{R}$ is an R-function if there exists a (binary) logic function $\Phi : \mathbb{B} \to \mathbb{B}$, called the companion function, that satisfies the relation*

$$\Phi(S_2(w_1), \ldots, S_2(w_n)) = S_2(\gamma_\Phi(w_1, \ldots, w_n)) \tag{A.1}$$

with $(w_1, \ldots, w_n) \in \mathbb{R}^n$ and S_2 the Heaviside characteristic function $S_2 : \mathbb{R} \to \mathbb{B}$ of the interval $[0+, \infty)$ defined as[1]

$$S_2(\chi) = \begin{cases} 0, & \chi \leq -0 \\ 1, & \chi \geq +0 \end{cases} \tag{A.2}$$

Informally, a real function γ_Φ is an R-function if it can change its property (sign) only when some of its arguments change the same property (sign) [176]. For example, the companion logic function for the R-function $\gamma(x, y) = xy$ is $X \Leftrightarrow Y$; we just check that $S_2(xy) = (S_2(x) \Leftrightarrow S_2(y))$.

[1] In [176], it is assumed that zero is always signed: either $+0$ or -0, which allows the authors to determine membership of zero either to the set of positive or to the set of negative numbers. This assumption is employed to resolve pathological cases, where the membership of zero causes R-function discontinuities and is not of particular importance in our setting.

In this work, we use the following (symbolically written) R-functions [176]

$$\neg x := -x \tag{A.3}$$

$$x_1 \wedge x_2 := x_1 + x_2 - (x_1^p + x_2^p)^{\frac{1}{p}} \tag{A.4}$$

$$x_1 \vee x_2 := x_1 + x_2 + (x_1^p + x_2^p)^{\frac{1}{p}} \tag{A.5}$$

with companion logic functions the logical negation $\neg$, conjunction $\wedge$ and disjunction $\vee$ respectively and p a positive integer. Intuitively, the author in [176] uses the triangle inequality with the L_p-norm to derive R-functions with specific properties.

A.1.2 Description of the Algorithm

R-functions have several interesting properties but, most importantly, provide machinery to construct implicit representations for sets built from other, primitive sets. Namely, in order to obtain a real function inequality $\gamma \geq 0$ defining a set Ω constructed from primitive sets Ω_j, it suffices to construct an appropriate R-function and substitute for its arguments the real functions ω_j defining the primitive sets Ω_j implicitly as $\omega_j \geq 0$ [176, Theorem 3]. In our case, the set Ω would be a polygon P_i we want to represent, the sets Ω_j would be half-spaces induced by the polygon edges, and the functions $\omega_j : \mathbb{R}^2 \to \mathbb{R}$ their corresponding hyperplane equations, given by

$$\omega_j(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_j)^\top \mathbf{n}_j \tag{A.6}$$

Here $\mathbf{x}_j$ is any arbitrary point on the edge hyperplane and $\mathbf{n}_j$ its normal vector, pointing towards the polygon's interior.

This result allows us to use a variant of the method presented in [176] and construct representations of polygons in the form of AND-OR trees [167], as shown in the example of Fig. A.1. Briefly, the interior of a polygon can be represented as the intersection of two or more *polygonal chains*, i.e. sequences of edges that meet at the polygon's convex hull. In the same way, each of these chains can then be split recursively into smaller subchains at the

vertices of its convex hull to form a tree structure. The root node of the tree is the original polygon, with each other node corresponding to a polygonal chain; the leaves of the tree are single hyperplanes, the edges of the polygon described by functions ω_j. If the split occurs at a concave vertex of the *original* polygon, then the subchains are combined using set union (i.e. disjunction); otherwise, they are combined using set intersection (i.e. conjunction), as shown in Fig. A.1. In this way, by having as input just the vertices of the polygon in counterclockwise order, we are able to construct an implicit representation for each node of the tree bottom-up, using the R-functions (A.4) and (A.5), until we reach the root node of the tree. If we want $\beta_i > 0$ in the exterior of P_i, we can negate the result (i.e., we use the R-function (A.3)) to obtain the function β_i, which is analytic everywhere except for the polygon vertices [176]. This is the reason our results in Section 7.3 still hold, with the map $\mathbf{h}^{\mathcal{I}}$ being a C^∞ diffeomorphism away from the polygon vertices.

A.1.3 R-functions as Approximations of the Distance Function

It is important to mention that, away from the corners and in a neighborhood of the polygon, *normalized* R-functions constructed using (A.3)-(A.5) behave as smooth p-th order approximations of the (non-differentiable) distance function to the polygon, as shown in Fig. A.1-(b),(c). The reader is referred to [176] for more details; in our setting, a sufficient condition for normalization is to make sure that for each ω_j given in (A.6), the corresponding normal vector $\mathbf{n}_j$ has unit norm [176]. This property is quite useful for our purposes, as it endows the implicit representation of our polygons with a physical meaning, compared to other representations (e.g., the homogeneous function representations in [164]). Numerical experimentation showed that even $p = 2$ gives sufficiently good results in our setting.

A.2 Construction of Polygonal Collars

Definitions 7.3, 7.5 and 7.7 provide the basic guidelines for constructing admissible polygonal collars that fit our formal results. However, there is not a unique way of performing this operation. Here, we describe the method employed for a single polygon P, contained in either $\mathcal{D}^{\mathcal{I}}_{map}$ or $\mathcal{B}^{\mathcal{I}}_{map}$, whose triangulation tree $\mathcal{T}_P := (\mathcal{V}_P, \mathcal{E}_P)$ has already been constructed,

according to Section 7.3.1, and the corresponding centers of transformation $\mathbf{x}_j^*$ have already been identified, according to Definitions 7.2, 7.4 and 7.6 for all triangles $j \in \mathcal{V}_P$. We assume that the value of the corresponding clearance ε_P, according to Assumption 7.3, is also known.

Algorithm A.1 Construction of the polygonal collars $\overline{\mathcal{Q}}_j$ for all triangles $j \in \mathcal{V}_P$ of a polygon P, whose triangulation tree $\mathcal{T}_P := (\mathcal{V}_P, \mathcal{E}_P)$ and associated clearance ε_P are known.

function COLLARCONSTRUCTION(P, ε_P)
  $\mathcal{V}_P \leftarrow \texttt{sort}(\mathcal{V}_P)$ ▷ Sort in descending depth
  do
    $j \leftarrow \texttt{pop}(\mathcal{V}_P)$ ▷ Pop next triangle
    $\mathcal{Q}_j \leftarrow \mathbf{x}_j^* \mathbf{x}_{2j} \mathbf{x}_{3j} \mathbf{x}_{1j} \mathbf{x}_j^*$
    $\mathcal{A}_j \leftarrow \texttt{dilate}(\mathcal{Q}_j, \varepsilon_P)$ ▷ Dilate $\mathcal{Q}_j$ by ε_P
    if j is `root` and $P \in \mathcal{D}_{map}^{\mathcal{I}}$ **then**
      $\overline{\mathcal{Q}}_j \leftarrow \mathcal{A}_j \cap \mathcal{F}_{map,j}^{\mathcal{I}}$
    else if j is `root` and $P \in \mathcal{B}_{map}^{\mathcal{I}}$ **then**
      $\mathcal{R}_j \leftarrow (\mathcal{A}_j \cap \mathcal{F}_{map,j}^{\mathcal{I}}) \cap (H_{1j} \cup H_{2j})$
      $\overline{\mathcal{Q}}_j \leftarrow \mathcal{R}_j \cup \mathbf{x}_j^* \mathbf{x}_{2j} \mathbf{x}_{1j} \mathbf{x}_j^*$
    else
      $\mathcal{R}_j \leftarrow (\mathcal{A}_j \cap \mathcal{F}_{map,j}^{\mathcal{I}}) \cap (H_{1j} \cup H_{2j})$
      $\mathcal{L}_j \leftarrow$ List of triangles that will succeed j
      do
        $i \leftarrow \texttt{pop}(\mathcal{L}_j)$ ▷ Pop next triangle
        if i is $p(j)$ **then**
          continue
        else
          $\mathcal{R}_j \leftarrow \mathcal{R}_j - i$ ▷ Polygon difference
        end if
      while $\mathcal{L}_j \neq \varnothing$
      $\{\mathcal{Z}\}_k \leftarrow \texttt{poly\_decomp}(\mathcal{R}_j)$ ▷ [98]
      $\overline{\mathcal{Q}}_j \leftarrow \mathcal{Z}_k$ such that $\mathcal{Q}_j \subset \mathcal{Z}_k$
    end if
  while $\mathcal{V}_P \neq \varnothing$
end function

Based on the above, the first step is to stack all triangles in $\mathcal{V}_P$ in order of descending depth, as prescribed by the sequence of purging transformations. Then, for each triangle $j \in \mathcal{V}_P$ in the stack, we first dilate the polygon $\mathcal{Q}_j$ by ε_P and take the intersection with $\mathcal{F}_{map,j}^{\mathcal{I}}$. If the triangle j is the root triangle of P and $P \in \mathcal{D}_{map}^{\mathcal{I}}$, this is enough to give an admissible polygonal collar. In any other case, we have to take the intersection of the

generated dilated polygon with the half spaces H_{1j} and H_{2j}, defined as

$$H_{1j} := \left\{ \mathbf{z} \in \mathbb{R}^2 \,\middle|\, (\mathbf{z} - \mathbf{x}_j^*)^\top \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{\mathbf{x}_j^* - \mathbf{x}_{1j}}{||\mathbf{x}_j^* - \mathbf{x}_{1j}||} \right) \geq 0 \right\} \tag{A.7}$$

$$H_{2j} := \left\{ \mathbf{z} \in \mathbb{R}^2 \,|\, (\mathbf{z} - \mathbf{x}_j^*)^\top \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{\mathbf{x}_{2j} - \mathbf{x}_j^*}{||\mathbf{x}_{2j} - \mathbf{x}_j^*||} \right) \geq 0 \right\} \tag{A.8}$$

i.e., the half spaces defined by hyperplanes passing through the center $\mathbf{x}_j^*$ and vertex $\mathbf{x}_{1j}$, and the center $\mathbf{x}_j^*$ and vertex $\mathbf{x}_{2j}$ respectively. The resulting polygon is guaranteed to be convex, but might intersect with triangles in $\mathcal{V}_P$ that will succeed j in the purging transformation. To solve that problem, we can take the difference of this polygon with all triangles that will succeed j in the purging transformation (except for its parent $p(j)$), decompose the final resulting polygon into its convex pieces using a variant of Keil's algorithm [98] (implemented in the C++ library CGAL [188] and in the Python package `poly_decomp` [153]), and use as $\overline{\mathcal{Q}}_j$ the convex piece that includes $\mathcal{Q}_j$, as prescribed by Definitions 7.3, 7.5 and 7.7. The whole procedure is shown in Algorithm A.1.

Appendix B

Derivations

B.1 Calculation of $D_{\mathbf{x}}\xi$

We can calculate

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{11}}{\partial x} = \sum_{j=1}^{M}\Bigg[2\sigma_j\frac{\partial\nu_j}{\partial x} + 2(\nu_j-1)\frac{\partial\sigma_j}{\partial x} + 2(x-x_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial x} + (x-x_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x^2} + (x-x_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x^2}\Bigg] \tag{B.1}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{11}}{\partial y} = \sum_{j=1}^{M}\Bigg[\sigma_j\frac{\partial\nu_j}{\partial y} + (\nu_j-1)\frac{\partial\sigma_j}{\partial y} + (x-x_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial x} + (x-x_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x\partial y} + (x-x_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial y} + (x-x_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x\partial y}\Bigg] \tag{B.2}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{12}}{\partial x} = \sum_{j=1}^{M}\Bigg[\sigma_j\frac{\partial\nu_j}{\partial y} + (x-x_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial y} + (x-x_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x\partial y} + (\nu_j-1)\frac{\partial\sigma_j}{\partial y} + (x-x_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial x} + (x-x_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x\partial y}\Bigg] \tag{B.3}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{12}}{\partial y} = \sum_{j=1}^{M}\Bigg[2(x-x_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial y} + (x-x_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial y^2} + (x-x_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial y^2}\Bigg] \tag{B.4}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{21}}{\partial x} = \sum_{j=1}^{M}\Bigg[2(y-y_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial x} + (y-y_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x^2} + (y-y_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x^2}\Bigg] \tag{B.5}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{21}}{\partial y} = \sum_{j=1}^{M}\Bigg[\sigma_j\frac{\partial\nu_j}{\partial x} + (y-y_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial x} + (y-y_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x\partial y} + (\nu_j-1)\frac{\partial\sigma_j}{\partial x} + (y-y_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial y} + (y-y_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x\partial y}\Bigg] \tag{B.6}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{22}}{\partial x} = \sum_{j=1}^{M}\Bigg[\sigma_j\frac{\partial\nu_j}{\partial x} + (\nu_j-1)\frac{\partial\sigma_j}{\partial x} + (y-y_j^*)\frac{\partial\sigma_j}{\partial x}\frac{\partial\nu_j}{\partial y} + (y-y_j^*)\sigma_j\frac{\partial^2\nu_j}{\partial x\partial y} + (y-y_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial x} + (y-y_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial x\partial y}\Bigg] \tag{B.7}$$

$$\frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{22}}{\partial y} = \sum_{j=1}^{M}\Bigg[2\sigma_j\frac{\partial\nu_j}{\partial y} + 2(\nu_j-1)\frac{\partial\sigma_j}{\partial y} + 2(y-y_j^*)\frac{\partial\sigma_j}{\partial y}\frac{\partial\nu_j}{\partial y} + \sigma_j(y-y_j^*)\frac{\partial^2\nu_j}{\partial y^2} + (y-y_j^*)(\nu_j-1)\frac{\partial^2\sigma_j}{\partial y^2}\Bigg] \tag{B.8}$$

In the expressions above, we use elements of the Hessians

$$\nabla^2\sigma_j(\mathbf{x}) = \eta''(\beta_j(\mathbf{x}))(\nabla\beta_j(\mathbf{x}))(\nabla\beta_j(\mathbf{x}))^\top + \eta'(\beta_j(\mathbf{x}))\nabla^2\beta_j(\mathbf{x}) \tag{B.9}$$

$$\nabla^2\nu_j(\mathbf{x}) = \frac{3\rho_j}{||\mathbf{x}-\mathbf{x}_j^*||^5}(\mathbf{x}-\mathbf{x}_j^*)(\mathbf{x}-\mathbf{x}_j^*)^\top - \frac{\rho_j}{||\mathbf{x}-\mathbf{x}_j^*||^3}\mathbf{I} \tag{B.10}$$

Eventually we can calculate $vD_{\mathbf{x}}\xi \begin{bmatrix} \cos\psi & \sin\psi \end{bmatrix}^\top$, used in (6.25), as follows

$$vD_{\mathbf{x}}\xi \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} = \frac{(\alpha_1\beta_1 + \alpha_2\beta_2)v}{||\mathbf{e}(\mathbf{x},\psi)||^2} \tag{B.11}$$

with

$$\alpha_1 = -\left([D_{\mathbf{x}}\mathbf{h}]_{21}\cos\psi + [D_{\mathbf{x}}\mathbf{h}]_{22}\sin\psi\right) \tag{B.12}$$

$$\alpha_2 = [D_{\mathbf{x}}\mathbf{h}]_{11}\cos\psi + [D_{\mathbf{x}}\mathbf{h}]_{12}\sin\psi \tag{B.13}$$

$$\beta_1 = \frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{11}}{\partial x}\cos^2\psi + \left(\frac{[D_{\mathbf{x}}\mathbf{h}]_{11}}{\partial y} + \frac{[D_{\mathbf{x}}\mathbf{h}]_{12}}{\partial x}\right)\sin\psi\cos\psi + \frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{12}}{\partial y}\sin^2\psi \tag{B.14}$$

$$\beta_2 = \frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{21}}{\partial x}\cos^2\psi + \left(\frac{[D_{\mathbf{x}}\mathbf{h}]_{21}}{\partial y} + \frac{[D_{\mathbf{x}}\mathbf{h}]_{22}}{\partial x}\right)\sin\psi\cos\psi + \frac{\partial[D_{\mathbf{x}}\mathbf{h}]_{22}}{\partial y}\sin^2\psi \tag{B.15}$$

B.2 Inductive Computation of the Diffeomorphism at Execution Time

From the description of the diffeomorphism $\mathbf{h}^{\mathcal{I}}$ in Section 7.3, we see that $\mathbf{h}^{\mathcal{I}}$ is constructed in multiple steps by composition. Therefore, we can compute the value of $\mathbf{h}^{\mathcal{I}}(\mathbf{x})$ at $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map}$ inductively, by setting $\mathbf{h}^{\mathcal{I}}_0(\mathbf{x}) = \mathbf{x}$ and computing $\mathbf{h}^{\mathcal{I}}_k(\mathbf{x}) = \mathbf{h}^{\mathcal{I}}_{k,k-1} \circ \mathbf{h}^{\mathcal{I}}_{k-1}(\mathbf{x})$, with k spanning all triangles in $\mathcal{V}_P$ for all known obstacles P in both $\mathcal{D}^{\mathcal{I}}_{map}$ and $\mathcal{B}^{\mathcal{I}}_{map}$, and $\mathbf{h}^{\mathcal{I}}_{k,k-1}$ given either in (7.18) or (7.28). We can then see that, due to Lemma 7.4, $\mathbf{h}^{\mathcal{I}}_{k,k-1}$ can be generally written in the following form

$$\begin{aligned} \mathbf{h}^{\mathcal{I}}_{k,k-1}(\mathbf{x}) =& \sigma_{k,k-1}(\mathbf{x})\left[\mathbf{x}^*_{k,k-1} + \nu_{k,k-1}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*_{k,k-1})\right] \\ &+ (1 - \sigma_{k,k-1}(\mathbf{x}))\,\mathbf{x} \end{aligned} \tag{B.16}$$

with the switch $\sigma_{k,k-1}$ (see (7.15), (7.23)), deforming factor $\nu_{k,k-1}$ (see (7.16), (7.24), (7.25)) and center of the transformation $\mathbf{x}^*_{k,k-1}$ (see Definitions 7.2, 7.4, 7.7) depending on the particular triangle being purged.

We can, therefore, set $D_{\mathbf{x}}\mathbf{h}_0^{\mathcal{I}} := \mathbf{I}$, compute

$$\begin{aligned} D_{\mathbf{x}}\mathbf{h}_{k,k-1}^{\mathcal{I}} = &\left(\nu_{k,k-1}(\mathbf{x}) - 1\right)(\mathbf{x} - \mathbf{x}_{k,k-1}^{*})\nabla\sigma_{k,k-1}(\mathbf{x})^{\top} \\ &+ \sigma_{k,k-1}(\mathbf{x})(\mathbf{x} - \mathbf{x}_{k,k-1}^{*})\nabla\nu_{k,k-1}(\mathbf{x})^{\top} \\ &+ \left[1 + \sigma_{k,k-1}(\mathbf{x})\left(\nu_{k,k-1}(\mathbf{x}) - 1\right)\right]\mathbf{I} \end{aligned} \tag{B.17}$$

and use the chain rule to write

$$D_{\mathbf{x}}\mathbf{h}_k^{\mathcal{I}} = \left(D_{\mathbf{x}}\mathbf{h}_{k,k-1}^{\mathcal{I}} \circ \mathbf{h}_{k-1}^{\mathcal{I}}(\mathbf{x})\right) \cdot D_{\mathbf{x}}\mathbf{h}_{k-1}^{\mathcal{I}} \tag{B.18}$$

Finally, since (7.49) requires partial derivatives of $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}$, we can follow a similar procedure and the chain rule to compute the partial derivatives $\frac{\partial[D_{\mathbf{x}}\mathbf{h}_k^{\mathcal{I}}]_{ml}}{\partial[\mathbf{x}]_n}$, as functions of $\frac{\partial[D_{\mathbf{x}}\mathbf{h}_{k-1}^{\mathcal{I}}]_{ml}}{\partial[\mathbf{x}]_n}$ and $\frac{\partial[D_{\mathbf{x}}\mathbf{h}_{k,k-1}^{\mathcal{I}}]_{ml}}{\partial[\mathbf{x}]_n}\Big|_{\mathbf{h}_{k-1}^{\mathcal{I}}(\mathbf{x})}$, after initially setting all partial derivatives to zero: $\frac{\partial[D_{\mathbf{x}}\mathbf{h}_0^{\mathcal{I}}]_{ml}}{\partial[\mathbf{x}]_n} = 0$, with the indices $m, l, n \in \{1, 2\}$. Namely:

$$\begin{aligned} \frac{\partial[D_{\mathbf{x}}\mathbf{h}_k^{\mathcal{I}}]_{ml}}{\partial[\mathbf{x}]_n} = &\sum_{r=1}^{2}\Bigg(\left[D_{\mathbf{x}}\mathbf{h}_{k,k-1}^{\mathcal{I}} \circ \mathbf{h}_{k-1}^{\mathcal{I}}(\mathbf{x})\right]_{mr} \cdot \\ &\cdot \frac{\partial[D_{\mathbf{x}}\mathbf{h}_{k-1}^{\mathcal{I}}]_{rl}}{\partial[\mathbf{x}]_n} \\ &+ [D_{\mathbf{x}}\mathbf{h}_{k-1}^{\mathcal{I}}]_{rl}\sum_{s=1}^{2}[D_{\mathbf{x}}\mathbf{h}_{k-1}^{\mathcal{I}}]_{sn} \cdot \\ &\cdot \frac{\partial\left[D_{\mathbf{x}}\mathbf{h}_{k,k-1}^{\mathcal{I}}\right]_{mr}}{\partial[\mathbf{x}]_s}\Big|_{\mathbf{h}_{k-1}^{\mathcal{I}}(\mathbf{x})}\Bigg) \end{aligned} \tag{B.19}$$

where, from (B.17), we can compute

$$\begin{aligned}
\frac{\partial \left[D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}_{k,k-1}\right]_{mr}}{\partial[\mathbf{x}]_s} &= (\nu_{k,k-1}-1)\frac{\partial \sigma_{k,k-1}}{\partial[\mathbf{x}]_r}\delta_{ms} \\
&+ ([\mathbf{x}]_m - [\mathbf{x}^*_{k,k-1}]_m)\frac{\partial \sigma_{k,k-1}}{\partial[\mathbf{x}]_r}\frac{\partial \nu_{k,k-1}}{\partial[\mathbf{x}]_s} \\
&+ (\nu_{k,k-1}-1)([\mathbf{x}]_m - [\mathbf{x}^*_{k,k-1}]_m)\frac{\partial^2 \sigma_{k,k-1}}{\partial[\mathbf{x}]_r\partial[\mathbf{x}]_s} \\
&+ ([\mathbf{x}]_m - [\mathbf{x}^*_{k,k-1}]_m)\frac{\partial \sigma_{k,k-1}}{\partial[\mathbf{x}]_s}\frac{\partial \nu_{k,k-1}}{\partial[\mathbf{x}]_r} \\
&+ \sigma_{k,k-1}\frac{\partial \nu_{k,k-1}}{\partial[\mathbf{x}]_r}\delta_{ms} \\
&+ \sigma_{k,k-1}([\mathbf{x}]_m - [\mathbf{x}^*_{k,k-1}]_m)\frac{\partial^2 \nu_{k,k-1}}{\partial[\mathbf{x}]_r\partial[\mathbf{x}]_s} \\
&+ \sigma_{k,k-1}\frac{\partial \nu_{k,k-1}}{\partial[\mathbf{x}]_s}\delta_{mr} + (\nu_{k,k-1}-1)\frac{\partial \sigma_{k,k-1}}{\partial[\mathbf{x}]_s}\delta_{mr}
\end{aligned} \tag{B.20}$$

by using elements of the Hessians $\nabla^2\sigma_{k,k-1}, \nabla^2\nu_{k,k-1}$.

Appendix C

Proofs

C.1 Proofs of Results in Chapter 3

Proof of Lemma 3.1. In the global frame, define $r_x := x^* - x = d\cos(\phi+\psi)$, $r_y := y^* - y = d\sin(\phi+\psi)$, with $\phi = \arctan2(\mathrm{x}^*_{BF})$. Since the goal does not move, we see that $\dot{r}_x = -\dot{x} = -v\cos\psi$ and $\dot{r}_y = -\dot{y} = -v\sin\psi$. On the other hand, we see from Fig. 3.10 that

$$\begin{aligned}
\phi + \psi &= \arctan\left(\frac{y^*_{BF} - y}{x^*_{BF} - x}\right) = \tan^{-1}\left(\frac{r_y}{r_x}\right) \\
\Rightarrow \dot{\phi} + \omega &= \frac{1}{d^2}(\dot{r}_y r_x - \dot{r}_x r_y) = \frac{v}{d}\sin\phi \\
\Rightarrow \dot{\phi} &= \frac{v}{d}\sin\phi - \omega
\end{aligned}$$

from the definitions above. Also, since $d = \sqrt{r_x^2 + r_y^2}$, we can easily derive $\dot{d} = \frac{\dot{r}_x r_x + \dot{r}_y r_y}{d} = -v\cos\phi$. Focusing now on the robot's body frame, we can see that $x^*_{BF} = d\cos\phi$ and $y^*_{BF} = d\sin\phi$, so that by differentiation $\dot{x}^*_{BF} = \dot{d}\cos\phi - d\dot{\phi}\sin\phi$ and $\dot{y}^*_{BF} = \dot{d}\sin\phi + d\dot{\phi}\cos\phi$. Simple substitution of $\dot{d}$ and $\dot{\phi}$ from above yields $\dot{x}^*_{BF} = -v + \omega\, y^*_{BF}$ and $\dot{y}^*_{BF} = -\omega\, x^*_{BF}$ and this concludes the proof. □

C.2 Proofs of Results in Chapter 4

Proof of Lemma 4.1. (i) We know that $\mathbf{x}$ lies in the interior of $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$ by construction. We also see that $||\mathbf{x}-\mathbf{x}_{\text{offset}}(\mathbf{x})|| = |\rho_{\mathbf{x}}(\theta_m)-r| = \rho_{\mathbf{x}}(\theta_m)-r = d(\mathbf{x},\partial\mathcal{F}) < \epsilon$. Therefore, $\mathbf{x}$ also lies in the interior of $\mathcal{D}_w(\mathbf{x})$ and, hence, in the interior of $\mathcal{LF}_w(\mathbf{x})$.

(ii) From the construction of $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$, we know that $d(\mathbf{x},\partial\mathcal{LF}_{\mathcal{L}}(\mathbf{x})) = \frac{1}{2}(\rho_{\mathbf{x}}(\theta_m) - r)$. Therefore, in fact, $H_{\mathbf{n}_w}(\mathbf{x})$ is one the half spaces whose intersection constructs $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$. Its generating hyperplane corresponds to the obstacle $O_k \in \mathcal{O}$ of minimum distance from $\mathbf{x}$ and intersects $\mathcal{D}_w(\mathbf{x})$ at two points, as can be shown by simple substitution in (4.9). On the other hand, Assumption 4.1 and the choice of ϵ in (4.10), show that this is the only hyperplane that intersects $\mathcal{D}_w(\mathbf{x})$ and belongs to the boundary of $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$ and this concludes the proof.

(iii) Since $\mathbf{t}_w(\mathbf{x}) = \mathbf{J}\,\mathbf{n}_w(\mathbf{x})$, it is not hard to verify that $||\mathbf{x}_p(\mathbf{x}) - \mathbf{x}_{\text{offset}}(\mathbf{x})|| = \epsilon$, which shows that $\mathbf{x}_p(\mathbf{x})$ lies on the boundary of $\mathcal{D}_w(\mathbf{x})$. Since $(\mathbf{x}_p(\mathbf{x}) - \mathbf{x}_h(\mathbf{x})) \cdot \mathbf{n}_w(\mathbf{x}) = \frac{1}{2}\left[\epsilon - (\rho_{\mathbf{x}} - r)\right] = \frac{1}{2}(\epsilon - d(\mathbf{x},\partial\mathcal{F})) > 0$, we get that $\mathbf{x}_p(\mathbf{x})$ belongs to the half space $H_{\mathbf{n}_w}(\mathbf{x})$. Since in (ii) we proved that $\mathcal{LF}_w(\mathbf{x}) = \mathcal{D}_w(\mathbf{x}) \cap H_{\mathbf{n}_w}(\mathbf{x})$, we conclude that $\mathbf{x}_p(\mathbf{x})$ lies on the boundary of $\mathcal{LF}_w(\mathbf{x})$.

□

Proof of Proposition 4.1. (i) Let $O_j \in \mathcal{O}$ denote the obstacle which the robot follows. Since $\rho_{\mathbf{x}}(\theta_m)$ corresponds to the minimum distance of $\mathbf{x}$ from O_j, we can write

$$\rho_{\mathbf{x}}(\theta_m) = ||\mathbf{x} - \Pi_{O_j}(\mathbf{x})||$$
$$\mathbf{n}_w(\mathbf{x}) = \frac{\mathbf{x} - \Pi_{O_j}(\mathbf{x})}{||\mathbf{x} - \Pi_{O_j}(\mathbf{x})||}$$

Since metric projections onto closed convex sets (such as O_j) are known to be piecewise continuously differentiable [115, 175], we conclude that both $\rho_{\mathbf{x}}(\theta_m)$ and $\mathbf{n}_w(\mathbf{x})$ are piecewise continuously differentiable functions of $\mathbf{x}$. Now from (4.14) and since $\mathbf{n}_w(\mathbf{x}) =$

$\mathbf{J}\,\mathbf{t}_w(\mathbf{x})$ we can write

$$\mathbf{u}(\mathbf{x}) = k\left[\left(\frac{\epsilon}{2} + r - \rho_{\mathbf{x}}(\theta_m)\right)\mathbf{n}_w(\mathbf{x}) + a\frac{\epsilon\sqrt{3}}{2}\mathbf{J}\mathbf{n}_w(\mathbf{x})\right] \tag{C.1}$$

Therefore, we conclude that the wall following law $\mathbf{u}(\mathbf{x})$ is piecewise continuously differentiable as a composition of piecewise continuously differentiable functions.

(ii) Since piecewise continuously differentiable functions are also locally Lipschitz [36], and since locally Lipschitz functions defined on a compact domain are also globally Lipschitz, we conclude that $\mathbf{u}(\mathbf{x})$ is Lipschitz continuous using (i). The existence, uniqueness and continuous differentiability of its flow follow directly from this property.

(iii) This follows directly from the form of the wall following law in (C.1), since the coefficient corresponding to $\mathbf{t}_w(\mathbf{x}) = \mathbf{J}\,\mathbf{n}_w(\mathbf{x})$ can never be zero.

(iv) From Lemma 4.1, we know that for any $\mathbf{x} \in \mathcal{F}$, the wall following local free space $\mathcal{LF}_w(\mathbf{x})$ is a closed convex subset of $\mathcal{F}$, which is collision-free (as a subset of $\mathcal{LF}_{\mathcal{L}}(\mathbf{x})$) and contains both $\mathbf{x}$ and $\mathbf{x}_p(\mathbf{x})$. Hence, $-k(\mathbf{x}-\mathbf{x}_p(\mathbf{x})) \in T_{\mathbf{x}}\mathcal{F}$ is either interior directed or at worst tangent to the boundary of $\mathcal{F}$ and this concludes the proof.

(v) Similarly, we see that for any $\mathbf{x} \in \mathcal{F}$ satisfying $d(\mathbf{p}, \partial\mathcal{F}) = \epsilon$, the choice of ϵ in (4.10) implies that there is a unique obstacle $j = \arg\min_i d(\mathbf{x}, O_i)$ such that $-k(\mathbf{x} - \mathbf{x}_p(\mathbf{x})) \in T_{\mathbf{x}}\mathcal{F}$ is interior directed to the set $\left\{\mathbf{p} \in \mathbb{R}^2 \,|\, d(\mathbf{p}, O_j) < \epsilon\right\}$ and, hence, interior directed to the set $\left\{\mathbf{p} \in \mathbb{R}^2 \,|\, d(\mathbf{p}, \partial\mathcal{F}) < \epsilon\right\}$. Similar reasoning leads to the conclusion that $\left\{\mathbf{p} \in \mathbb{R}^2 \,\middle|\, d(\mathbf{p}, \partial\mathcal{F}) > \frac{\epsilon}{2}\right\}$ is positively invariant under the wall following law, since for $d(\mathbf{x}, \partial\mathcal{F}) = \rho_{\mathbf{x}}(\theta_m) - r = \frac{\epsilon}{2}$, (C.1) gives $\mathbf{u}(\mathbf{x}) \parallel \mathbf{t}_w(\mathbf{x})$ and this concludes the proof.

□

Proof of Proposition 4.2. Suppose this is not true. Then there exists $t_c > 0$ such that $l = \arg\min_i d(\mathbf{x}^{t_c}, O_i) \neq k$ and $d(\mathbf{x}^{t_c}, \partial\mathcal{F}) < \epsilon$.[1] This implies that $d(\mathbf{x}^{t_c}, \partial\mathcal{F}) = d(\mathbf{x}^{t_c}, O_l) - r$,

[1] Here we slightly abuse the notation, since O_l could as well correspond to the boundary of the workspace $\partial\mathcal{W}$. The analysis still holds, because of the particular bounds provided for ϵ in (4.10).

which gives $d(\mathbf{x}^{t_c}, O_l) < r + \epsilon < \frac{1}{2} d(O_k, O_l)$ from the choice of ϵ. Hence, from the triangle inequality, we have

$$\begin{aligned} d(\mathbf{x}^{t_c}, O_k) &\geq d(O_k, O_l) - d(\mathbf{x}^{t_c}, O_l) \\ &> \frac{1}{2} d(O_k, O_l) \\ &\geq \frac{1}{2}\eta \\ &> (r + \max_j \rho_j) + \epsilon \\ &> r + \epsilon \end{aligned}$$

from (4.10). Therefore, we get that $d(\mathbf{x}^{t_c}, O_k) > r + \epsilon$. From the assumptions, we have $d(\mathbf{x}^0, \partial\mathcal{F}) = d(\mathbf{x}^0, O_k) - r < \epsilon$, which implies $d(\mathbf{x}^0, O_k) < r + \epsilon$. Since $d(\mathbf{x}^t, O_k)$ is continuous, as the composition of the distance function to a subset of $\mathbb{R}^2$ with the continuous flow $\mathbf{x}^t$, we can use the Intermediate Value Theorem to deduce that there exists a time $t_m \in (0, t_c)$ such that $d(\mathbf{x}^{t_m}, O_k) = r + \epsilon$. From the choice of ϵ in (4.10), this implies that

$$d(\mathbf{x}^{t_m}, O_k) < \frac{1}{2}\eta \tag{C.2}$$

so that $d(\mathbf{x}^{t_m}, \partial\mathcal{F}) = d(\mathbf{x}^{t_m}, O_k) - r = \epsilon$, violating the assumption that $d(\mathbf{x}^t, \partial\mathcal{F}) < \epsilon$ for all $t > 0$ and leading to a contradiction. □

Proof of Theorem 4.1. The fact that, under the wall following law in (4.13), the robot follows the boundary of a unique obstacle follows readily from Proposition 4.2, the continuity of the flow and the positive invariance of $\left\{\mathbf{p} \in \mathbb{R}^2 \,|\, d(\mathbf{p}, \partial\mathcal{F}) < \epsilon\right\}$ as derived in Proposition 4.1. Finally, from (C.1), notice that

$$\mathbf{u}(\mathbf{x}) \cdot \mathbf{t}_w(\mathbf{x}) = a\frac{\epsilon\sqrt{3}}{2} \tag{C.3}$$

which implies that $|\sigma(\mathbf{x})| \in (0, 1]$ and $\text{sign}(\sigma(\mathbf{x})) = \text{sign}(a)$. Since $\mathbf{u}(\mathbf{x}) \cdot \mathbf{t}_w(\mathbf{x})$ expresses the component of $\mathbf{u}(\mathbf{x})$ along the tangent to the obstacle boundary $\mathbf{t}_w(\mathbf{x})$, is always nonzero and

does not change sign, we conclude that the robot will follow the boundary of the obstacle clockwise or counterclockwise, depending on a. The rest of the claims derive immediately from Proposition 4.1. □

Proof Sketch of Theorem 4.3. Positive invariance of $\mathcal{F}$ is guaranteed from [7] with the particular choice of $\mathcal{LF}_v(\mathbf{x}), \mathcal{LF}_\omega(\mathbf{x})$. The existence, uniqueness and continuous differentiability of the flow are guaranteed through the piecewise continuous differentiability of the vector field, similarly to the proof of Proposition 4.1-(ii). We can also prove the positive invariance of the set $\{\mathbf{p} \in \mathcal{W} \,|\, d(\mathbf{p}, \partial\mathcal{F}) < \epsilon\}$ by the particular selection of $\mathbf{x}_p(\mathbf{x})$, as in Proposition 4.1-(v). The only problem, unique to differential drive robots, is that the robot orientation might not originally be aligned with $\mathbf{x}_p(\mathbf{x})$. However, since the robot is not allowed to move backwards (from (4.16)), the angular control law in (4.17) with $\mathbf{x}^* = \mathbf{x}_p(\mathbf{x})$ will force the robot to turn towards $\mathbf{x}_p(\mathbf{x})$ in finite time and continue following that direction onwards. □

C.3 Proofs of Results in Chapter 5

Proof of Lemma 5.1. It is shown in [44] that the extended local sets $\mathcal{R}_\mathcal{S}(\mathbf{x}_0, r, \alpha)$ are open. Now consider a point $\mathbf{u} \in U_{\rho(\cdot)}(\mathcal{S})$. Then, by definition, there exists a $\mathbf{y} \in \mathcal{S}$, a direction $\zeta \in N^P(\mathcal{S}; \mathbf{y}) \cap \mathsf{B}(\mathbf{0}, 1)$ and a real $t \in [0, \rho(\mathbf{y})]$ such that $\mathbf{u} = \mathbf{y} + t\zeta$. This shows that $\mathbf{u}$ also belongs in the set $\mathcal{R}_\mathcal{S}(\mathbf{y}, \rho(\mathbf{y}), \alpha)$ for some $\alpha > 0$, and concludes the proof. □

Proof of Theorem 5.1. The proof is almost identical to the proof of Theorem 4.1. The only difference here is that the wall following law is not piecewise continuously differentiable but just locally Lipschitz. This, however, does not change its basic properties. The key in the proof is the requirement that $\min \rho_{O_i} > r + \epsilon$ for each O_i. Since it is shown in Proposition 4.1 that $\left\{\mathbf{p} \in \mathcal{W} \,\middle|\, \frac{\epsilon}{2} < d(\mathbf{p}, \partial\mathcal{F}) < \epsilon\right\}$ is positively invariant under the flow of the wall following law, we are guaranteed that the robot will never exit $U_{\rho(\cdot)}(\mathcal{S})$, which ensures local Lipschitz continuity of the vector field. □

C.4 Proofs of Results in Chapter 6

C.4.1 Proofs of Results in Section 6.2

Proof of Lemma 6.2. Since both the switches σ_j and the deforming factors ν_j are smooth, for $j = 1, \ldots, M$, the only technical challenge here is introduced by the fact that the number M of discovered star-shaped obstacles in $\mathcal{F}_{map}$ is not constant and changes as the robot navigates the workspace.

Notice from (6.6) that all the derivatives of η_j used in the construction of the switch σ_j for any j are zero if and only if η_j is zero. Therefore, in order to guarantee smoothness of $\mathbf{h}$, we just have to ensure that when a new obstacle k is added to the semantic map, the value of σ_k will be zero. This follows directly from the assumption that the sensor range R is much greater than ε_k, which implies that when obstacle k is discovered, the robot position $\mathbf{x}$ will lie outside the set $\{\mathbf{q} \in \mathcal{F}_{map} \,|\, 0 \leq \beta_k(\mathbf{q}) < \varepsilon_k\}$ and therefore the value of σ_k will be zero. □

Proof of Proposition 6.1. First of all, the map $\mathbf{h}$ is smooth as shown in Lemma 6.2. Therefore, in order to prove that $\mathbf{h}$ is a C^∞ diffeomorphism, we will follow the procedure outlined in [131], also followed in [164], to show that

1. $\mathbf{h}$ has a non-singular differential on $\mathcal{F}_{map}$
2. $\mathbf{h}$ preserves boundaries, i.e., $\mathbf{h}(\partial_j \mathcal{F}_{map}) \subset \partial_j \mathcal{F}_{model}, j \in \{0, \ldots, M+N\}$.[2]
3. the boundary components of $\mathcal{F}_{map}$ and $\mathcal{F}_{model}$ are pairwise homeomorphic, i.e., $\partial_j \mathcal{F}_{map} \cong \partial_j \mathcal{F}_{model}, j \in \{0, \ldots, M+N\}$.

We begin with property 1. Using Lemma 6.1 and observing from (6.7) and (6.8) that a switch $\sigma_k, k \in \{1, \ldots, M\}$ is zero if and only if its gradient $\nabla\sigma_k$ is zero, we observe from (6.13) that $D_{\mathbf{x}}\mathbf{h}$ is either the identity map (which is non-singular) or depends only a single switch $\sigma_k, k \in \{1, \ldots, M\}$ when $0 \leq \beta_k(\mathbf{x}) < \varepsilon_k$. In that case, we can isolate the k-th

[2]Here we denote by $\partial_j \mathcal{F}$ the j-th connected component of the boundary of $\mathcal{F}$ (that corresponding to $\tilde{O}_j$), with $\partial_0 \mathcal{F}$ the outer boundary of $\mathcal{F}$.

term in (6.13) and write the map differential as

$$\begin{aligned} D_{\mathbf{x}}\mathbf{h} = D_{\mathbf{x}}\mathbf{h}_k =& \left[1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1)\right]\mathbf{I} + (\mathbf{x}-\mathbf{x}_k^*)\Big[\sigma_k(\mathbf{x})\nabla\nu_k(\mathbf{x})^\top \\ &+(\nu_k(\mathbf{x})-1)\nabla\sigma_k(\mathbf{x})^\top\Big] \\ =& \left[1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1)\right]\mathbf{I} + (\mathbf{x}-\mathbf{x}_k^*)\left[-\frac{\rho_k\sigma_k(\mathbf{x})}{||\mathbf{x}-\mathbf{x}_k^*||^3}(\mathbf{x}-\mathbf{x}_k^*)^\top\right. \\ &\left.+\eta_k'(\beta_k(\mathbf{x}))(\nu_k(\mathbf{x})-1)\nabla\beta_k(\mathbf{x})^\top\right] \end{aligned} \tag{C.4}$$

From this expression, we can find with some computation

$$\begin{aligned} \mathrm{tr}(D_{\mathbf{x}}\mathbf{h}_k) =& [1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1)] + (1-\sigma_k(\mathbf{x})) \\ &+ \eta_k'(\beta_k(\mathbf{x}))(\nu_k(\mathbf{x})-1)(\mathbf{x}-\mathbf{x}_k^*)^\top\nabla\beta_k(\mathbf{x}) \end{aligned} \tag{C.5}$$

However, we know that

$$\frac{\sigma_k(\mathbf{x})-1}{\sigma_k(\mathbf{x})} \leq 0 < \nu_k(\mathbf{x}) \tag{C.6}$$

since $0 < \sigma_k(\mathbf{x}) \leq 1$, giving $1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1) > 0$. Also, $\eta_k'(\beta_k(\mathbf{x})) < 0$ by construction (since $\beta_k(\mathbf{x}) < \varepsilon_k$), $\nu_k(\mathbf{x}) - 1 < 0$ and $(\mathbf{x}-\mathbf{x}_k^*)^\top\nabla\beta_k(\mathbf{x}) > 0$ in the set $\{\mathbf{x} \in \mathcal{F}_{map} \,|\, 0 \leq \beta_k(\mathbf{x}) < \varepsilon_k\}$, because of Assumption 6.1-(c). Therefore, we get $\mathrm{tr}(D_{\mathbf{x}}\mathbf{h}_k) > 0$ for all $\mathbf{x}$ such that $0 \leq \beta_k(\mathbf{x}) < \varepsilon_k$. Also, since $\mathcal{F}_{map} \subset \mathbb{R}^2$, we can similarly compute

$$\begin{aligned} \det(D_{\mathbf{x}}\mathbf{h}_k) =& g_k'(\beta_k(\mathbf{x}))(\nu_k(\mathbf{x})-1)[1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1)](\mathbf{x}-\mathbf{x}_k^*)^\top\nabla\beta_k(\mathbf{x}) \\ &+ (1-\sigma_k(\mathbf{x}))[1+\sigma_k(\mathbf{x})(\nu_k(\mathbf{x})-1)] \end{aligned} \tag{C.7}$$

which leads to $\det(D_{\mathbf{x}}\mathbf{h}_k) > 0$ for all $\mathbf{x}$ such that $\beta_k(\mathbf{x}) < \varepsilon_k$. Since $\det(D_{\mathbf{x}}\mathbf{h}_k) > 0$ and $\mathrm{tr}(D_{\mathbf{x}}\mathbf{h}_k) > 0$, we conclude that $D_{\mathbf{x}}\mathbf{h}_k$ has two strictly positive eigenvalues in the set $\{\mathbf{x} \in \mathcal{F}_{map} \,|\, 0 \leq \beta_k(\mathbf{x}) < \varepsilon_k\}$. Since this is true for any $k \in \{1, \ldots, M\}$, it follows that $D_{\mathbf{x}}\mathbf{h}$ has two strictly positive eigenvalues in $\mathcal{F}_{map}$ and, thus, is non-singular in $\mathcal{F}_{map}$.

Next, pick a point $\mathbf{x} \in \partial_j\mathcal{F}_{map}$ for any $j \in \{0, \ldots, M+N\}$. This point could lie on the outer boundary of $\mathcal{F}_{map}$, on the boundary of one of the N unknown but visible convex

obstacles, or on the boundary of one of the M star-shaped obstacles. In the first two cases, we have $\mathbf{h}(\mathbf{x}) = \mathbf{x}$, while in the latter case

$$\mathbf{h}(\mathbf{x}) = \mathbf{x}_k^* + \frac{\rho_k}{||\mathbf{x} - \mathbf{x}_k^*||}(\mathbf{x} - \mathbf{x}_k^*) \tag{C.8}$$

for some $k \in \{1, \ldots, M\}$, sending $\mathbf{x}$ to the boundary of the k-th disk in $\mathcal{F}_{model}$. This shows that we always have $\mathbf{h}(\mathbf{x}) \in \partial_j \mathcal{F}_{model}$ and, therefore, the map satisfies property 2.

Finally, property 3 derives from above and the fact that each boundary segment $\partial_j \mathcal{F}_{map}$ is an one-dimensional manifold, the boundary of either a convex set or a star-shaped set, both of which are homeomorphic to the corresponding boundary $\partial_j \mathcal{F}_{model}$. □

C.4.2 Proofs of Results in Section 6.3

Proof of Proposition 6.2. Since $\mathbf{h}$ is just the identity transformation away from any star-shaped obstacle and the control law $\mathbf{u}$ guarantees collision avoidance in that case, as shown in [7], it suffices to show that the robot can never penetrate any star-shaped obstacle, i.e., for any $\mathbf{x}_c$ such that $\beta_k(\mathbf{x}_c) = 0$ for some $k \in \{1, \ldots, M\}$, we have $\mathbf{u}(\mathbf{x}_c)^\top \nabla \beta_k(\mathbf{x}_c) \geq 0$. For such a point $\mathbf{x}_c$, we get from (6.10) and (6.13)

$$\begin{aligned}
D_{\mathbf{x}}\mathbf{h}(\mathbf{x}_c) = D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c) =& \left[1 + \sigma_k(\mathbf{x}_c)(\nu_k(\mathbf{x}_c) - 1)\right]\mathbf{I} + (\mathbf{x}_c - \mathbf{x}_k^*)\Big[\sigma_k(\mathbf{x}_c)\nabla\nu_k(\mathbf{x}_c)^\top \\
&+ (\nu_k(\mathbf{x}_c) - 1)\nabla\sigma_k(\mathbf{x}_c)^\top\Big] \\
=& \left[1 + \sigma_k(\mathbf{x}_c)(\nu_k(\mathbf{x}_c) - 1)\right]\mathbf{I} \\
&+ (\mathbf{x}_c - \mathbf{x}_k^*)\left[-\frac{\rho_k\sigma_k(\mathbf{x}_c)}{||\mathbf{x}_c - \mathbf{x}_k^*||^3}(\mathbf{x}_c - \mathbf{x}_k^*)^\top\right. \\
&\left.+\eta_k'(\beta_k(\mathbf{x}_c))(\nu_k(\mathbf{x}_c) - 1)\nabla\beta_k(\mathbf{x}_c)^\top\right] \\
=& \frac{\rho_k}{||\mathbf{x}_c - \mathbf{x}_k^*||}\mathbf{I} + (\mathbf{x}_c - \mathbf{x}_k^*)\left[-\frac{\rho_k}{||\mathbf{x}_c - \mathbf{x}_k^*||^3}(\mathbf{x}_c - \mathbf{x}_k^*)^\top\right. \\
&\left.+\eta_k'(\beta_k(\mathbf{x}_c))(\nu_k(\mathbf{x}_c) - 1)\nabla\beta_k(\mathbf{x}_c)^\top\right]
\end{aligned} \tag{C.9}$$

since $\sigma_k(\mathbf{x}_c) = 1$. Since $\mathcal{F}_{map} \subset \mathbb{R}^2$, we can explicitly compute the inverse of the 2x2 matrix $D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)$ from its four elements $[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{11}$, $[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{12}$, $[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{21}$, $[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{22}$ as

$$D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)^{-1} = \frac{1}{\det(D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c))} \begin{bmatrix} [D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{22} & -[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{12} \\ -[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{21} & [D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]_{11} \end{bmatrix} \tag{C.10}$$

and after some simple computations, we can eventually find

$$[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]^{-\top}\nabla\beta_k(\mathbf{x}_c) = \frac{\rho_k\,(\mathbf{x}_c - \mathbf{x}_k^*)^\top \nabla\beta_k(\mathbf{x}_c)}{||\mathbf{x}_c - \mathbf{x}_k^*||^3 \det(D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)))}(\mathbf{x}_c - \mathbf{x}_k^*) \tag{C.11}$$

On the other hand,

$$\mathbf{u}(\mathbf{x}_c) = -k\,[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]^{-1}\,\left(\mathbf{h}(\mathbf{x}_c) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}_c))}(\mathbf{x}_d)\right) \tag{C.12}$$

Since $\mathbf{x}_c$ belongs to the boundary of the obstacle k, then by construction of the diffeomorphism, $\mathbf{h}(\mathbf{x}_c)$ will belong to the boundary of the disk with radius ρ_k centered at $\mathbf{x}_k^*$ and the associated hyperplane [7] will be tangent to that disk at $\mathbf{h}(\mathbf{x}_c)$. Therefore, the projected goal $\Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}_c))}(\mathbf{x}_d)$ will belong to the halfspace defined by the outward normal vector from $\mathbf{x}_k^*$ to $\mathbf{h}(\mathbf{x}_c)$ at $\mathbf{h}(\mathbf{x}_c)$ and we have

$$\mathbf{u}(\mathbf{x}_c) = [D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]^{-1}\,\mathbf{t}(\mathbf{x}_c) \tag{C.13}$$

with $\mathbf{t}(\mathbf{x}_c)^\top(\mathbf{h}(\mathbf{x}_c) - \mathbf{x}_k^*) \geq 0$. Since by construction of the diffeomorphism $\mathbf{h}(\mathbf{x}_c) = \mathbf{x}_k^* + \rho_k \frac{\mathbf{x}_c - \mathbf{x}_k^*}{||\mathbf{x}_c - \mathbf{x}_k^*||}$, we derive that

$$\mathbf{t}(\mathbf{x}_c)^\top(\mathbf{x}_c - \mathbf{x}_k^*) \geq 0 \tag{C.14}$$

Using the above results, we see that

$$\begin{aligned} \mathbf{u}(\mathbf{x}_c)^\top\nabla\beta_k(\mathbf{x}_c) &= \left[[D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)]^{-\top}\,\nabla\beta_k(\mathbf{x}_c)\right]^\top \mathbf{t}(\mathbf{x}_c) \\ &= \frac{\rho_k\,(\mathbf{x}_c - \mathbf{x}_k^*)^\top\nabla\beta_k(\mathbf{x}_c)}{||\mathbf{x}_c - \mathbf{x}_k^*||^3 \det\left(D_{\mathbf{x}}\mathbf{h}_k(\mathbf{x}_c)\right)}(\mathbf{x}_c - \mathbf{x}_k^*)^\top\mathbf{t}(\mathbf{x}_c) \geq 0 \end{aligned} \tag{C.15}$$

using (C.14) and the fact that $(\mathbf{x_c} - \mathbf{x}_k^*)^\top \nabla \beta_k(\mathbf{x}_c) > 0$, since $\mathbf{x}_c$ belongs to the boundary of a star-shaped obstacle [164]. □

Proof of Lemma 6.3. The proof of this lemma derives immediately from [7, Propositions 5,11], from which we can infer that the set of stationary points of the vector field $D_{\mathbf{x}}\mathbf{h} \cdot \mathbf{u}(\mathbf{x})$, defined on $\mathcal{F}_{model}$, is $\{\mathbf{x}_d\} \bigcup \{\mathbf{s}_j\}_{j\in\{1,\dots,M\}} \bigcup_{i=1}^{N} \mathcal{G}_i$, with $\mathbf{x}_d$ being a locally stable equilibrium of $D_{\mathbf{x}}\mathbf{h} \cdot \mathbf{u}(\mathbf{x})$ and each other point being a nondegenerate saddle, since [7, Assumption 2] is satisfied for the obstacles in $\mathcal{F}_{model}$ by construction. To complete the proof, we just have to note that the index of an isolated zero of a vector field does not change under diffeomorphisms of the domain [78]. □

Proof of Proposition 6.3. Consider the smooth Lyapunov function candidate $V(\mathbf{x}) = ||\mathbf{h}(\mathbf{x}) - \mathbf{x}_d||^2$, justified by the fact that $\mathbf{h}(\mathbf{x}_d) = \mathbf{x}_d$ by construction of the diffeomorphism, since we have assumed that $\beta_j(\mathbf{x}_d) > \varepsilon_j$ for all $j \in \{1, \dots, M\}$. Using (6.16)

$$\begin{aligned}
\frac{dV}{dt} =& 2(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top (\mathbf{D_x h})\dot{\mathbf{x}} = -2k(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right) \\
=& -2k \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d) + \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d) - \mathbf{x}_d\right)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right) \\
=& -2k||\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)||^2 \\
& + 2k \left(\mathbf{x}_d - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right) \\
\leq& -2k||\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)||^2 \leq 0
\end{aligned} \tag{C.16}$$

since $\mathbf{h}(\mathbf{x}) \in \mathcal{LF}(\mathbf{h}(\mathbf{x}))$, which implies that

$$\left(\mathbf{x}_d - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)\right) \leq 0 \tag{C.17}$$

since either $\mathbf{x}_d = \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)$, or $\mathbf{x}_d$ and $\mathbf{h}(\mathbf{x})$ are separated by a hyperplane passing through $\Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)$. Therefore, similarly to [7], using LaSalle's invariance principle we see that every trajectory starting in $\mathcal{F}_{map}$ approaches the largest invariant set in $\{\mathbf{x} \in \mathcal{F}_{map} \,|\, \dot{V}(\mathbf{x}) = 0\}$, i.e. the equilibrium points of (6.16). The desired result follows from Lemma 6.3, since $\mathbf{x}_d$ is the only locally stable equilibrium of our control law and the rest

of the stationary points are nondegenerate saddles, whose regions of attraction have empty interior in $\mathcal{F}_{map}$. □

Proof of Proposition 6.4. Note that the jacobian of $\overline{\mathbf{h}}$ will be given by

$$D_{\overline{\mathbf{x}}}\overline{\mathbf{h}} = \left[\begin{array}{c|c} D_{\mathbf{x}}\mathbf{h} & \mathbf{0}_{2\times 1} \\ \hline D_{\mathbf{x}}\xi & \dfrac{\partial \xi}{\partial \psi} \end{array}\right] \tag{C.18}$$

Since we have already shown in Proposition 6.1 that $D_{\mathbf{x}}\mathbf{h}$ is non-singular, it suffices to show that $\frac{\partial \xi}{\partial \psi} \neq 0$ for all $\overline{\mathbf{x}} \in \mathcal{F}_{map} \times S^1$. From (6.20) we can derive

$$\frac{\partial \xi}{\partial \psi} = \frac{\det(D_{\mathbf{x}}\mathbf{h})}{||\mathbf{e}(\overline{\mathbf{x}})||^2} \tag{C.19}$$

Therefore, we immediately get that $\frac{\partial \xi}{\partial \psi} \neq 0$ for all $\overline{\mathbf{x}} \in \mathcal{F}_{map} \times S^1$ since $\det(D_{\mathbf{x}}\mathbf{h}) \neq 0$ and $||\mathbf{e}(\overline{\mathbf{x}})|| \neq 0$ for all $\mathbf{x} \in \mathcal{F}_{map}$, because $D_{\mathbf{x}}\mathbf{h}$ is non-singular on $\mathcal{F}_{map}$. This implies that $D_{\overline{\mathbf{x}}}\overline{\mathbf{h}}$ is non-singular on $\mathcal{F}_{map} \times S^1$.

Next, we note that $\partial\left(\mathcal{F}_{map} \times S^1\right) = \partial\mathcal{F}_{map} \times S^1$, since S^1 is a manifold without boundary. Similarly, $\partial\left(\mathcal{F}_{model} \times S^1\right) = \partial\mathcal{F}_{model} \times S^1$. Hence, we can easily complete the proof following a similar procedure with the end of the proof of Proposition 6.1. □

Proof of Theorem 6.2. We have already established that $||\mathbf{e}(\overline{\mathbf{x}})||$ and $\frac{\partial \xi}{\partial \psi}$ are nonzero for all $\overline{\mathbf{x}} \in \mathcal{F}_{map} \times S^1$ in the proof of Proposition 6.4, which implies that v and ω can have no singular points. Also notice that $||\mathbf{e}(\overline{\mathbf{x}})||$, $\frac{\partial \xi}{\partial \psi}$ and $D_{\mathbf{x}}\xi \begin{bmatrix} \cos\psi & \sin\psi \end{bmatrix}^\top$ are all smooth. Hence, the uniqueness and existence of the flow generated by control law (6.26) can be established similarly to [7] through the flow properties of the controller in [12] (that we use here in (6.27)) and the facts that metric projections onto moving convex cells are piecewise continuously differentiable [115, 175] and the composition of piecewise continuously differentiable functions is piecewise continuously differentiable and, therefore, locally Lipschitz [36].

Positive invariance of $\mathcal{F}_{map} \times S^1$ can be proven following similar patterns with the proof of Proposition 6.2. Namely, it suffices to show that the robot can never penetrate an obstacle,

i.e., for any placement $(\mathbf{x}_c, \psi_c)$ such that $\beta_k(\mathbf{x}_c) = 0$ for some index $k \in \{1, \ldots, M\}$, we definitely have

$$\nabla\beta_k(\mathbf{x}_c)^\top \begin{bmatrix} v_c \cos\psi_c \\ v_c \sin\psi_c \end{bmatrix} \geq 0 \tag{C.20}$$

for any $\psi_c \in S^1$. We know from (6.22) that

$$\begin{bmatrix} v_c \cos\psi_c \\ v_c \sin\psi_c \end{bmatrix} = [D_{\mathbf{x}}\mathbf{h}(\mathbf{x}_c)]^{-1} \begin{bmatrix} \hat{v}_c \cos\varphi_c \\ \hat{v}_c \sin\varphi_c \end{bmatrix} \tag{C.21}$$

Therefore

$$\begin{aligned} \nabla\beta_k(\mathbf{x}_c)^\top \begin{bmatrix} v_c \cos\psi_c \\ v_c \sin\psi_c \end{bmatrix} =& \nabla\beta_k(\mathbf{x}_c)^\top \left([D_{\mathbf{x}}\mathbf{h}(\mathbf{x}_c)]^{-1} \begin{bmatrix} \hat{v}_c \cos\varphi_c \\ \hat{v}_c \sin\varphi_c \end{bmatrix} \right) \\ =& \left([D_{\mathbf{x}}\mathbf{h}(\mathbf{x}_c)]^{-\top} \nabla\beta_k(\mathbf{x}_c) \right)^\top \begin{bmatrix} \hat{v}_c \cos\varphi_c \\ \hat{v}_c \sin\varphi_c \end{bmatrix} \\ =& \frac{\rho_k \left(\mathbf{x}_c - \mathbf{x}_k^*\right)^\top \nabla\beta_k(\mathbf{x}_c)}{||\mathbf{x}_c - \mathbf{x}_k^*||^3 \det(D_{\mathbf{x}}\mathbf{h}(\mathbf{x}_c))} (\mathbf{x}_c - \mathbf{x}_k^*)^\top \begin{bmatrix} \hat{v}_c \cos\varphi_c \\ \hat{v}_c \sin\varphi_c \end{bmatrix} \end{aligned} \tag{C.22}$$

using (C.11). Hence, using the results from Proposition 6.2, we see that positive invariance of $\mathcal{F}_{map} \times S^1$ under law (6.26) is equivalent to positive invariance of $\mathcal{F}_{model} \times S^1$ under law (6.27), which is guaranteed from [7, Proposition 12].

Finally, consider the smooth Lyapunov function candidate $V(\mathbf{x}) = ||\mathbf{h}(\mathbf{x}) - \mathbf{x}_d||^2$. Then

$$\begin{aligned}
\frac{dV}{dt} =& 2(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top (\mathbf{D_x h})\dot{\mathbf{x}} \\
=& 2v\,(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top (\mathbf{D_x h}) \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \\
=& 2\overline{v}(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top \begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) \\ \sin\xi(\overline{\mathbf{x}}) \end{bmatrix} \\
=& -2k(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top \begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) \\ \sin\xi(\overline{\mathbf{x}}) \end{bmatrix} \begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) \\ \sin\xi(\overline{\mathbf{x}}) \end{bmatrix}^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right) \\
=& -2k(\mathbf{h}(\mathbf{x}) - \mathbf{x}_d)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right)
\end{aligned}$$

since $\begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) \\ \sin\xi(\overline{\mathbf{x}}) \end{bmatrix} \begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) \\ \sin\xi(\overline{\mathbf{x}}) \end{bmatrix}^\top$ is just the projection operator on the line defined by the vector $\begin{bmatrix} \cos\xi(\overline{\mathbf{x}}) & \sin\xi(\overline{\mathbf{x}}) \end{bmatrix}^\top$, with which $\left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right)$ is already parallel. Following this result, we get

$$\frac{dV}{dt} \leq -2k\left|\left|\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right|\right|^2 \leq 0 \tag{C.23}$$

since, similarly to the proof of Proposition 6.3, we have

$$\left(\mathbf{x}_d - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right)^\top \left(\mathbf{h}(\mathbf{x}) - \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\right) \leq 0 \tag{C.24}$$

Therefore, using LaSalle's invariance principle, we see that every trajectory starting in $\mathcal{F}_{map} \times S^1$ approaches the largest invariant set in $\{(\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1 \,|\, \dot{V}(\mathbf{x}) = 0\} = \{(\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1 \,|\, \mathbf{h}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\}$. At the same time, we know from (6.27) that $\mathbf{h}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)$ implies $v = 0$. From (6.26), for $v = 0$, we get that ω will be

zero at points where $\hat{\omega}$ is zero, i.e. at points $(\mathbf{x}, \psi) \in \mathcal{F}_{map} \times S^1$ where

$$\begin{bmatrix} -\sin\xi(\overline{\mathbf{x}}) \\ \cos\xi(\overline{\mathbf{x}}) \end{bmatrix}^\top \left(\mathbf{h}(\mathbf{x}) - \frac{\Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_G}(\mathbf{x}_d) + \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)}{2} \right) = 0 \tag{C.25}$$

Therefore the largest invariant set in $\{(\mathbf{x}, \psi) \,|\, \mathbf{h}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d)\}$ is the set of points $\overline{\mathbf{x}} = (\mathbf{x}, \psi)$ where the following two conditions are satisfied

$$\mathbf{h}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_\parallel}(\mathbf{x}_d) \tag{C.26}$$

$$\begin{bmatrix} -\sin\xi(\overline{\mathbf{x}}) \\ \cos\xi(\overline{\mathbf{x}}) \end{bmatrix}^\top \left(\mathbf{h}(\mathbf{x}) - \frac{\Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))\cap H_G}(\mathbf{x}_d) + \Pi_{\mathcal{LF}(\mathbf{h}(\mathbf{x}))}(\mathbf{x}_d)}{2} \right) = 0 \tag{C.27}$$

Using a similar argument to [7, Proposition 12], we can, therefore, verify that the set of stationary points of law (6.26) is given by

$$\{\mathbf{x}_d\} \times (-\pi, \pi] \cup \left\{ (\mathbf{q}, \psi) \,\middle|\, \mathbf{q} \in \{\mathbf{h}^{-1}(\mathbf{s}_j)\}_{j\in\{1,\dots,M\}} \bigcup_{i=1}^{N} \mathcal{G}_i, \begin{bmatrix} -\sin\xi(\mathbf{q}, \psi) \\ \cos\xi(\mathbf{q}, \psi) \end{bmatrix}^\top (\mathbf{q} - \mathbf{x}_d) = 0 \right\} \tag{C.28}$$

using (6.17). We can then invoke a similar argument to Proposition 6.3 to show that $\mathbf{x}_d$ locally attracts with any orientation ψ, while any configuration associated with any other equilibrium point is a nondegenerate saddle whose stable manifold is a set of measure zero, and the result follows. □

C.5 Proofs of Results in Chapter 7

C.5.1 Proofs of Results in Section 7.3

Proof of Lemma 7.1. With the procedure outlined in Appendix A.1, the only points where γ_{j_i} and δ_{j_i} are not smooth are vertices of $\mathcal{Q}_{j_i}$ and $\overline{\mathcal{Q}}_{j_i}$ respectively. Therefore, with the definition of $\sigma_{\delta_{j_i}}$ as in (7.14) and the use of the smooth, non-analytic function ζ from (7.11),

we see that $\sigma_{\delta_{j_i}}$ is smooth everywhere, since $\mathbf{x}^*_{j_i}$ does not belong in $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and δ_{j_i} is exactly 0 on the vertices of $\overline{\mathcal{Q}}_{j_i}$. Therefore, σ_{j_i} can only be non-smooth on the vertices of $\mathcal{Q}_{j_i}$ except for $\mathbf{x}^*_{j_i}$ (i.e., on the vertices of the triangle j_i), and on points where its denominator becomes zero. Since both $\sigma_{\gamma_{j_i}}$ and $\sigma_{\delta_{j_i}}$ vary between 0 and 1, this can only happen when $\sigma_{\gamma_{j_i}}(\mathbf{x}) = 1$ and $\sigma_{\delta_{j_i}}(\mathbf{x}) = 0$, i.e., only on $\mathbf{x}_{1j_i}$ and $\mathbf{x}_{2j_i}$. The fact that σ_{j_i} is smooth everywhere else derives immediately from the fact that $\sigma_{\delta_{j_i}}$ is a smooth function, and $\sigma_{\gamma_{j_i}}$ is smooth everywhere except for the triangle vertices. □

Proof of Lemma 7.2. From Lemma 7.1, we already know that the switch σ_{j_i} is smooth away from the vertices of j_i. On the other hand, the singular points of the deforming factor ν_{j_i} are the solutions of the equation $(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \mathbf{n}_{j_i} = 0$ and, therefore, lie on the hyperplane passing through $\mathbf{x}^*_{j_i}$ with normal vector $\mathbf{n}_{j_i}$and, due to the construction of $\overline{\mathcal{Q}}_{j_i}$ as in Definition 7.3, lie outside of $\overline{\mathcal{Q}}_{j_i}$ and do not affect the map $\mathcal{F}^{\mathcal{I}}_{map,j_i}$. Hence, the map $\mathbf{h}^{\mathcal{I}}_{j_i}$ is smooth everywhere in $\mathcal{F}^{\mathcal{I}}_{map,j_i}$, except for the vertices of the triangle j_i, as a composition of smooth functions with the same properties. □

Proof of Proposition 7.1. First of all, the map $\mathbf{h}^{\mathcal{I}}_{j_i}$ is smooth everywhere except for the vertices of the triangle j_i, as shown in Lemma 7.2. Therefore, in order to prove that $\mathbf{h}^{\mathcal{I}}_{j_i}$ is a C^∞ diffeomorphism away from the triangle vertices $\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$, we follow the procedure outlined in [131], also followed in [164], to show that

1. $\mathbf{h}^{\mathcal{I}}_{j_i}$ has a non-singular differential on $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ except for $\mathbf{x}_{1j_i}, \mathbf{x}_{2j_i}, \mathbf{x}_{3j_i}$.

2. $\mathbf{h}^{\mathcal{I}}_{j_i}$ preserves boundaries, i.e., $\mathbf{h}^{\mathcal{I}}_{j_i}(\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i}) \subset \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$, and $\partial_k \mathcal{F}$ the k-th connected component of the boundary of $\mathcal{F}$ with $\partial_0 \mathcal{F}$ the outer boundary of $\mathcal{F}$.

3. the boundary components of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ are pairwise homeomorphic, i.e., $\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i} \cong \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$.

We begin with Property 1 and examine the space away from the triangle vertices $\mathbf{x}_{1j_i}$, $\mathbf{x}_{2j_i}$, $\mathbf{x}_{3j_i}$. The case where $\sigma_{\delta_{j_i}}$ is 0 (outside of the polygonal collar $\overline{\mathcal{Q}}_{j_i}$) is not interesting, since $\mathbf{h}_{j_i}^{\mathcal{I}}$ defaults to the identity map and $D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}} = \mathbf{I}$. When $\sigma_{\delta_{j_i}}$ is not 0, we can compute the jacobian of the map as

$$\begin{aligned} D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}} = &\left(\nu_{j_i}(\mathbf{x}) - 1\right)(\mathbf{x} - \mathbf{x}_{j_i}^*)\nabla\sigma_{j_i}(\mathbf{x})^\top \\ &+ \sigma_{j_i}(\mathbf{x})(\mathbf{x} - \mathbf{x}_{j_i}^*)\nabla\nu_{j_i}(\mathbf{x})^\top \\ &+ \left[1 + \sigma_{j_i}(\mathbf{x})\left(\nu_{j_i}(\mathbf{x}) - 1\right)\right]\mathbf{I} \end{aligned} \tag{C.29}$$

For the deforming factor ν_{j_i} we compute from (7.16)

$$\nabla\nu_{j_i}(\mathbf{x}) = -\frac{\left(\mathbf{x}_{1j_i} - \mathbf{x}_{j_i}^*\right)^\top \mathbf{n}_{j_i}}{\left[\left(\mathbf{x} - \mathbf{x}_{j_i}^*\right)^\top \mathbf{n}_{j_i}\right]^2}\mathbf{n}_{j_i} \tag{C.30}$$

Note that we interestingly get

$$\left(\mathbf{x} - \mathbf{x}_{j_i}^*\right)^\top \nabla\nu_{j_i}(\mathbf{x}) = -\nu_{j_i}(\mathbf{x}) \tag{C.31}$$

From (C.29) it can be seen that $D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}} = \mathbf{A} + \mathbf{u}\mathbf{v}^\top$ with $\mathbf{A} = \left[1 + \sigma_{j_i}(\mathbf{x})\left(\nu_{j_i}(\mathbf{x}) - 1\right)\right]\mathbf{I}$, $\mathbf{u} = \mathbf{x} - \mathbf{x}_{j_i}^*$ and $\mathbf{v} = \left(\nu_{j_i}(\mathbf{x}) - 1\right)\nabla\sigma_{j_i}(\mathbf{x}) + \sigma_{j_i}(\mathbf{x})\nabla\nu_{j_i}(\mathbf{x})$.

Due to the fact that $0 \leq \sigma_{j_i}(\mathbf{x}) \leq 1$ and $0 < \nu_{j_i}(\mathbf{x}) < 1$ in the interior of an admissible polygonal collar $\overline{\mathcal{Q}}_{j_i}$ (see Definition 7.3), we get $1 + \sigma_{j_i}(\mathbf{x})\left(\nu_{j_i}(\mathbf{x}) - 1\right) > 0$. Hence, $\mathbf{A}$ is invertible, and by using the matrix determinant lemma and (C.31), the determinant of $D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}$ can be computed as

$$\begin{aligned} &\det(D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}) = \det\mathbf{A} + (\det\mathbf{A})\mathbf{v}^\top\mathbf{A}^{-1}\mathbf{u} \\ &= \left[1 + \sigma_{j_i}(\mathbf{x})\left(\nu_{j_i}(\mathbf{x}) - 1\right)\right]\cdot \\ &\cdot \left[(1 - \sigma_{j_i}(\mathbf{x})) + \left(\nu_{j_i}(\mathbf{x}) - 1\right)(\mathbf{x} - \mathbf{x}_{j_i}^*)^\top\nabla\sigma_{j_i}(\mathbf{x})\right] \end{aligned} \tag{C.32}$$

Similarly the trace of $D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}$ can be computed as

$$\begin{aligned}\operatorname{tr}(D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}) &= \left[1+\sigma_{j_i}(\mathbf{x})\left(\nu_{j_i}(\mathbf{x})-1\right)\right]+\left(1-\sigma_{j_i}(\mathbf{x})\right)\\ &+\left(\nu_{j_i}(\mathbf{x})-1\right)\left(\mathbf{x}-\mathbf{x}_{j_i}^{*}\right)^{\top}\nabla\sigma_{j_i}(\mathbf{x})\end{aligned} \tag{C.33}$$

Also, by construction of the switch σ_{j_i}, we see that $\nabla\sigma_{j_i}(\mathbf{x})=\mathbf{0}$ when $\sigma_{j_i}(\mathbf{x})=0$. Hence, using the above expressions, we can show that $\det(D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}), \operatorname{tr}(D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}})>0$ (and therefore establish that $D_{\mathbf{x}}\mathbf{h}_{j_i}^{\mathcal{I}}$ is not singular in the interior of $\overline{\mathcal{Q}}_{j_i}$, since $\mathcal{F}_{map,j_i}^{\mathcal{I}}\subseteq\mathbb{R}^2$) by showing that $\left(\mathbf{x}-\mathbf{x}_{j_i}^{*}\right)^{\top}\nabla\sigma_{j_i}(\mathbf{x})<0$ when $\sigma_{j_i}(\mathbf{x})>0$, where

$$\begin{aligned}\nabla\sigma_{j_i}(\mathbf{x}) =& \frac{\sigma_{\delta_{j_i}}(\mathbf{x})}{\left[\sigma_{\gamma_{j_i}}(\mathbf{x})\sigma_{\delta_{j_i}}(\mathbf{x})+\left(1-\sigma_{\gamma_{j_i}}(\mathbf{x})\right)\right]^2}\nabla\sigma_{\gamma_{j_i}}(\mathbf{x})\\ &+\frac{\sigma_{\gamma_{j_i}}(\mathbf{x})\left(1-\sigma_{\gamma_{j_i}}(\mathbf{x})\right)}{\left[\sigma_{\gamma_{j_i}}(\mathbf{x})\sigma_{\delta_{j_i}}(\mathbf{x})+\left(1-\sigma_{\gamma_{j_i}}(\mathbf{x})\right)\right]^2}\nabla\sigma_{\delta_{j_i}}(\mathbf{x})\end{aligned} \tag{C.34}$$

with

$$\nabla\sigma_{\gamma_{j_i}}(\mathbf{x})=\begin{cases}-\dfrac{\mu_{\gamma_{j_i}}\sigma_{\gamma_{j_i}}(\mathbf{x})}{\left(\epsilon_{j_i}-\gamma_{j_i}(\mathbf{x})\right)^2}\nabla\gamma_{j_i}(\mathbf{x}), & \gamma_{j_i}(\mathbf{x})<\epsilon_{j_i}\\ \mathbf{0}, & \gamma_{j_i}(\mathbf{x})\geq\epsilon_{j_i}\end{cases} \tag{C.35}$$

$$\nabla\sigma_{\delta_{j_i}}(\mathbf{x})=\begin{cases}\dfrac{\mu_{\delta_{j_i}}\sigma_{\delta_{j_i}}(\mathbf{x})}{\alpha_{j_i}(\mathbf{x})^2}\nabla\alpha_{j_i}(\mathbf{x}), & \delta_{j_i}(\mathbf{x})>0\\ \mathbf{0}, & \delta_{j_i}(\mathbf{x})\leq 0\end{cases} \tag{C.36}$$

and $\alpha_{j_i}(\mathbf{x}):=\delta_{j_i}(\mathbf{x})/||\mathbf{x}-\mathbf{x}_{j_i}^{*}||$. Therefore, it suffices to show that when $\sigma_{j_i}(\mathbf{x})>0$:

$$\left(\mathbf{x}-\mathbf{x}_{j_i}^{*}\right)^{\top}\nabla\gamma_{j_i}(\mathbf{x})>0 \tag{C.37}$$

$$\left(\mathbf{x}-\mathbf{x}_{j_i}^{*}\right)^{\top}\nabla\alpha_{j_i}(\mathbf{x})<0 \tag{C.38}$$

Following the procedure outlined in Appendix A.1 for generating implicit functions for polygons, it can be seen that the implicit function γ_{j_i}, describing the exterior of the quadri-

lateral $\mathcal{Q}_{j_i}$, can be written as

$$\gamma_{j_i}(\mathbf{x}) = \neg\left((\gamma_{1j_i}(\mathbf{x}) \wedge \gamma_{2j_i}(\mathbf{x})) \wedge (\gamma_{3j_i}(\mathbf{x}) \wedge \gamma_{4j_i}(\mathbf{x}))\right) \tag{C.39}$$

with the negation ($\neg$) and conjunction ($\wedge$) operators defined as in (A.3) and (A.4) respectively, and $\gamma_{1j_i}, \gamma_{2j_i}, \gamma_{3j_i}, \gamma_{4j_i}$ the hyperplane equations describing $\mathcal{Q}_{j_i}$ defined as follows

$$\gamma_{1j_i}(\mathbf{x}) := (\mathbf{x} - \mathbf{x}^*_{j_i})^\top \mathbf{n}_{1j_i}, \mathbf{n}_{1j_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}^*_{j_i} - \mathbf{x}_{1j_i}}{||\mathbf{x}^*_{j_i} - \mathbf{x}_{1j_i}||} \tag{C.40}$$

$$\gamma_{2j_i}(\mathbf{x}) := (\mathbf{x} - \mathbf{x}^*_{j_i})^\top \mathbf{n}_{2j_i}, \mathbf{n}_{2j_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}_{2j_i} - \mathbf{x}^*_{j_i}}{||\mathbf{x}_{2j_i} - \mathbf{x}^*_{j_i}||} \tag{C.41}$$

$$\gamma_{3j_i}(\mathbf{x}) := (\mathbf{x} - \mathbf{x}_{3j_i})^\top \mathbf{n}_{3j_i}, \mathbf{n}_{3j_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}_{3j_i} - \mathbf{x}_{2j_i}}{||\mathbf{x}_{3j_i} - \mathbf{x}_{2j_i}||} \tag{C.42}$$

$$\gamma_{4j_i}(\mathbf{x}) := (\mathbf{x} - \mathbf{x}_{3j_i})^\top \mathbf{n}_{4j_i}, \mathbf{n}_{4j_i} := \mathbf{R}_{\frac{\pi}{2}} \frac{\mathbf{x}_{1j_i} - \mathbf{x}_{3j_i}}{||\mathbf{x}_{1j_i} - \mathbf{x}_{3j_i}||} \tag{C.43}$$

We therefore get

$$\begin{aligned}\nabla\gamma_{j_i} = &-\left(1 - \frac{\gamma_{1j_i} \wedge \gamma_{2j_i}}{\sqrt{(\gamma_{1j_i} \wedge \gamma_{2j_i})^2 + (\gamma_{3j_i} \wedge \gamma_{4j_i})^2}}\right) \nabla(\gamma_{1j_i} \wedge \gamma_{2j_i}) \\ &-\left(1 - \frac{\gamma_{3j_i} \wedge \gamma_{4j_i}}{\sqrt{(\gamma_{1j_i} \wedge \gamma_{2j_i})^2 + (\gamma_{3j_i} \wedge \gamma_{4j_i})^2}}\right) \nabla(\gamma_{3j_i} \wedge \gamma_{4j_i})\end{aligned} \tag{C.44}$$

with

$$\begin{aligned}
\nabla(\gamma_{1j_i} \wedge \gamma_{2j_i}) &= \left(1 - \frac{\gamma_{1j_i}}{\sqrt{\gamma_{1j_i}^2+\gamma_{2j_i}^2}}\right) \nabla\gamma_{1j_i} \\
&\quad + \left(1 - \frac{\gamma_{2j_i}}{\sqrt{\gamma_{1j_i}^2+\gamma_{2j_i}^2}}\right) \nabla\gamma_{2j_i} \\
&= \left(1 - \frac{\gamma_{1j_i}}{\sqrt{\gamma_{1j_i}^2+\gamma_{2j_i}^2}}\right) \mathbf{n}_{1j_i} \\
&\quad + \left(1 - \frac{\gamma_{2j_i}}{\sqrt{\gamma_{1j_i}^2+\gamma_{2j_i}^2}}\right) \mathbf{n}_{2j_i} \qquad \text{(C.45)} \\
\nabla(\gamma_{3j_i} \wedge \gamma_{4j_i}) &= \left(1 - \frac{\gamma_{3j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) \nabla\gamma_{3j_i} \\
&\quad + \left(1 - \frac{\gamma_{4j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) \nabla\gamma_{4j_i} \\
&= \left(1 - \frac{\gamma_{3j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) \mathbf{n}_{3j_i} \\
&\quad + \left(1 - \frac{\gamma_{4j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) \mathbf{n}_{4j_i} \qquad \text{(C.46)}
\end{aligned}$$

It is then not hard to show that $(\mathbf{x}-\mathbf{x}_{j_i}^*)^\top \nabla(\gamma_{1j_i} \wedge \gamma_{2j_i}) = \gamma_{1j_i} \wedge \gamma_{2j_i}$. The term corresponding to $(\gamma_{3j_i} \wedge \gamma_{4j_i})$ is more complicated, but we can follow a similar procedure to get

$$\begin{aligned}
&(\mathbf{x} - \mathbf{x}_{j_i}^*)^\top \nabla(\gamma_{3j_i} \wedge \gamma_{4j_i}) = \gamma_{3j_i} \wedge \gamma_{4j_i} \\
&- \left(1 - \frac{\gamma_{3j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) (\mathbf{x}_{j_i}^* - \mathbf{x}_{3j_i})^\top \mathbf{n}_{3j_i} \\
&- \left(1 - \frac{\gamma_{4j_i}}{\sqrt{\gamma_{3j_i}^2+\gamma_{4j_i}^2}}\right) (\mathbf{x}_{j_i}^* - \mathbf{x}_{3j_i})^\top \mathbf{n}_{4j_i} \\
&< \gamma_{3j_i} \wedge \gamma_{4j_i} \qquad \text{(C.47)}
\end{aligned}$$

since $(\mathbf{x}_{j_i}^* - \mathbf{x}_{3j_i})^\top \mathbf{n}_{3j_i} > 0$ and $(\mathbf{x}_{j_i}^* - \mathbf{x}_{3j_i})^\top \mathbf{n}_{4j_i} > 0$, because $\mathcal{Q}_{j_i}$ is convex. Therefore, using the facts that $(\mathbf{x}-\mathbf{x}_{j_i}^*)^\top \nabla(\gamma_{1j_i} \wedge \gamma_{2j_i}) = \gamma_{1j_i} \wedge \gamma_{2j_i}$ and $(\mathbf{x}-\mathbf{x}_{j_i}^*)^\top \nabla(\gamma_{3j_i} \wedge \gamma_{4j_i}) < \gamma_{3j_i} \wedge \gamma_{4j_i}$,

we can get the desired result using (C.44) as follows

$$\begin{aligned}(\mathbf{x}-\mathbf{x}_{j_i}^*)^\top \nabla\gamma_{j_i}(\mathbf{x}) >& -\left((\gamma_{1j_i}\wedge\gamma_{2j_i})\wedge(\gamma_{3j_i}\wedge\gamma_{4j_i})\right)\\ =&\neg((\gamma_{1j_i}\wedge\gamma_{2j_i})\wedge(\gamma_{3j_i}\wedge\gamma_{4j_i}))\\ =&\gamma_{j_i}(\mathbf{x})>0\end{aligned} \tag{C.48}$$

The proof of (C.38) follows similar patterns. Here, we focus on δ_{j_i}. The external polygonal collar $\overline{\mathcal{Q}}_{j_i}$ can be assumed to have n sides, which means that we can write $\delta_{j_i} = ((\delta_{1j_i}\wedge\delta_{2j_i})\wedge\ldots\wedge\delta_{nj_i})$. Following the procedure outlined above for the proof of (C.37), we can expand each term in the conjunction individually and then combine them to get

$$(\mathbf{x}-\mathbf{x}_{j_i}^*)^\top\nabla\delta_{j_i}(\mathbf{x}) < \delta_{j_i}(\mathbf{x}) \tag{C.49}$$

We also have

$$\begin{aligned}\nabla\alpha_{j_i}(\mathbf{x}) &= \nabla\left(\frac{\delta_{j_i}(\mathbf{x})}{||\mathbf{x}-\mathbf{x}_{j_i}^*||}\right)\\ &= \frac{||\mathbf{x}-\mathbf{x}_{j_i}^*||\nabla\delta_{j_i}(\mathbf{x})-\delta_{j_i}(\mathbf{x})\frac{\mathbf{x}-\mathbf{x}_{j_i}^*}{||\mathbf{x}-\mathbf{x}_{j_i}^*||}}{||\mathbf{x}-\mathbf{x}_{j_i}^*||^2}\end{aligned} \tag{C.50}$$

which gives the desired result using (C.49)

$$(\mathbf{x}-\mathbf{x}_{j_i}^*)\nabla\alpha_{j_i}(\mathbf{x}) = \frac{(\mathbf{x}-\mathbf{x}_{j_i}^*)\nabla\delta_{j_i}(\mathbf{x})-\delta_{j_i}(\mathbf{x})}{||\mathbf{x}-\mathbf{x}_{j_i}^*||} < 0 \tag{C.51}$$

This concludes the proof that $\mathbf{h}_{j_i}^{\mathcal{I}}$ satisfies Property 1.

Next, we focus on Property 2. Pick a point $\mathbf{x}\in\partial_k\mathcal{F}_{map,j_i}^{\mathcal{I}}$. This point could lie:

1. on the outer boundary of $\mathcal{F}_{map,j_i}^{\mathcal{I}}$ and away from P_i
2. on the boundary of one of the $|\mathcal{J}_{\mathcal{C}}|$ unknown but visible convex obstacles
3. on the boundary of one of the $(|\mathcal{J}_{\mathcal{D}}^{\mathcal{I}}|+|\mathcal{J}_{\mathcal{B}}^{\mathcal{I}}|-1)$ familiar obstacles that are not P_i

4. on the boundary of P_i but not on the boundary of the triangle j_i

5. on the boundary of the triangle j_i

In the first four cases, we have $\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) = \mathbf{x}$, whereas in the last case, we have

$$\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) = \mathbf{x}^*_{j_i} + \frac{\left(\mathbf{x}_{1j_i} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}}{\left(\mathbf{x} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}}(\mathbf{x} - \mathbf{x}^*_{j_i}) \tag{C.52}$$

It can be verified that $\left(\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) - \mathbf{x}_{1j_i}\right)^\top \mathbf{n}_{j_i} = 0$, which means that $\mathbf{x}$ is sent to the shared hyperplane between j_i and $p(j_i)$ as desired. This shows that we always have $\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) \in \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ and the map satisfies Property 2.

Finally, Property 3 derives from above and the fact that each boundary segment $\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i}$ is an one-dimensional manifold, the boundary of either a convex set or a polygon, both of which are homeomorphic to S^1 and, therefore, the corresponding boundary $\partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$. □

Proof of Lemma 7.3. The proof follows similar patterns with the proof of Lemma 7.1. With the procedure outlined in Appendix A.1, the only points where γ_{r_i} and δ_{r_i} are not smooth are vertices of $\mathcal{Q}_{r_i}$ and $\overline{\mathcal{Q}}_{r_i}$ respectively. Therefore, with the definition of $\sigma_{\delta_{r_i}}$ as in (7.22) and the use of the smooth, non-analytic function ζ from (7.11), we see that $\sigma_{\delta_{r_i}}$ is smooth everywhere, since $\mathbf{x}^*_i$ does not belong in $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and δ_{r_i} is exactly 0 on the vertices of $\overline{\mathcal{Q}}_{r_i}$. Therefore, σ_{r_i} can only be non-smooth on the vertices of $\mathcal{Q}_{r_i}$ (i.e., on the vertices of the triangle r_i), and on points where its denominator becomes zero. Since both $\sigma_{\gamma_{r_i}}$ and $\sigma_{\delta_{r_i}}$ vary between 0 and 1, this can only happen when $\sigma_{\gamma_{r_i}}(\mathbf{x}) = 1$ and $\sigma_{\delta_{r_i}}(\mathbf{x}) = 0$, which is not allowed by Definition 7.5, requiring $\mathcal{Q}_{r_i} \subset \overline{\mathcal{Q}}_{r_i}$. The fact that σ_{r_i} is smooth everywhere else derives immediately from the fact that $\sigma_{\delta_{r_i}}$ is a smooth function, and $\sigma_{\gamma_{r_i}}$ is smooth everywhere except for the triangle vertices. □

Proof of Proposition 7.2. The proof follows similar patterns with that of Proposition 7.1. As shown in Lemma 7.5, the map $\hat{\mathbf{h}}^{\mathcal{I}}$ is smooth in $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ away from any sharp corners.

Therefore, we need to focus again on the Massey conditions [131] and show that

1. $\hat{\mathbf{h}}^{\mathcal{I}}$ has a non-singular differential on $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ away from any sharp corners.

2. $\hat{\mathbf{h}}^{\mathcal{I}}$ preserves boundaries, i.e., $\hat{\mathbf{h}}^{\mathcal{I}}(\partial_k \hat{\mathcal{F}}^{\mathcal{I}}_{map}) \subset \partial_k \mathcal{F}^{\mathcal{I}}_{model}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$.

3. the boundary components of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and $\mathcal{F}^{\mathcal{I}}_{model}$ are pairwise homeomorphic, i.e., $\partial_k \hat{\mathcal{F}}^{\mathcal{I}}_{map} \cong \partial_k \mathcal{F}^{\mathcal{I}}_{model}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$.

We begin with Property 1 and examine the space away from any sharp corners in $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$. By construction of the polygonal collars $\overline{\mathcal{Q}}_{r_i}$ and the definition of $\hat{\mathbf{h}}^{\mathcal{I}}$ in (7.28), we see that $\hat{\mathbf{h}}^{\mathcal{I}}$ is either the identity map (which implies that $D_{\mathbf{x}}\hat{\mathbf{h}}^{\mathcal{I}} = \mathbf{I}$), or depends only on a single switch σ_{r_k}. In that case, we can isolate the k-th term of the map jacobian to write

$$\begin{aligned} D_{\mathbf{x}}\hat{\mathbf{h}}^{\mathcal{I}} = D_{\mathbf{x}}\hat{\mathbf{h}}^{\mathcal{I}}|_k = &\left(\nu_{r_k}(\mathbf{x}) - 1\right)(\mathbf{x} - \mathbf{x}^*_k)\nabla\sigma_{r_k}(\mathbf{x})^\top \\ &+ \sigma_{r_k}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*_k)\nabla\nu_{r_k}(\mathbf{x})^\top \\ &+ \left[1 + \sigma_{r_k}(\mathbf{x})\left(\nu_{r_k}(\mathbf{x}) - 1\right)\right]\mathbf{I} \end{aligned} \tag{C.53}$$

It is then straightforward to follow exactly the same procedure outlined in the proof of Proposition 7.1 and show that $\det(D_{\mathbf{x}}\hat{\mathbf{h}}^{\mathcal{I}}|_k), \mathrm{tr}(D_{\mathbf{x}}\hat{\mathbf{h}}^{\mathcal{I}}|_k) > 0$ for all $\mathbf{x} \in \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ away from sharp corners.

Next, we focus on Property 2. Pick a point $\mathbf{x} \in \partial_k \hat{\mathcal{F}}^{\mathcal{I}}_{map}$. This point could lie

1. on the outer boundary of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$, but not on a root triangle corresponding to an obstacle $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$

2. on the boundary of one of the $\mathcal{J}_{\mathcal{C}}$ unknown but visible convex obstacles

3. on the outer boundary of $\hat{\mathcal{F}}^{\mathcal{I}}_{map}$ and on a root triangle corresponding to an obstacle $P_i \in \mathcal{B}^{\mathcal{I}}_{map}$, or

4. on the boundary of one of the $|\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}|$ root triangles corresponding to obstacles in $\mathcal{D}^{\mathcal{I}}_{map}$

In the first two cases, we have $\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) = \mathbf{x}$, in the third case we have $\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) \in \partial_0 \mathcal{F}^{\mathcal{I}}_{model} = \partial \mathcal{F}_e$ by construction of (7.23) and (7.25), while in the last case

$$\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) = \mathbf{x}^*_k + \frac{\rho_k}{||\mathbf{x} - \mathbf{x}^*_k||}(\mathbf{x} - \mathbf{x}^*_k) \tag{C.54}$$

for some $k \in \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, sending $\mathbf{x}$ to the boundary of the k-th disk in $\mathcal{F}^{\mathcal{I}}_{model}$. This shows that we always have $\hat{\mathbf{h}}^{\mathcal{I}}(\mathbf{x}) \in \partial_k \mathcal{F}^{\mathcal{I}}_{model}$ and the map satisfies Property 2.

Finally, Property 3 derives from above and the fact that each boundary segment $\partial_k \hat{\mathcal{F}}^{\mathcal{I}}_{map}$ is an one-dimensional manifold, the boundary of either a convex set or a triangle, both of which are homeomorphic to S^1 and, therefore, the corresponding boundary $\partial_k \mathcal{F}^{\mathcal{I}}_{model}$. □

C.5.2 Proofs of Results in Section 7.4

Proof of Lemma 7.6. The proof of this lemma derives immediately from [7, Propositions 5,11] and Assumption 7.5, from which we can infer that the set of stationary points of the vector field $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}} \cdot \mathbf{u}^{\mathcal{I}}(\mathbf{x})$, defined on $\mathcal{F}^{\mathcal{I}}_{model}$, is $\{\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)\} \bigcup \{\mathbf{s}_i\}_{i \in \mathcal{J}^{\mathcal{I}}_{\mathcal{D}}} \bigcup \{\mathcal{G}_k\}_{k \in \mathcal{J}_{\mathcal{C}}}$, with $\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$ being a locally stable equilibrium of $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}} \cdot \mathbf{u}^{\mathcal{I}}(\mathbf{x})$ and each other point being a nondegenerate saddle, since [7, Assumption 2] is satisfied for the obstacles in $\mathcal{F}^{\mathcal{I}}_{model}$ from Assumption 7.1. To complete the proof, we just have to note that the index of an isolated zero of a vector field does not change under diffeomorphisms of the domain [78]. □

Proof of Proposition 7.4. Consider the smooth Lyapunov function candidate $V^{\mathcal{I}}(\mathbf{x}) =$

$||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||^2$. Using (7.40) and writing $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ and $\mathbf{y}_d = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$, we get

$$\begin{aligned}
\frac{dV^{\mathcal{I}}}{dt} =& 2(\mathbf{y} - \mathbf{y}_d)^\top (\mathbf{D}_{\mathbf{x}} \mathbf{h}^{\mathcal{I}}) \dot{\mathbf{x}} \\
=& -2k(\mathbf{y} - \mathbf{y}_d)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
=& -2k \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d) + \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d) - \mathbf{y}_d\right)^\top \\
& \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
=& -2k||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 \\
& + 2k \left(\mathbf{y}_d - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
\leq& -2k||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 \leq 0
\end{aligned} \tag{C.55}$$

since $\mathbf{y} \in \mathcal{LF}(\mathbf{y})$, which implies that

$$\left(\mathbf{y}_d - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \leq 0 \tag{C.56}$$

since either $\mathbf{y}_d = \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)$, or $\mathbf{y}_d$ and $\mathbf{y}$ are separated by a hyperplane passing through $\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)$. Therefore, similarly to [7], using LaSalle's invariance principle we see that every trajectory starting in $\mathcal{F}^{\mathcal{I}}_{map}$ approaches the largest invariant set in $\{\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map} \,|\, \dot{V}^{\mathcal{I}}(\mathbf{x}) = 0\}$, i.e. the equilibrium points of (7.40). The desired result follows from Lemma 7.6, since $\mathbf{x}_d$ is the only locally stable equilibrium of our control law and the rest of the stationary points are nondegenerate saddles, whose regions of attraction have empty interior in $\mathcal{F}^{\mathcal{I}}_{map}$. □

Proof of Proposition 7.5. Note that the jacobian of $\overline{\mathbf{h}}^{\mathcal{I}}$ will be given by

$$D_{\bar{\mathbf{x}}} \overline{\mathbf{h}}^{\mathcal{I}} = \left[\begin{array}{c|c} D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}} & \mathbf{0}_{2\times 1} \\ \hline D_{\mathbf{x}} \xi^{\mathcal{I}} & \dfrac{\partial \xi^{\mathcal{I}}}{\partial \psi} \end{array}\right] \tag{C.57}$$

Since we already have from Corollary 7.3 that $D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}}$ is non-singular, it suffices to show that

$\frac{\partial \xi^{\mathcal{I}}}{\partial \psi} \neq 0$ for all $\overline{\mathbf{x}} \in \mathcal{F}^{\mathcal{I}}_{map} \times S^1$. From (7.43) we can derive

$$\frac{\partial \xi^{\mathcal{I}}}{\partial \psi} = \frac{\det(D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}})}{||\mathbf{e}(\overline{\mathbf{x}})||^2} \tag{C.58}$$

Therefore, we immediately get that $\frac{\partial \xi^{\mathcal{I}}}{\partial \psi} \neq 0$ for all $\overline{\mathbf{x}} \in \mathcal{F}^{\mathcal{I}}_{map} \times S^1$ since $\det(D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}}) \neq 0$ and $||\mathbf{e}(\overline{\mathbf{x}})|| \neq 0$ for all $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map}$, because $D_{\mathbf{x}} \mathbf{h}^{\mathcal{I}}$ is non-singular on $\mathcal{F}^{\mathcal{I}}_{map}$. This implies that $D_{\overline{\mathbf{x}}} \overline{\mathbf{h}}^{\mathcal{I}}$ is non-singular on $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$.

Next, we note that $\partial \left(\mathcal{F}^{\mathcal{I}}_{map} \times S^1 \right) = \partial \mathcal{F}^{\mathcal{I}}_{map} \times S^1$, since S^1 is a manifold without boundary. Similarly, $\partial \left(\mathcal{F}^{\mathcal{I}}_{model} \times S^1 \right) = \partial \mathcal{F}^{\mathcal{I}}_{model} \times S^1$. Hence, we can easily complete the proof following a similar procedure with the end of the proofs of Propositions 7.1 and 7.2 to show that $\overline{\mathbf{h}}^{\mathcal{I}}$ preserves boundaries, and the boundaries of $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ and $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$ are pairwise homeomorphic. □

Proof of Theorem 7.2. We have already established that $||\mathbf{e}(\overline{\mathbf{x}})||$ and $\frac{\partial \xi^{\mathcal{I}}}{\partial \psi}$ are nonzero for all $\overline{\mathbf{x}} \in \mathcal{F}^{\mathcal{I}}_{map} \times S^1$ in the proof of Proposition 7.5, which implies that v and ω can have no singular points. Also notice that $||\mathbf{e}(\overline{\mathbf{x}})||$, $\frac{\partial \xi^{\mathcal{I}}}{\partial \psi}$ and $D_{\mathbf{x}} \xi^{\mathcal{I}} \begin{bmatrix} \cos\psi & \sin\psi \end{bmatrix}^\top$ are all smooth away from corners in $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$. Hence, the uniqueness and existence of the flow generated by control law (7.54) can be established similarly to [7] through the flow properties of the controller in [12] (that we use here in (7.55)) and the facts that metric projections onto moving convex cells are piecewise continuously differentiable [115, 175], and the composition of piecewise continuously differentiable functions is piecewise continuously differentiable and, therefore, locally Lipschitz [36].

Next, as shown in (7.45), the vector field $\mathbf{B}(\psi)\overline{\mathbf{u}}^{\mathcal{I}}$ on $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ is the pullback of the complete vector field $\mathbf{B}(\varphi)\overline{\mathbf{v}}^{\mathcal{I}}$, guaranteed to retain $\mathcal{F}^{\mathcal{I}}_{model} \times S^1$ positively invariant under its flow as shown in [7], under the smooth change of coordinates $\overline{\mathbf{h}}^{\mathcal{I}}$ away from sharp corners in $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$. This shows that the freespace $\mathcal{F}^{\mathcal{I}}_{map} \times S^1$ is positively invariant under law (7.54).

Finally, consider the smooth Lyapunov function candidate $V^{\mathcal{I}}(\mathbf{x}) = ||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||^2$.

Then, by writing $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ and $\mathbf{y}_d = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$, we get

$$\begin{aligned}
\frac{dV^{\mathcal{I}}}{dt} =& 2(\mathbf{y}-\mathbf{y}_d)^\top(\mathbf{D}_{\mathbf{x}}\mathbf{h}^{\mathcal{I}})\dot{\mathbf{x}} \\
=& 2v^{\mathcal{I}}\,(\mathbf{y}-\mathbf{y}_d)^\top(\mathbf{D}_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}) \begin{bmatrix}\cos\psi \\ \sin\psi\end{bmatrix} \\
=& 2\hat{v}^{\mathcal{I}}(\mathbf{y}-\mathbf{y}_d)^\top \begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix} \\
=& -2k_v(\mathbf{y}-\mathbf{x}_d)^\top \begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix}\begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix}^\top \\
& \Big(\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big) \\
=& -2k_v(\mathbf{y}-\mathbf{y}_d)^\top\Big(\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big)
\end{aligned}$$

since $\begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix}\begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix}^\top$ is just the projection operator on the line defined by the vector $\begin{bmatrix}\cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) & \sin\xi^{\mathcal{I}}(\overline{\mathbf{x}})\end{bmatrix}^\top$, with which $\Big(\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big)$ is already parallel. Following this result, we get

$$\frac{dV^{\mathcal{I}}}{dt} \leq -2k_v\Big|\Big|\mathbf{h}^{\mathcal{I}}(\mathbf{y})-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big|\Big|^2 \leq 0 \tag{C.59}$$

since, similarly to the proof of Proposition 7.4, we have

$$\Big(\mathbf{y}_d-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big)^\top\Big(\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})\cap H_\parallel}(\mathbf{y}_d)\Big) \leq 0 \tag{C.60}$$

Therefore, using LaSalle's invariance principle, we see that every trajectory starting in $\mathcal{F}^{\mathcal{I}}_{map}\times S^1$ approaches the largest invariant set in $\{(\mathbf{x},\psi)\in\mathcal{F}^{\mathcal{I}}_{map}\times S^1 \,|\, \dot{V}^{\mathcal{I}}(\mathbf{x})=0\} = \{(\mathbf{x},\psi)\in\mathcal{F}^{\mathcal{I}}_{map}\times S^1 \,|\, \mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}))\cap H_\parallel}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))\}$. At the same time, we know from (7.55) that $\mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}))\cap H_\parallel}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))$ implies $v^{\mathcal{I}}=0$. From (7.54), for $v^{\mathcal{I}}=0$, we get that $\omega^{\mathcal{I}}$ will be zero at points where $\hat{\omega}^{\mathcal{I}}$ is zero, i.e. at points $(\mathbf{x},\psi)\in\mathcal{F}^{\mathcal{I}}_{map}\times S^1$

where

$$\begin{bmatrix} -\sin\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \end{bmatrix}^{\top} \left(\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{y}_{d,G}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \xi^{\mathcal{I}}(\psi)\right) = 0 \tag{C.61}$$

with the angular local goal $\mathbf{y}_{d,G}$ defined as in (7.57). Therefore the largest invariant set in $\{(\mathbf{x},\psi) \,|\, \mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}))\cap H_{\|}}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))\}$ is the set of points $\overline{\mathbf{x}} = (\mathbf{x},\psi)$ where the following two conditions are satisfied

$$\mathbf{h}^{\mathcal{I}}(\mathbf{x}) = \Pi_{\mathcal{LF}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}))\cap H_{\|}}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)) \tag{C.62}$$

$$\begin{bmatrix} -\sin\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \\ \cos\xi^{\mathcal{I}}(\overline{\mathbf{x}}) \end{bmatrix}^{\top} \left(\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{y}_{d,G}(\mathbf{h}^{\mathcal{I}}(\mathbf{x}), \xi^{\mathcal{I}}(\psi))\right) = 0 \tag{C.63}$$

Using a similar argument to [7, Proposition 12], we can, therefore, verify that the set of stationary points of law (7.54) is given by

$$\begin{aligned} &\{\mathbf{x}_d\} \times (-\pi, \pi] \\ &\bigcup \left\{ (\mathbf{q},\psi) \,\middle|\, \mathbf{q} \in \{(\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{s}_i)\}_{i\in\mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \bigcup_{k\in\mathcal{J}_{\mathcal{C}}} \mathcal{G}_k, \right. \\ &\left. \begin{bmatrix} -\sin\xi^{\mathcal{I}}(\mathbf{q},\psi) \\ \cos\xi^{\mathcal{I}}(\mathbf{q},\psi) \end{bmatrix}^{\top} (\mathbf{q} - \mathbf{x}_d) = 0 \right\} \end{aligned} \tag{C.64}$$

using (7.41). We can then invoke a similar argument to Proposition 7.4 to show that $\mathbf{x}_d$ locally attracts with any orientation ψ, while any configuration associated with any other equilibrium point is a nondegenerate saddle whose stable manifold is a set of measure zero, and the result follows. □

Proof of Lemma 7.7. We can show this by contradiction. Assume that the robot is in mode $\mathcal{I}$ and two guards $G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_1}$ and $G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_2}$, indexed by two different subsets $\mathcal{I}_1 \neq \mathcal{I}_2$, each playing the role of $\mathcal{I}_u$ in (7.31), nevertheless overlap, $G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_1} \cap G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_2} \neq \varnothing$. That means that there exists at least one state $\mathbf{x} \in \mathcal{F}^{\mathcal{I}}$, such that $\mathbf{x} \in G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_1}$ and $\mathbf{x} \in G^{\mathcal{I},\mathcal{I}\cup\mathcal{I}_2}$, for

two nonempty sets $\mathcal{I}_1, \mathcal{I}_2$ with $\mathcal{I} \cap \mathcal{I}_1 = \varnothing$, $\mathcal{I} \cap \mathcal{I}_2 = \varnothing$. Since $\mathcal{I}_1 \neq \mathcal{I}_2$ by assumption, this implies that there exists at least one index i that is contained in one of these index sets, but is not contained in the other. Without loss of generality, assume that $i \in \mathcal{I}_1$ and $i \notin \mathcal{I}_2$. We immediately arrive at a contradiction, since the requirement $\mathbf{x} \in G^{\mathcal{I}, \mathcal{I} \cup \mathcal{I}_2}$ requires $\overline{\mathsf{B}(\mathbf{x}, R)} \cap \tilde{\mathcal{P}}_{\mathcal{N}_\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{I}_2)} = \varnothing$, but we know that the requirement $\mathbf{x} \in G^{\mathcal{I}, \mathcal{I} \cup \mathcal{I}_1}$ implies $\overline{\mathsf{B}(\mathbf{x}, R)} \cap \tilde{P}_i \neq \varnothing$ with $\tilde{P}_i \in \tilde{\mathcal{P}}_{\mathcal{N}_\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{I}_2)}$. □

Proof of Lemma 7.8. If the system is in the terminal mode $\mathcal{I} = \mathcal{N}_\mathcal{P}$, according to Definition 7.8, then finite time escape through the boundary of the hybrid domain is not possible, since the vector field $\mathbf{u}^\mathcal{I}$ leaves its domain positively invariant under its flow, as described in Theorem 7.1. For $\mathcal{I} \neq \mathcal{N}_\mathcal{P}$, the only way in which the flow can escape is through the boundary of an obstacle $\tilde{P} \notin \{\tilde{P}_i\}_{i \in \mathcal{I}}$, since $\mathbf{u}^\mathcal{I}$ guarantees safety only against familiar obstacles in $\mathcal{I}$ and any unknown obstacles encountered along the way. We are going to show that this cannot happen by contradiction. Assume that at time t_0 the robot is at $\mathbf{x}_0 \in \mathcal{F}^\mathcal{I}$, and at time $t_1 > t_0$ it crosses the boundary of an obstacle $\tilde{P} \notin \{\tilde{P}_i\}_{i \in \mathcal{I}}$. This means that the robot travels distance $d > 0$ between t_0 and t_1 in mode $\mathcal{I}$, without triggering a transition to another hybrid mode $\mathcal{I}'$ that includes $\tilde{P}$ (and therefore guarantees safety against it by Theorem 7.1), which is impossible since the sensor footprint has a positive radius R and $\overline{\mathsf{B}(\mathbf{x}^t, R)}$ would have hit $\tilde{P}_i$ at some time $t < t_1$ before colliding with it.

Moreover, the restriction of the reset map in each separate mode is just the identity transform, which, by the argument made above, implies that the discrete transition itself is never blocking, assuming that the initial condition lies in the freespace $\mathcal{F}$. This is because $\mathcal{F} \subseteq \mathcal{F}^\mathcal{I}$ for all modes $\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}$.

Finally, hybrid ambiguity is avoided by the construction of the guard in (7.31); if the robot at a position $\mathbf{x}^-$ in the interior of the domain is in mode $\mathcal{I}$ at time t^- before a discrete transition and in mode $\mathcal{I}'$ at time t^+ after the transition, we are guaranteed that the sensor footprint $\overline{\mathsf{B}(\mathbf{x}^+, R)}$ after the transition does not intersect any obstacle $\tilde{P}_i$ with $i \notin \mathcal{I}'$. This implies that $\mathbf{x}^+$ lies in the interior of the domain and away from the guard, and the application of the reset map provides the unique extension to the execution. □

Proof of Theorem 7.3. As stated in Section 7.4.1, the hybrid system described in the tuple H has $2^{|\mathcal{N}_\mathcal{P}|}$ modes. Unique piecewise continuous differentiability of the flow derives immediately by the unique continuous differentiability of the control law $U^\mathcal{I}$, defined in (7.40) for each separate mode $\mathcal{I}$, as summarized in Theorem 7.1. Moreover, positive invariance derives from the first part of the proof of Lemma 7.8, which guarantees that the hybrid flow cannot escape from the hybrid domain through a point on the boundary of the domain in the complement of the guard, or the guard itself.

For stability, we note that each mode (indexed by $\mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}$) is associated with a candidate Lyapunov function $V^\mathcal{I}(\mathbf{x}) = ||\mathbf{h}^\mathcal{I}(\mathbf{x}) - \mathbf{h}^\mathcal{I}(\mathbf{x}_d)||^2$, as shown in the proof of Proposition 7.4. Moreover, also by the results of Proposition 7.4, $\mathbf{x}_d$ is the unique asymptotically stable equilibrium of each control vector field $U^\mathcal{I}$, thus, almost every execution that remains in mode $\mathcal{I}$ for all future time has a trajectory that asymptotically approaches the goal. Then, the key for the proof is the observation that once the robot exits a mode defined by $\mathcal{I}$, it can never re-enter it. This is because the robot stores information in its semantic map and this knowledge can only be incremental; in the worst case, the robot will explore all familiar obstacles in the environment, and stay in mode $\mathcal{I} = \mathcal{N}_\mathcal{P}$ for all following time.

Based on this observation, we notice that the collection of functions $\{V^\mathcal{I} \,|\, \mathcal{I} \in 2^{\mathcal{N}_\mathcal{P}}\}$ are *Lyapunov-like*, in the sense of [32, Definition 2.2], for all time their corresponding mode is active, since they never reset. We complete the proof by invoking [32, Theorem 2.3], which states that if a collection of Lyapunov-like functions for a hybrid system are associated with corresponding vector fields that share the same equilibrium, then the hybrid system itself is Lyapunov stable around this equilibrium. □

C.6 Proofs of Results in Chapter 8

Proof of Proposition 8.1. We follow similar patterns to the proof of Proposition 7.1. We first need to show that the functions $\sigma_{j_i}, \nu_{j_i} : \mathcal{F}^\mathcal{I}_{map,j_i} \to \mathbb{R}$ are smooth away from the polygon vertices, none of which lies in the interior of $\mathcal{F}^\mathcal{I}_{map,j_i}$. We begin with σ_{j_i}. First of all, with the procedure outlined in Appendix A.1, the only points where γ_{j_i} and δ_{j_i} are not smooth

are vertices of $\mathcal{Q}_{j_i}$ and $\overline{\mathcal{Q}}_{j_i}$ respectively. Therefore, by defining σ_{j_i} as in (7.15), we get that σ_{j_i} can only be non-smooth on the vertices of $\mathcal{Q}_{j_i}$ except for $\mathbf{x}^*_{j_i}$ (i.e., on the vertices of the polygon j_i), and on points where its denominator becomes zero. Since both $\sigma_{\gamma_{j_i}}$ and $\sigma_{\delta_{j_i}}$ vary between 0 and 1, this can only happen when $\sigma_{\gamma_{j_i}}(\mathbf{x}) = 1$ and $\sigma_{\delta_{j_i}}(\mathbf{x}) = 0$, i.e., only on $\mathbf{x}_{1j_i}$ and $\mathbf{x}_{2j_i}$. The fact that σ_{j_i} is smooth everywhere else derives immediately from the fact that $\sigma_{\delta_{j_i}}$ is a smooth function, and $\sigma_{\gamma_{j_i}}$ is smooth everywhere except for the polygon vertices.

On the other hand, the singular points of the deforming factor ν_{j_i}, defined in (7.16), are the solutions of the equation $(\mathbf{x}-\mathbf{x}^*_{j_i})^\top \mathbf{n}_{j_i} = 0$, which lie on the hyperplane passing through $\mathbf{x}^*_{j_i}$ with normal vector $\mathbf{n}_{j_i}$ and, due to the construction of $\overline{\mathcal{Q}}_{j_i}$ as in Definition 8.2, lie outside of $\overline{\mathcal{Q}}_{j_i}$ and do not affect the map $\mathcal{F}^{\mathcal{I}}_{map,j_i}$. Hence, the map $\mathbf{h}^{\mathcal{I}}_{j_i}$ is smooth everywhere in $\mathcal{F}^{\mathcal{I}}_{map,j_i}$, except for the vertices of the polygon j_i, as a composition of smooth functions with the same properties.

Now, in order to prove that $\mathbf{h}^{\mathcal{I}}_{j_i}$ is a C^∞ diffeomorphism away from the vertices of j_i, we follow the procedure outlined in [131], also followed in [164] and in the proof of Proposition 7.1, to show that

1. $\mathbf{h}^{\mathcal{I}}_{j_i}$ has a non-singular differential on $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ except for the vertices of polygon j_i.

2. $\mathbf{h}^{\mathcal{I}}_{j_i}$ preserves boundaries, i.e., $\mathbf{h}^{\mathcal{I}}_{j_i}(\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i}) \subset \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$, and $\partial_k \mathcal{F}$ the k-th connected component of the boundary of $\mathcal{F}$ with $\partial_0 \mathcal{F}$ the outer boundary of $\mathcal{F}$.

3. the boundary components of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and $\mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ are pairwise homeomorphic, i.e., $\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i} \cong \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$, with k spanning both the indices of familiar obstacles $\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}$, $\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}$ as well as the indices of unknown obstacles $\mathcal{J}_{\mathcal{C}}$.

We begin with Property 1 and examine the space away from the vertices of j_i. The case where $\sigma_{\delta_{j_i}}$ is 0 (outside of the polygonal collar $\overline{\mathcal{Q}}_{j_i}$) is not interesting, since $\mathbf{h}^{\mathcal{I}}_{j_i}$ defaults to the identity map and $D_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}_{j_i} = \mathbf{I}$. When $\sigma_{\delta_{j_i}}$ is not 0, we can follow the same procedure outlined

in the proof of Proposition 7.1 to establish that it suffices to show that when $\sigma_{j_i}(\mathbf{x}) > 0$:

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla \gamma_{j_i}(\mathbf{x}) > 0 \tag{C.65}$$

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla \alpha_{j_i}(\mathbf{x}) < 0 \tag{C.66}$$

Following the procedure outlined in Appendix A.1 for the implicit representation of polygonal obstacles and assuming that the polygon $\mathcal{Q}_{j_i}$ has m sides, we can describe $\mathcal{Q}_{j_i}$ with the implicit function $\gamma_{j_i} = \neg\left((\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \ldots \wedge \gamma_{mj_i}\right)$, with the companion R-function [176] of the logic negation for a function x defined as $\neg x := -x$, the companion R-function of the logic conjunction $\wedge$ for two functions x_1, x_2 defined as $x_1 \wedge x_2 := x_1 + x_2 - (x_1^p + x_2^p)^{\frac{1}{p}}$, and γ_{kj_i} the k-th hyperplane equation describing $\mathcal{Q}_{j_i}$, given as $\gamma_{kj_i}(\mathbf{x}) := (\mathbf{x} - \mathbf{x}_{kj_i})^\top \mathbf{n}_{kj_i}$. Note here that the first two hyperplanes γ_{1j_i} and γ_{2j_i} pass through the center $\mathbf{x}^*_{j_i}$, i.e., we can write $\gamma_{1j_i}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*_{j_i})^\top \mathbf{n}_{1j_i}$ and $\gamma_{2j_i}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*_{j_i})^\top \mathbf{n}_{2j_i}$. Based on this observation, it is easy to derive the following expression for any $\mathbf{x}$ that satisfies $\sigma_{j_i}(\mathbf{x}) > 0$

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla(\gamma_{1j_i} \wedge \gamma_{2j_i}) = \gamma_{1j_i} \wedge \gamma_{2j_i} \tag{C.67}$$

We can then similarly compute

$$\nabla\left((\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \gamma_{3j_i}\right) = \left(1 - \frac{\gamma_{3j_i}}{\sqrt{(\gamma_{1j_i} \wedge \gamma_{2j_i})^2 + \gamma_{3j_i}^2}}\right) \nabla \gamma_{3j_i}$$
$$+ \left(1 - \frac{\gamma_{1j_i} \wedge \gamma_{2j_i}}{\sqrt{(\gamma_{1j_i} \wedge \gamma_{2j_i})^2 + \gamma_{3j_i}^2}}\right) \nabla(\gamma_{1j_i} \wedge \gamma_{2j_i})$$

and observe that $(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla \gamma_{3j_i} = (\mathbf{x} - \mathbf{x}_{3j_i})^\top \nabla \gamma_{3j_i} - (\mathbf{x}^*_{j_i} - \mathbf{x}_{3j_i})^\top \nabla \gamma_{3j_i} = \gamma_{3j_i} - (\mathbf{x}^*_{j_i} - \mathbf{x}_{3j_i})^\top \mathbf{n}_{3j_i} < \gamma_{3j_i}$, which implies that $(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla\left((\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \gamma_{3j_i}\right) < (\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \gamma_{3j_i}$.

We can repeat this step inductively for all hyperplanes comprising $\mathcal{Q}_{j_i}$ to show that

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla\left((\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \ldots \wedge \gamma_{mj_i}\right) <$$
$$\left((\gamma_{1j_i} \wedge \gamma_{2j_i}) \wedge \ldots \wedge \gamma_{mj_i}\right)$$

The last step is to apply the negation induced by the R-function and arrive at the desired result:

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla \gamma_{j_i}(\mathbf{x}) > \gamma_{j_i}(\mathbf{x}) \geq 0 \tag{C.68}$$

The proof of (C.66) follows similar patterns. Here, we focus on δ_{j_i}. The external polygonal collar $\overline{\mathcal{Q}}_{j_i}$ can be assumed to have n sides, which means that we can write $\delta_{j_i} = ((\delta_{1j_i} \wedge \delta_{2j_i}) \wedge \ldots \wedge \delta_{nj_i})$. Following the procedure outlined above for the proof of (C.65), we can expand each term in the conjunction individually and then combine them to get

$$(\mathbf{x} - \mathbf{x}^*_{j_i})^\top \nabla \delta_{j_i}(\mathbf{x}) < \delta_{j_i}(\mathbf{x}) \tag{C.69}$$

We also have

$$\begin{aligned} \nabla \alpha_{j_i}(\mathbf{x}) &= \nabla \left(\frac{\delta_{j_i}(\mathbf{x})}{||\mathbf{x} - \mathbf{x}^*_{j_i}||} \right) \\ &= \frac{||\mathbf{x} - \mathbf{x}^*_{j_i}|| \nabla \delta_{j_i}(\mathbf{x}) - \delta_{j_i}(\mathbf{x}) \frac{\mathbf{x} - \mathbf{x}^*_{j_i}}{||\mathbf{x} - \mathbf{x}^*_{j_i}||}}{||\mathbf{x} - \mathbf{x}^*_{j_i}||^2} \end{aligned} \tag{C.70}$$

which gives the desired result using (C.69)

$$(\mathbf{x} - \mathbf{x}^*_{j_i}) \nabla \alpha_{j_i}(\mathbf{x}) = \frac{(\mathbf{x} - \mathbf{x}^*_{j_i}) \nabla \delta_{j_i}(\mathbf{x}) - \delta_{j_i}(\mathbf{x})}{||\mathbf{x} - \mathbf{x}^*_{j_i}||} < 0 \tag{C.71}$$

This concludes the proof that $\mathbf{h}^{\mathcal{I}}_{j_i}$ satisfies Property 1.

Next, we focus on Property 2. Pick a point $\mathbf{x} \in \partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i}$. This point could lie:

1. on the outer boundary of $\mathcal{F}^{\mathcal{I}}_{map,j_i}$ and away from P_i
2. on the boundary of one of the $|\mathcal{J}_{\mathcal{C}}|$ unknown but visible convex obstacles
3. on the boundary of one of the $(|\mathcal{J}^{\mathcal{I}}_{\mathcal{D}}| + |\mathcal{J}^{\mathcal{I}}_{\mathcal{B}}| - 1)$ familiar obstacles that are not P_i
4. on the boundary of P_i but not on the boundary of the polygon j_i
5. on the boundary of the polygon j_i

In the first four cases, we have $\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) = \mathbf{x}$, whereas in the last case, we have

$$\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) = \mathbf{x}^*_{j_i} + \frac{\left(\mathbf{x}_{1j_i} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}}{\left(\mathbf{x} - \mathbf{x}^*_{j_i}\right)^\top \mathbf{n}_{j_i}}(\mathbf{x} - \mathbf{x}^*_{j_i}) \tag{C.72}$$

It can be verified that $\left(\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) - \mathbf{x}_{1j_i}\right)^\top \mathbf{n}_{j_i} = 0$, which means that $\mathbf{x}$ is sent to the shared hyperplane between j_i and $p(j_i)$ as desired. This shows that we always have $\mathbf{h}^{\mathcal{I}}_{j_i}(\mathbf{x}) \in \partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$ and the map satisfies Property 2.

Finally, Property 3 derives from above and the fact that each boundary segment $\partial_k \mathcal{F}^{\mathcal{I}}_{map,j_i}$ is an one-dimensional manifold, the boundary of either a convex set or a polygon, both of which are homeomorphic to S^1 and, therefore, the corresponding boundary $\partial_k \mathcal{F}^{\mathcal{I}}_{map,p(j_i)}$. □

Proof of Theorem 8.1. We first focus on the proof of (the more specific) part 2 of Theorem 8.1 and follow similar patterns with the proof of Theorem 7.1. First of all, the vector field $\mathbf{u}^{\mathcal{I}}$ is Lipschitz continuous since $\mathbf{v}^{\mathcal{I}}(\mathbf{y})$ is shown to be Lipschitz continuous in [9] and $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ is a smooth change of coordinates away from sharp corners. Therefore, the vector field $\mathbf{u}^{\mathcal{I}}$ generates a unique continuously differentiable partial flow. To ensure completeness (i.e., absence of finite time escape through boundaries in $\mathcal{F}^{\mathcal{I}}_{map}$) we must verify that the robot never collides with any obstacle in the environment, i.e., leaves its freespace positively invariant. However, this property follows directly from the fact that the vector field $\mathbf{u}^{\mathcal{I}}$ on $\mathcal{F}^{\mathcal{I}}_{map}$ is the pushforward of the complete vector field $\mathbf{v}^{\mathcal{I}}$ through $(\mathbf{h}^{\mathcal{I}})^{-1}$, guaranteed to insure that $\mathcal{F}^{\mathcal{I}}_{model}$ remain positively invariant under its flow as shown in [9], away from sharp corners on the boundary of $\mathcal{F}^{\mathcal{I}}_{map}$. Therefore, with $\mathcal{I} = \mathcal{N}_{\mathcal{P}}$ the terminal mode of the hybrid controller, the freespace interior $\mathcal{F}^{\mathcal{I}}_{map}$ is positively invariant under (8.1).

Next, we focus on the critical points of (8.1). As shown in Lemma 7.6, with $\mathcal{I} = \mathcal{N}_{\mathcal{P}}$ the terminal mode of the hybrid controller:

1. The set of stationary points of control law (8.1) is given as

$$\{\mathbf{x}_d\} \bigcup \{(\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{s}_i)\}_{i \in \mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \bigcup \{\mathcal{G}_k\}_{k \in \mathcal{J}_{\mathcal{C}}}$$

where

$$\mathbf{s}_i = \mathbf{x}_i^* - \rho_i \frac{\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d) - \mathbf{x}_i^*}{||\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d) - \mathbf{x}_i^*||} \tag{C.73a}$$

$$\mathcal{G}_k = \left\{ \mathbf{q} \in \mathcal{F}_{map}^{\mathcal{I}} \middle| d(\mathbf{q}, C_k) = r, \kappa(\mathbf{q}) = 1 \right\} \tag{C.73b}$$

with

$$\kappa(\mathbf{q}) := \frac{(\mathbf{q} - \Pi_{\overline{C}_k}(\mathbf{q}))^\top (\mathbf{q} - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d))}{||\mathbf{q} - \Pi_{\overline{C}_k}(\mathbf{q})|| \cdot ||\mathbf{q} - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||}$$

2. The goal $\mathbf{x}_d$ is the only locally stable equilibrium of control law (8.1) and all the other stationary points $\{(\mathbf{h}^{\mathcal{I}})^{-1}(\mathbf{s}_i)\}_{i \in \mathcal{J}_{\mathcal{D}}^{\mathcal{I}}} \bigcup \{\mathcal{G}_k\}_{k \in \mathcal{J}_{\mathcal{C}}}$, each associated with an obstacle, are nondegenerate saddles.

Consider the smooth Lyapunov function candidate $V^{\mathcal{I}}(\mathbf{x}) = ||\mathbf{h}^{\mathcal{I}}(\mathbf{x}) - \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)||^2$. Using (8.1) and writing $\mathbf{y} = \mathbf{h}^{\mathcal{I}}(\mathbf{x})$ and $\mathbf{y}_d = \mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)$, we get

$$\begin{aligned}
\frac{dV^{\mathcal{I}}}{dt} =& 2(\mathbf{y} - \mathbf{y}_d)^\top \mathbf{D}_{\mathbf{x}} \mathbf{h}^{\mathcal{I}} \dot{\mathbf{x}} \\
=& -2k(\mathbf{y} - \mathbf{y}_d)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
=& -2k \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d) + \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d) - \mathbf{y}_d\right)^\top \\
& \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
=& -2k||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 \\
& + 2k \left(\mathbf{y}_d - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \\
\leq& -2k||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 \leq 0
\end{aligned} \tag{C.74}$$

since $\mathbf{y} \in \mathcal{LF}(\mathbf{y})$, which implies that

$$\left(\mathbf{y}_d - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right)^\top \left(\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) \leq 0 \tag{C.75}$$

since either $\mathbf{y}_d = \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)$, or $\mathbf{y}_d$ and $\mathbf{y}$ are separated by a hyperplane passing through $\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)$. Therefore, similarly to [9], using LaSalle's invariance principle we see that every trajectory starting in $\mathcal{F}^{\mathcal{I}}_{map}$ approaches the largest invariant set in $\{\mathbf{x} \in \mathcal{F}^{\mathcal{I}}_{map} \,|\, \dot{V}^{\mathcal{I}}(\mathbf{x}) = 0\}$, i.e. the equilibrium points of (8.1). The desired result follows directly from the fact that $\mathbf{x}_d$ is the only locally stable equilibrium of our control law and the rest of the stationary points are nondegenerate saddles, whose regions of attraction have empty interior in $\mathcal{F}^{\mathcal{I}}_{map}$, as discussed above.

Next, we focus on the more general part 1 of Theorem 8.1. Since the target now moves, we compute the time derivative of $V^{\mathcal{I}}$, using (C.75), as

$$\begin{aligned}\frac{dV^{\mathcal{I}}}{dt} =& 2(\mathbf{y}-\mathbf{y}_d)^\top \left[\mathbf{D}_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}(\mathbf{x})\cdot\dot{\mathbf{x}} - \mathbf{D}_{\mathbf{x}}\mathbf{h}^{\mathcal{I}}(\mathbf{x}_d)\cdot\dot{\mathbf{x}}_d\right] \\ =& -2k(\mathbf{y}-\mathbf{y}_d)^\top\left(\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)\right) - 2(\mathbf{y}-\mathbf{y}_d)^\top\dot{\mathbf{y}}_d \\ \leq& -2k||\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 - 2(\mathbf{y}-\mathbf{y}_d)^\top\dot{\mathbf{y}}_d\end{aligned}$$

If $(\mathbf{y}-\mathbf{y}_d)^\top\dot{\mathbf{y}}_d > 0$, then the desired result $\dfrac{dV^{\mathcal{I}}}{dt} \leq 0$ is immediately derived. On the other hand, if $||\dot{\mathbf{y}}_d|| \leq k\,\dfrac{||\mathbf{y}-\Pi_{\mathsf{B}\left(\mathbf{y},0.5d(\mathbf{y},\partial\mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{y}_d)||^2}{||\mathbf{y}-\mathbf{y}_d||}$, then we use the Cauchy-Schwarz inequality $-2(\mathbf{y}-\mathbf{y}_d)^\top\dot{\mathbf{y}}_d \leq 2||\mathbf{y}-\mathbf{y}_d||\,||\dot{\mathbf{y}}_d||$ to write

$$\begin{aligned}\frac{dV^{\mathcal{I}}}{dt} \leq& -2k||\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 + 2||\mathbf{y}-\mathbf{y}_d||\,||\dot{\mathbf{y}}_d|| \\ \leq& -2k||\mathbf{y}-\Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||^2 \\ &+ k\,||\mathbf{y}-\Pi_{\mathsf{B}\left(\mathbf{y},0.5d(\mathbf{y},\partial\mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{y}_d)||^2 \end{aligned} \tag{C.76}$$

Note here that by construction of the convex local freespace in the model space $\mathcal{LF}(\mathbf{y})$ as in [9, Eqn. (25)], which guarantees that the distance of $\mathbf{y}$ to the boundary of $\mathcal{LF}(\mathbf{y})$ is $\frac{d(\mathbf{y},\partial\mathcal{F}^{\mathcal{I}}_{model})}{2}$, we get that $\mathsf{B}\left(\mathbf{y}, 0.5\,d(\mathbf{y},\partial\mathcal{F}^{\mathcal{I}}_{model})\right) \subset \mathcal{LF}(\mathbf{y})$.

We need to distinguish between two cases:

1. If $\mathbf{y}_d \in \mathsf{B}\left(\mathbf{y}, 0.5\, d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right)$, then:

$$\Pi_{\mathsf{B}\left(\mathbf{y}, 0.5 d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{y}_d) = \mathbf{y}_d$$

and $||\mathbf{y} - \Pi_{\mathsf{B}\left(\mathbf{y}, 0.5 d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{y}_d)|| = ||\mathbf{y} - \mathbf{y}_d||$. Moreover, since $\mathsf{B}\left(\mathbf{y}, 0.5\, d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right) \subset \mathcal{LF}(\mathbf{y})$, $||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)|| = ||\mathbf{y} - \mathbf{y}_d||$. From (C.76), we now immediately get that

$$\frac{dV^{\mathcal{I}}}{dt} \leq -k||\mathbf{y} - \mathbf{y}_d||^2 \leq 0$$

2. If $\mathbf{y}_d \notin \mathsf{B}\left(\mathbf{y}, 0.5\, d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right)$, then $||\mathbf{y} - \Pi_{\mathsf{B}\left(\mathbf{y}, 0.5 d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right)}(\mathbf{y}_d)|| = \frac{d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})}{2} \leq ||\mathbf{y} - \Pi_{\mathcal{LF}(\mathbf{y})}(\mathbf{y}_d)||$, since $\mathsf{B}\left(\mathbf{y}, 0.5\, d(\mathbf{y}, \partial\mathcal{F}^{\mathcal{I}}_{model})\right) \subset \mathcal{LF}(\mathbf{y})$. The desired result $\frac{dV^{\mathcal{I}}}{dt} \leq 0$ is now derived from (C.76) by simple substitution.

□

C.7 Proofs of Results in Chapter 9

Proof of Proposition 9.1. To show this result it suffices to show that eventually the accepting condition of the NBA is satisfied, i.e., the robot will visit at least one of the final NBA states q_F infinitely often. Equivalently, as discussed in Section 9.2.2, it suffices to show that accepting edges $(q_B, q'_B) \in \mathcal{E}$, where $q_B, q'_B \in \mathcal{V}$ are traversed infinitely often.

First, consider an infinite sequence of time instants $\mathbf{t} = t_0, t_1, \ldots, t_k, \ldots$ where $t_{k+1} \geq t_k$, so that an edge in $\mathcal{G}$, defined in Section 9.2.2, is traversed at every time instant t_k. Let $e(t_k) \in \mathcal{E}$ denote the edge that is traversed at time t_k. Thus, $\mathbf{t}$ yields the following sequence of edges $\mathbf{e} = e(t_0), e(t_1), \ldots, e(t_k) \ldots$ where $e(t_k) = (q_B(t_k), q_B(t_{k+1}))$, $q_B(t_0) = q_B^{\text{aux}}$, $q_B(t_k) \in \mathcal{V}$, and the state q_B^{k+1} is defined based on the following two cases. If $q_B(t_k) \notin \mathcal{V}_F$, then the state $q_B(t_{k+1})$ is closer to $\mathcal{V}_F$ than $q_B(t_k)$ is, i.e., $d_F(q_B(t_{k+1}), \mathcal{V}_F) = d_F(q_B(t_k), \mathcal{V}_F) - 1$, where d_F is defined in (9.7). If $q_B(t_k) \in \mathcal{V}_F$, then $q_B(t_{k+1})$ is selected so that an accepting edge originating from $q_B(t_k)$ is traversed. By definition of $q_B(t_k)$, the 'distance' to $\mathcal{V}_F$ decreases as t_k increases, i.e., given any time instant t_k, there exists a time instant $t'_k \geq t_k$

so that $q_B(t'_k) \in \mathcal{V}_F$ and then at the next time instant an accepting edge is traversed. This means that $\mathbf{e}$ includes an infinite number of accepting edges. This sequence $\mathbf{e}$ exists since, by assumption, there exists an infinite sequence of manipulation actions that satisfies ϕ. Particularly, recall that by construction of the graph $\mathcal{G}$, the set of edges in this graph captures all NBA transitions besides those that (i) require the robot to be in more than one region simultaneously or (ii) multi-hop NBA transitions that require the robot to jump instantaneously from one region of interest which are not meaningful in practice. As a result, if there does not exist at least one sequence $\mathbf{e}$, i.e., at least one infinite path in $\mathcal{G}$ that starts from the initial state and traverses at least one accepting edge infinitely often, then this means that there is no path that satisfies ϕ (unless conditions (i)-(ii) mentioned before are violated).

Assume that the discrete controller selects NBA states as discussed in Section 9.2.3. To show that the discrete controller is complete, it suffices to show that it can generate a infinite sequence of edges $\mathbf{e}$ as defined before. Note that the discrete controller selects next NBA states that the robot should reach in the same way as discussed before. Also, by assumption, the environmental structure and the continuous-time controller ensure that at least one of the candidate next NBA states (i.e., the ones that can decrease the distance to $\mathcal{V}_F$) can be reached. Based on these two observations, we conclude that such a sequence $\mathbf{e}$ will be generated, completing the proof. □

Appendix D

Accompanying Software

This Appendix briefly describes developed software packages, implementing algorithms presented in this thesis.

D.1 Software Package `doubly_reactive_matlab`

This package is a MATLAB-ROS implementation of the doubly-reactive, sensor-based homing algorithm for Minitaur, using a LIDAR and range-only target localization, as presented in Chapter 3. The software package can be found here: `https://github.com/KodlabPenn/doubly_reactive_matlab`.

The doubly-reactive operations and the functions included in the package are based on [6] and [7]. The ROS wrapper for the PulsON P440 and P410 ultra-wideband radios from Time Domain, used to extract range measurements from the robot to the goal, and publishing the `/minitaur/ranges/ranges` ROS topic can be found here: `https://github.com/vvasilo/pulson_ros`.

D.1.1 Preliminaries

The main script is `ros_doubly_reactive.m`, while `startupROS.m` needs to be run first for initialization.

The script assumes an active ROS master on the robot and published topics streaming IMU data (`/minitaur/imu`), proprioceptive speed estimates (`/minitaur/speed`), distance

to target (`/minitaur/ranges/ranges`) and LIDAR data (`/minitaur/scan`). A joystick is assumed to be connected to the desktop computer and used to stop the behavior.

The node publishes the desired robot behavior to `/minitaur/set_cmd`, and the desired linear and angular speed of the robot to `/minitaur/set_twist`.

D.1.2 Tuning and Use

The commands, joystick buttons and collision avoidance, control, particle filter and twist filtering parameters are tuned in lines 29-76 of `ros_doubly_reactive.m`.

D.2 Software Package `semnav`

This package can be used for doubly reactive navigation with semantic feedback, using C++ and ROS, as presented in Chapters 7 - 8. The software package can be found here: `https://github.com/KodlabPenn/semnav`.

The doubly-reactive operations in the model space are based on [6] and [7]. It has been tested with Ubuntu 18.04 and ROS Melodic, on three different robots: Turtlebot, Ghost Robotics Minitaur and Ghost Robotics Spirit.

D.2.1 Hardware Setup

The package assumes that the robot possesses:

1. a LIDAR sensor, for estimating distance to unknown obstacles.
2. a way of generating a semantic map of its surroundings with familiar obstacles (see details in Semantic SLAM interfaces below).
3. a way of generating its own odometry estimate.

These three inputs are given as topics in the `navigation_*` launch files (see below).

D.2.2 Prerequisites

Note the following:

- For our experiments, we use the ZED Mini stereo camera. A resolution of

720HD@60Hz works well with an NVIDIA TX2 or an NVIDIA Xavier (make sure to enable maximum performance first by running: `sudo nvpmodel -m 0`).

- For reading a Hokuyo LIDAR sensor, we use the `urg_node` package, found here: `http://wiki.ros.org/urg_node`.
- For LIDAR downsampling, we use the (forked and modified) `laser_scan_sparsifier` package (found here: `https://github.com/vvasilo/scan_tools/tree/indigo/laser_scan_sparsifier`), included in `scan_tools` (found here: `https://github.com/vvasilo/scan_tools`). This package depends on `csm` (found here: `https://github.com/AndreaCensi/csm`) which must be installed first.
- We use the `robot_localization` package (found here: `http://wiki.ros.org/robot_localization`) for fusing odometry inputs from multiple sources.
- We use Boost Geometry (`https://www.boost.org/doc/libs/1_70_0/libs/geometry/doc/html/index.html`) for basic operations with planar polygons, which must be already installed in the user's system.
- For more advanced computational geometry operations, we use the CGAL library. See here for installation instructions: `https://www.cgal.org/download.html`.
- We implement the ear clipping triangulation method in C++ using the `earcut.hpp` package, included here: `https://github.com/KodlabPenn/semnav/blob/master/include`. For the Python implementation, we use the `tripy` package, included here: `https://github.com/linuxlewis/tripy`.
- Except for the ROS Python packages (already included with ROS), the following Python packages are also needed: `shapely`, `scipy` and `numpy`.
- For properly using the visualization functionalities in `visualization.py`, we need the Python modules `matplotlib` and `imagemagick`.

- For benchmark experiments with Vicon, the `motion_capture_system` package is used, found here: `https://github.com/KumarRobotics/motion_capture_system`.

The user can install all the prerequisites, by first independently installing the ZED SDK (`https://www.stereolabs.com/developers/`), and then running the following commands:

```
sudo apt-get install ros-melodic-urg-node ros-melodic-robot-localization
sudo apt-get install python-shapely python-scipy python-numpy libcgal-dev
cd ~/catkin_ws/src
git clone https://github.com/stereolabs/zed-ros-wrapper.git
git clone https://github.com/AndreaCensi/csm.git
git clone https://github.com/vvasilo/scan_tools.git
git clone https://github.com/KumarRobotics/motion_capture_system.git
catkin build csm
catkin build
pip install tripy
```

D.2.3 Installation

Once all the prerequisites above are satisfied, the user can install the package with:

```
cd ~/catkin_ws/src
git clone https://github.com/vvasilo/semnav.git
cp -r semnav/extras/object_pose_interface_msgs .
catkin build
```

D.2.4 Semantic SLAM Interfaces

This package needs an external Semantic SLAM engine, which is not included by default. However, any such engine can be used. The only restriction is associated with the type of messages used, i.e., the semantic map has to be given in a specific way. In our implementation, these messages are included in a separate package called `object_pose_interface_msgs`. We include pointers to the necessary message formats in the `extras` folder. We provide the semantic map in the form of a `SemanticMapObjectArray`

message. Each `SemanticMapObject` in the array has a `classification` and `pose` element, as well as a number of 3D `keypoints`.

Using `semslam_polygon_publisher.py`, we project those keypoints on the horizontal plane of motion, and republish the semantic map object with a CCW-oriented `polygon2d` element (i.e., the projection of this 3D object on the 2D plane). To do so, we use a pre-defined object mesh, given in the form of a .mat file. The `mesh_location` for all objects is defined in the associated `tracking_*` launch file (see below), and we include examples for different objects here: `https://github.com/KodlabPenn/semnav/blob/master/extras/meshes`.

Note that if the user knows the 2D polygon directly, the above procedure is not necessary - only the `polygon2d` element of each `SemanticMapObject` is used for navigation.

D.2.5 Types of Files and Libraries

To use the code on a real robot, the user needs to launch one of each type of launch files below:

- The files with name `bringup_*` launch the sensors for each corresponding robot. For example, the file `bringup_turtlebot.launch` launches:
  1. the stereo camera launch file (`zed_no_tf.launch`).
  2. the Vicon launch file (if present).
  3. the Kobuki node to bring up Turtlebot's control.
  4. the `urg_node` node for the Hokuyo LIDAR sensor.
  5. the `laser_scan_sparsifier` node for downsampling the LIDAR data.
- The files with name `tracking_*` launch the tracking files needed for semantic navigation. For example, the file `tracking_turtlebot_semslam_onboard.launch` launches:
  1. the corresponding semantic SLAM launch file from the semantic SLAM package.
  2. the necessary tf transforms (e.g., between the camera and the robot and between the LIDAR and the robot) for this particular robot.

3. the `semslam_polygon_publisher.py` node that subscribes to the output of the semantic SLAM and publishes 2D polygons on the plane.

- The files with name `navigation_*` launch the reactive controller. For example, the file `navigation_turtlebot_onboard.launch` launches the main navigation node for Turtlebot, which subscribes to:

  1. the local odometry node (in this case provided directly by the ZED stereo camera).
  2. the LIDAR data, after downsampling.
  3. the 2D polygons from `semslam_polygon_publisher.py`.
  4. necessary tf updates to correct local odometry as new updates from the semantic SLAM pipeline become available.

We also include a debugging launch file, that communicates with fake LIDAR, odometry and semantic map publishers (see here: `https://github.com/KodlabPenn/semnav/blob/master/launch/navigation_debug.launch`).

D.3 Software Package `semnav_matlab`

This package communicates with the Python scripts of the `semnav` package, to simulate doubly reactive navigation with semantic feedback in MATLAB, as presented in Chapters 7 - 9. The software package can be found here: `https://github.com/KodlabPenn/semnav_matlab`.

The doubly-reactive operations in the model space are based on [6] and [7]. Except for diffeomorphism-based navigation, the simulation also includes support for RRT-X [143], adapted from an implementation that can be found here: `https://github.com/rahul-sb/RRTx`.

D.3.1 Prerequisites

Note the following:

- The user needs to make sure that `semnav` is downloaded (not necessarily installed).
- The user needs to open the `startup` script (found here: `https://github.com/KodlabPenn/semnav_matlab/blob/master/startup.m`) and:
    1. modify the Python path (`path_python_ubuntu` or `path_python_mac`) depending on whether the operating system is Ubuntu or Mac.
    2. modify the path to `semnav` (`path_semnav_ubuntu` or `path_semnav_mac`) depending on whether the operating system is Ubuntu or Mac.
    3. (Mac users might also need to specify the `path_packages` variable.)
- The user needs to run `startup.m` to load Python and `semnav`.
- If it doesn't already exist, the user needs to make a folder called `multimedia`.

D.3.2 Running the Simulation

In order to run the simulation, the user needs to make a scenario. Many examples of scenarios are included in the corresponding `scenario.m` file, found here: `https://github.com/KodlabPenn/semnav_matlab/blob/master/demo/scenario.m`. We suggest copying one of them and modifying it appropriately. Scenario parameters and their meaning are described near the top of the file.

The user also needs to add/modify the plot options corresponding to the scenario number in `option.m`. The default settings should work well. Following that:

1. In order to run the diffeomorphism-based doubly reactive navigation scheme, the user needs to call `demoDiffeo` with the number of the corresponding scenario. We also include `vectorField.m` if the user needs to see the generated vector field, assuming no prior memory for the robot.
2. In order to run RRT-X, the user needs to call `demoRRT` with the number of the corresponding scenario.

Both files include several parameters (`flagSaveVideo`, `flagSaveGif`, `flagSaveFigure`) that can be set to 1 or 0 to toggle output. All generated multimedia files are saved in the multimedia folder. We also include a `jobs.m` file for multiple simulation jobs.

Bibliography

[1] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit. Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. *Autonomous Robots*, 42(4):801–824, 2018.

[2] Amazon, Inc. Amazon prime air, 2014. URL `http://www.amazon.com/b?node=8037720011`.

[3] ANYbotics. ANYmal – Let Robots Go Anywhere, 2018. URL `https://www.youtube.com/watch?v=m1-s8iOJaI4`.

[4] ANYbotics. World's First Autonomous Offshore Robot, 2018. URL `https://youtu.be/DzTvIPrt0DY`.

[5] ANYbotics. Last-Meter Robotic Package Delivery with ANYmal (CES 2019, ANYbotics & Continental), 2019. URL `https://www.youtube.com/watch?v=v3g5xp5Kr2g`.

[6] O. Arslan and D. E. Koditschek. Exact Robot Navigation Using Power Diagrams. In *IEEE International Conference on Robotics and Automation*, pages 1–8, 2016. doi: 10.1109/ICRA.2016.7487090.

[7] O. Arslan and D. E. Koditschek. Sensor-Based Reactive Navigation in Unknown Convex Sphere Worlds. In *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[8] O. Arslan and D. E. Koditschek. Smooth Extensions of Feedback Motion Planners via Reference Governors. In *IEEE International Conference on Robotics and Automation*, pages 4414–4421, 2017.

[9] O. Arslan and D. E. Koditschek. Sensor-Based Reactive Navigation in Unknown Convex Sphere Worlds. *International Journal of Robotics Research*, 38(1-2):196–223, Jul 2018.

[10] O. Arslan, D. P. Guralnik, and D. E. Koditschek. Coordinated robot navigation via hierarchical clustering. *IEEE Transactions on Robotics*, 32(2):352–371, 2016.

[11] O. Arslan, V. Pacelli, and D. E. Koditschek. Sensory steering for sampling-based motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3708–3715, 2017.

[12] A. Astolfi. Exponential Stabilization of a Wheeled Mobile Robot Via Discontinuous Control. *Journal of Dynamic Systems, Measurement, and Control*, 121(1):121–126, 1999.

[13] A. Ataka, H. Lam, and K. Althoefer. Reactive Magnetic-Field-Inspired Navigation for Non-Holonomic Mobile Robots in Unknown Environments. In *IEEE International Conference on Robotics and Automation*, pages 6983–6988, 2018.

[14] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas. Localization from semantic observations via the matrix permanent. *The International Journal of Robotics Research*, 35(1-3):73–99, 2016.

[15] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu. What Happened at the DARPA Robotics Challenge, and Why?, 2015. URL `https://www.cs.cmu.edu/~cga/drc/jfr-what.pdf`.

[16] F. Aurenhammer. Power Diagrams: Properties, Algorithms and Applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.

[17] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[18] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin. An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments. *arXiv: 1905.00532*, 2019.

[19] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining Optimal Control and Learning for Visual Navigation in Novel Environments. *arXiv preprint*, 2019. URL `https://arxiv.org/abs/1903.02531`.

[20] Y. Baryshnikov and B. Shapiro. How to run a centipede: a topological perspective. *Geometric Control Theory and Sub-Riemannian Geometry*, pages 37–51, 2014.

[21] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005.

[22] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning on constraint manifolds. In *IEEE International Conference on Robotics and Automation*, 2009.

[23] S. Bhattacharya, R. Ghrist, and V. Kumar. Persistent Homology for Path Planning in Uncertain Environments. *IEEE Transactions on Robotics*, 31(3):578–590, 2015.

[24] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli. Efficient Collision Checking in Sampling-based Motion Planning. In *The 10th International Workshop on the Algorithmic Foundations of Robotics*, 2012.

[25] A. Bisoffi and D. V. Dimarogonas. A hybrid barrier certificate approach to satisfy

linear temporal logic specifications. In *American Control Conference*, pages 634–639, 2018.

[26] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2245 – 2252, 2018.

[27] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[28] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[29] Boston Dynamics. SpotMini Autonomous Navigation, 2018. URL `https://www.youtube.com/watch?v=Ve9kWX_KXus/`.

[30] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic SLAM. In *IEEE International Conference on Robotics and Automation*, pages 1722–1729, May 2017.

[31] H. I. Bozma and D. E. Koditschek. Assembly as a noncooperative game of its pieces: analysis of 1D sphere assemblies. *Robotica*, 19(01):93–108, 2001.

[32] M. S. Branicky. Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[33] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, pages 341–346, 1999.

[34] R. W. Brockett. Asymptotic stability and feedback stabilization. In *Differential Geometric Control Theory*, pages 181–191. Birkhauser, 1983.

[35] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential Composition of Dynamically Dexterous Robot Behaviors. *The International Journal of Robotics Research*, 18:534–555, 1999.

[36] R. W. Chaney. Piecewise C^k functions in nonsmooth analysis. *Nonlinear Analysis: Theory, Methods & Applications*, 15(7):649–660, 1990. ISSN 0362-546X. doi: 10.1016/0362-546X(90)90005-2.

[37] D. Chang and J. Marsden. Gyroscopic Forces and Collision Avoidance with Convex Obstacles. *New Trends in Nonlinear Dynamics and Control and their Applications*, pages 145–159, 2003.

[38] R. Chatila. Deliberation and reactivity in autonomous mobile robots. *Robotics and Autonomous Systems*, 16(2):197–211, Dec 1995. ISSN 0921-8890. doi: 10.1016/0921-8890(96)81009-8. URL `http://www.sciencedirect.com/science/article/pii/0921889096810098`.

[39] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[40] R. Cisneros, K. Yokoi, and E. Yoshida. Impulsive Pedipulation of a Spherical Object with 3D Goal Position by a Humanoid Robot: A 3D Targeted Kicking Motion Generator. *International Journal of Humanoid Robotics*, 13(2), 2016.

[41] E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. *Advances in Spatial Databases*, pages 277–295, 1993.

[42] D. Conner, H. Choset, and A. A. Rizzi. Integrating planning and control for single-bodied wheeled mobile robots. *Autonomous Robots*, 30(3):243–264, 2011.

[43] P. I. Corke. *Robotics, Vision & Control*. Springer, 2017. ISBN 978-3-319-54413-7.

[44] R. Correa, D. Salas, and L. Thibault. Smoothness of the metric projection onto non-convex bodies in Hilbert spaces. *Journal of Mathematical Analysis and Applications*, 457(2):1307–1332, 2018.

[45] DARPA. DARPA Robotics Challenge (DRC) (Archived), 2020. URL `https://www.darpa.mil/program/darpa-robotics-challenge`.

[46] A. De and D. E. Koditschek. Toward dynamical sensor management for reactive wall-following. In *IEEE International Conference on Robotics and Automation*, pages 2400—-2406, 2013.

[47] A. De and D. E. Koditschek. Parallel composition of templates for tail-energized planar hopping. In *IEEE International Conference on Robotics and Automation*, pages 4562–4569, 2015.

[48] A. De and D. E. Koditschek. Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting. *The International Journal of Robotics Research*, 37(7):743–778, 2018.

[49] A. De and D. E. Koditschek. Event-driven Reactive Dynamical Quadrupedal Walking. *In Prep*, 2020.

[50] F. Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. *Technical Report number GT-RIM-CP&R-2012-002*, 2012. URL `https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pd`.

[51] A. Deshpande, L. P. Kaelbling, and T. Lozano-Perez. Decidability of semi-holonomic prehensile task and motion planning. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016.

[52] X. Ding and F. Yang. Study on hexapod robot manipulation using legs. *Robotica*, 34 (2):468–481, 2016.

[53] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 1973.

[54] M. J. Egenhofer and J. R. Herring. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. *International Series of Monographs on Computer Science*, 1991.

[55] H. ElGindy, H. Everett, and G. T. Toussaint. Slicing an ear using prune-and-search. *Pattern Recognition Letters*, 14(9):719–722, 1993.

[56] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Hybrid controllers for path planning: A temporal logic approach. In *IEEE Conference on Decision and Control, European Control Conference*, pages 4885–4890, 2005.

[57] M. Farber. Topology of Robot Motion Planning. *Morse Theoretic Methods in Nonlinear Analysis and in Symplectic Topology*, pages 185–230, 2006.

[58] M. Farber, M. Grant, G. Lupton, and J. Oprea. An upper bound for topological complexity. *Topology and its Applications*, 255:109–125, 2019.

[59] I. F. Filippidis and K. J. Kyriakopoulos. Navigation Functions for Everywhere Partially Sufficiently Curved Worlds. In *IEEE International Conference on Robotics and Automation*, pages 2115–2120, 2012.

[60] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[61] R. J. Full and D. E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 1999.

[62] F. Gao, W. Wu, W. Gao, and S. Shen. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics*, 36(4):710–733, 2019.

[63] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 37(13-14):1796–1825, 2018.

[64] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *International Conference on Computer Aided Verification*, pages 53–65, 2001.

[65] Ghost Robotics. Ghost Robotics Minitaur, 2016. URL `http://www.ghostrobotics.io/minitaur/`.

[66] Ghost Robotics. Ghost Robotics: Q-UGV Capabilities Summary, 2018. URL `https://www.youtube.com/watch?v=kfQIFw5kbOY`.

[67] Ghost Robotics. Ghost Robotics Spirit, 2020. URL `http://www.ghostrobotics.io`.

[68] D. H. Greene. The decomposition of polygons into convex parts. *Computational Geometry*, 1:235–259, 1983.

[69] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[70] M. Guo, K. H. Johansson, and D. V. Dimarogonas. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *IEEE International Conference on Robotics and Automation*, pages 5025–5032, 2013.

[71] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7272–7281, 2017.

[72] H.-N. Wu, M. Levihn, and M. Stilman. Navigation Among Movable Obstacles in unknown environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1433–1438, 2010.

[73] R. I. Hartley. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[74] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi. Towards manipulation planning with temporal logic specifications. In *IEEE International Conference on Robotics and Automation*, pages 346–352, 2015.

[75] P. Hebert, M. Bajracharya, J. Ma, N. Hudson, A. Aydemir, J. Reid, C. Bergh, J. Borders, M. Frost, M. Hagman, J. Leichty, P. Backes, B. Kennedy, P. Karplus, B. Satzinger, K. Byl, K. Shankar, and J. Burdick. Mobile Manipulation and Mobility as Manipulation—Design and Algorithms of RoboSimian. *Journal of Field Robotics*, 32(2):255–274, 2015.

[76] P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to navigate through crowded environments. In *IEEE International Conference on Robotics and Automation*, pages 981–986, 2010.

[77] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Consistency Analysis and Improvement of Vision-aided Inertial Navigation. *IEEE Transactions on Robotics*, 30(1):158–176, 2014.

[78] M. W. Hirsch. *Differential Topology*. Springer, 1976.

[79] Hokuyo. UTM-30LX, 2020. URL `https://www.hokuyo-aut.jp/search/single.php?serial=169`.

[80] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the "Warehouseman's Problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[81] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger. ANYmal - a highly mobile and dynamic quadrupedal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 38–44, 2016.

[82] IEEE Spectrum. A compilation of robots falling down at the DARPA Robotics Challenge, 2015. URL `https://www.youtube.com/watch?v=g0TaYhjpOfo`.

[83] B. D. Ilhan, A. M. Johnson, and D. E. Koditschek. Autonomous legged hill ascent. *Journal of Field Robotics*, 35(5):802–832, Aug 2018. ISSN 1556-4967. doi: 10.1002/rob.21779.

[84] L. Janson, T. Hu, and M. Pavone. Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies. In *Robotics: Science and Systems*, 2018.

[85] R. P. Jayakumar, M. S. Madhav, F. Savelli, H. T. Blair, N. J. Cowan, and J. J. Knierim. Recalibration of path integration in hippocampal place cells. *Nature*, 566 (7745):533–537, 2019.

[86] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek. Autonomous legged hill and stairwell ascent. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 134–142, 2011.

[87] A. M. Johnson, S. A. Burden, and D. E. Koditschek. A hybrid systems model for simple manipulation and self-manipulation systems. *The International Journal of Robotics Research*, 35(11):1354–1392, 2016.

[88] E. W. Justh and P. S. Krishnaprasad. Natural frames and interacting particles in three dimensions. In *IEEE Conference on Decision and Control*, pages 2841–2846, 2005.

[89] L. P. Kaelbling and T. Lozano-Perez. Hierarchical task and motion planning in the

now. In *IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011.

[90] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[91] Y. Kantaros and M. M. Zavlanos. STyLuS*: A Temporal Logic Optimal Control Synthesis Algorithm for Large-Scale Multi-Robot Systems. *The International Journal of Robotics Research*, 39(7):812–836, 2020.

[92] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas. Reactive temporal logic planning for multiple robots in unknown environments. In *IEEE International Conference on Robotics and Automation*, page (to appear), 2020.

[93] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1966–1974, 2015.

[94] C. S. Karagoz, H. I. Bozma, and D. E. Koditschek. Feedback-based event-driven parts moving. *IEEE Transactions on Robotics*, 20(6):1012–1018, 2004.

[95] C. S. Karagoz, H. I. Bozma, and D. E. Koditschek. Coordinated Navigation of Multiple Independent Disk-Shaped Robots. *IEEE Transactions on Robotics*, 30(6):1289–1304, December 2014.

[96] S. Karaman and E. Frazzoli. High-speed flight in an ergodic forest. In *IEEE International Conference on Robotics and Automation*, pages 2899–2906, 2012.

[97] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[98] J. M. Keil. Decomposing a Polygon into Simpler Components. *SIAM Journal on Computing*, 14(4):799–817, 1985.

[99] M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. *International Journal of Computational Geometry & Applications*, 12(3): 181–192, 2002.

[100] G. Kenneally and D. E. Koditschek. Leg Design for Energy Management in an Electromechanical Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5712–5718, 2015.

[101] G. Kenneally, A. De, and D. E. Koditschek. Design Principles for a Family of Direct-Drive Legged Robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

[102] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[103] M. Kloetzer and C. Belta. A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[104] D. E. Koditschek and E. Rimon. Robot Navigation Functions on Manifolds with Boundary. *Advances in Applied Mathematics*, 11(4):412–442, 1990.

[105] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[106] C. Kong, C.-H. Lin, and S. Lucey. Using Locally Corresponding CAD Models for Dense 3D Reconstructions from a Single Image. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 5603–5611, 2017.

[107] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From Skills to Symbols: Learning

Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[108] G. D. Konidaris, L. Kaelbling, and T. Lozano-Perez. Constructing Symbolic Representations for High-Level Planning. In *proc. AAAI*, 2014.

[109] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan. Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots. *arXiv: 1809.06746*, 2018.

[110] N. Koyachi, H. Adachi, M. Izumi, and T. Hirose. Control of walk and manipulation by a hexapod with integrated limb mechanism: MELMANTIS-1. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3553–3558, 2002.

[111] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[112] A. Krontiris and K. Bekris. Dealing with difficult instances of object rearrangement. In *Robotics: Science and Systems*, 2015.

[113] A. Kumar, S. Gupta, D. Fouhey, S. Levine, and J. Malik. Visual Memory for Robust Path Following. In *Advances in Neural Information Processing Systems 31*, pages 765–774, 2018.

[114] KumarRobotics. Drivers for motion capture systems, 2020. URL `https://github.com/KumarRobotics/motion_capture_system`.

[115] L. Kuntz and S. Scholtes. Structural Analysis of Nonsmooth Mappings, Inverse Functions, and Metric Projections. *Journal of Mathematical Analysis and Applications*, 188(2):346–386, 1994. ISSN 0022-247X. doi: 10.1006/jmaa.1994.1431.

[116] D. Kurth, G. Kantor, and S. Singh. Experimental results in range-only localization with radio. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 974–979, 2003.

[117] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics*, 32(3):583–599, 2016.

[118] J. C. Latombe and D. Hsu. *Randomized single-query motion planning in expansive spaces*. PhD thesis, Stanford University, 2000.

[119] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[120] M. Levihn, J. Scholz, and M. Stilman. Planning with movable obstacles in continuous environments with uncertain dynamics. In *IEEE International Conference on Robotics and Automation*, pages 3832–3838, 2013.

[121] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 35(1-2):100–123, 2006.

[122] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. In *ACM Symposium on Solid and Physical Modeling*, pages 121–131, 2007.

[123] G. Lionis, X. Papageorgiou, and K. J. Kyriakopoulos. Towards locally computable polynomial navigation functions for convex obstacle workspaces. In *IEEE International Conference on Robotics and Automation*, pages 3725–3730, 2008.

[124] S. C. Livingston, R. M. Murray, and J. W. Burdick. Backtracking temporal logic synthesis for uncertain environments. In *IEEE International Conference on Robotics and Automation*, pages 5163–5170, 2012.

[125] S. C. Livingston, P. Prabhakar, A. B. Jose, and R. M. Murray. Patching task-level robot controllers based on a local μ-calculus formula. In *IEEE International Conference on Robotics and Automation*, pages 4588–4595, 2013.

[126] A. Majumdar and R. Tedrake. Funnel Libraries for Real-Time Robust Feedback Motion Planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[127] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *The 16th International Conference on Hybrid Systems: Computation and Control*, pages 353–362, 2013.

[128] B. Marthi, S. Russell, and J. Wolfe. Angelic Hierarchical Planning: Optimal and Online Algorithms. In *International Conference on Automated Planning and Scheduling*, 2008.

[129] M. T. Mason and K. M. Lynch. Dynamic manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 152–159, 1993.

[130] M. T. Mason, D. Pai, D. Rus, L. R. Taylor, and M. Erdmann. A Mobile Manipulator. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2322–2327, May 1999.

[131] W. S. Massey. Sufficient conditions for a local homeomorphism to be injective. *Topology and its Applications*, 47:133–148, 1992.

[132] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: The Planning Domain Definition Language. *Technical Report CVC TR98003/DCS TR1165*, 1998.

[133] G. H. Meisters. Polygons have ears. *American Mathematical Monthly*, 82:648–651, 1975.

[134] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Neural Autonomous Navigation with Riemannian Motion Policy. In *IEEE International Conference on Robotics and Automation*, pages 8860–8866, 2019.

[135] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar. Fast, autonomous flight in GPS-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120, 2018.

[136] C. A. Mueller and A. Birk. Conceptualization of Object Compositions Using Persistent Homology. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1095–1102, 2018.

[137] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[138] I. Noreen, A. Khan, and Z. Habib. Optimal path planning using RRT* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, 7(11):97–107, 2016.

[139] NVIDIA. Jetson TX2 Module, 2019. URL `https://developer.nvidia.com/embedded/jetson-tx2`.

[140] P. Ogren and N. E. Leonard. Obstacle avoidance in formation. In *IEEE International Conference on Robotics and Automation*, page 2492–2497, 2003.

[141] OpenAI. OpenAI Dota 2 1v1 bot, 2017. URL `https://openai.com/the-international/`.

[142] J. O'Rourke. Art Gallery Theorems and Algorithms. *International Series of Monographs on Computer Science*, 1987.

[143] M. Otte and E. Frazzoli. RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7):797–822, 2015.

[144] P. Bovbel. Pointcloud-to-laserscan ROS package, 2020. URL `http://wiki.ros.org/pointcloud_to_laserscan`.

[145] D. Panagou and H. Tanner. Modeling of a Hexapod Robot; Kinematic Equivalence to a Unicycle. *UDME Technical Report Number UDMETR-2009-0001*, 2009.

[146] A. A. Paranjape, K. C. Meier, X. Shi, S.-J. Chung, and S. Hutchinson. Motion primitives and 3D path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015.

[147] S. Paternain, D. E. Koditschek, and A. Ribeiro. Navigation Functions for Convex Potentials in a Space with Convex Obstacles. *IEEE Transactions on Automatic Control*, 2017. ISSN 0018-9286. doi: 10.1109/TAC.2017.2775046.

[148] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DoF object pose from semantic keypoints. In *IEEE International Conference on Robotics and Automation*, pages 2011–2018, May 2017.

[149] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380, 2006.

[150] F. T. Pokorny and D. Kragic. Data-Driven Topological Motion Planning with Persistent Cohomology. In *Robotics: Science and Systems*, 2015.

[151] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[152] R. A. Poliquin, R. T. Rockafellar, and L. Thibault. Local Differentiability of Distance Functions. *Transactions of the American Mathematical Society*, 352(11):5231–5249, 2000.

[153] PolyDecomp. poly_decomp: Decompose 2D polygons into convex pieces, 2019. URL `https://github.com/wsilva32/poly_decomp.py`.

[154] Qualisys. Oqus platform, 2020. URL `http://www.qualisys.com/cameras/oqus/`.

[155] 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.

[156] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, R. Berka, R. Ambrose, M. Myles Markee, N. J. Fraser-Chanpong, C. McQuin, J. D. Yamokoski, S. Hart, R. Guo, A. Parsons, B. Wightman, P. Dinh, B. Ames, C. Blakely, C. Edmondson, B. Sommers, R. Rea, C. Tobler, H. Bibby, B. Howard, L. Niu, A. Lee, M. Conover, L. Truong, R. Reed, D. Chesney, R. Platt Jr, G. Johnson, C.-L. Fok, N. Paine, L. Sentis, E. Cousineau, R. Sinnet, J. Lack, M. Powell, B. Morris, A. Ames, and J. Akinyode. Valkyrie: NASA's First Bipedal Humanoid Robot. *Journal of Field Robotics*, 32(3): 397–419, 2015.

[157] Alireza Ramezani, Jonathan W. Hurst, Kaveh Akbari Hamed, and J. W. Grizzle. Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot. *Journal of Dynamic Systems, Measurement, and Control*, 2013.

[158] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[159] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini. Towards a multi-legged mobile manipulator. In *IEEE International Conference on Robotics and Automation*, pages 3618–3624, 2016.

[160] P. Reverdy, B. D. Ilhan, and D. E. Koditschek. A drift-diffusion model for robotic obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6113–6120, 2015.

[161] P. B. Reverdy and D. E. Koditschek. A dynamical system for prioritizing and coordinating motivations. *SIAM Journal on Applied Dynamical Systems*, 17(2):1683–1715, 2018.

[162] S. Revzen, B. D. Ilhan, and D. E. Koditschek. Dynamical Trajectory Replanning for Unknown Environments. In *IEEE Conference on Decision and Control*, pages 3476–3483, 2012.

[163] E. Rimon. *Exact robot navigation using artificial potential functions*. PhD thesis, Yale University, 1990.

[164] E. Rimon and D. E. Koditschek. The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds. *Transactions of the American Mathematical Society*, 327(1):71–116, 1989.

[165] E. Rimon and D. E. Koditschek. Exact Robot Navigation Using Artificial Potential Functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[166] C. Rockey and M. O'Driscoll, 2020. URL `http://wiki.ros.org/urg_node`.

[167] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[168] V. L. Rvachev. An analytic description of certain geometric objects. In *Doklady Akademii Nauk*, volume 153, pages 765–767. Russian Academy of Sciences, 1963.

[169] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. *Arxiv preprint: 180300653*, 2018.

[170] B. Schling. *The Boost C++ Libraries*. XML Press, 2011.

[171] J. Scholz, N. Jindal, M. Levihn, C. L. Isbell, and H. I. Christensen. Navigation Among Movable Obstacles with learned dynamic constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3706–3713, 2016.

[172] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *arXiv: 1502.05477*, 2015.

[173] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, and S. Schüller. NimbRo Rescue: Solving Disaster-response Tasks with the Mobile Manipulation Robot Momaro. *Journal of Field Robotics*, 34(2):400–425, 2017.

[174] Shapely. Shapely: Manipulation and analysis of geometric objects, 2019. URL https://github.com/Toblerity/Shapely.

[175] A. Shapiro. Sensitivity Analysis of Nonlinear Programs and Differentiability Properties of Metric Projections. *SIAM Journal on Control and Optimization*, 26(3):628–645, 1988. doi: 10.1137/0326037.

[176] V. Shapiro. Semi-analytic geometry with R-functions. *Acta Numerica*, 16:239–303, 2007.

[177] A. Shariati. pulson_ros, 2020. URL https://github.com/ashariati/pulson_ros.

[178] W. B. Shen, D. Xu, Y. Zhu, L. J. Guibas, L. Fei-Fei, and S. Savarese. Situational Fusion of Visual Representation for Visual Navigation. In *IEEE International Conference on Computer Vision*, pages 2881–2890, 2019.

[179] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada. SMC: Satisfiability Modulo Convex Programming. *Proceedings of the IEEE*, 106(9):1655–1679, 2018.

[180] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3375–3382, 1996.

[181] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation*, pages 639–646, 2014.

[182] StereoLabs. Positional Tracking with the ZED Mini stereo camera, 2019. URL https://www.stereolabs.com/docs/positional-tracking/.

[183] StereoLabs. ZED Mini Stereo Camera, 2019. URL https://www.stereolabs.com/zed-mini/.

[184] M. Stilman and J. Kuffner. Planning Among Movable Obstacles with Artificial Constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[185] T. Strom-Hansen, M. Thor, L. B. Larsen, E. Baird, and P. Manoonpong. Distributed Sensor-Driven Control for Bio-Inspired Walking and Ball Rolling of a Dung Beetle-Like Robot. In *SWARM*, pages 196–199, 2017.

[186] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv: 1804.10332*, 2018.

[187] S. Tang and V. Kumar. Autonomous Flight. *Annual Review of Control, Robotics and Autonomous Systems*, 1(1):29–52, 2018.

[188] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019. URL `https://doc.cgal.org/4.14/Manual/packages.html`.

[189] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[190] Time Domain. P-440, 2020. URL `https://usermanual.wiki/Humatics/P440-A`.

[191] T. T. Topping, V. Vasilopoulos, A. De, and D. E. Koditschek. Composition of templates for transitional pedipulation behaviors. In *The International Symposium on Robotics Research*, 2019.

[192] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[193] Tripy. tripy: Simple polygon triangulation algorithms in pure python, 2019. URL `https://github.com/linuxlewis/tripy`.

[194] TurtleBot2. TurtleBot2: Open-source robot development kit for apps on wheels, 2019. URL `https://www.turtlebot.com/turtlebot2/`.

[195] Unitree Robotics. Laikago: a four leg robot is coming to you, 2017. URL `https://www.youtube.com/watch?v=d6Ja643GqL8`.

[196] G. Vallicrosa and P. Ridao. Sum of gaussian single beacon range-only localization for AUV homing. *Annual Reviews in Control*, 42(Supplement C):177–187, 2016.

[197] J. van den Berg, M. Lin, and D. Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation*, pages 1928–1935, 2008.

[198] J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha. Path planning among movable obstacles: A probabilistically complete approach. In *The 8th International Workshop on the Algorithmic Foundations of Robotics*, 2010.

[199] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. *Reciprocal n-Body Collision Avoidance*, pages 3–19. Springer Berlin Heidelberg, 2011.

[200] V. Vasilopoulos and D. E. Koditschek. Reactive Navigation in Partially Known Non-Convex Environments. In *13th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[201] V. Vasilopoulos, O. Arslan, A. De, and D. E. Koditschek. Sensor-Based Legged Robot Homing Using Range-Only Target Localization. In *IEEE International Conference on Robotics and Biomimetics*, pages 2630–2637, 2017.

[202] V. Vasilopoulos, T. T. Topping, W. Vega-Brown, N. Roy, and D. E. Koditschek. Sensor-Based Reactive Execution of Symbolic Rearrangement Plans by a Legged Mobile Manipulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3298–3305, 2018.

[203] V. Vasilopoulos, W. Vega-Brown, O. Arslan, N. Roy, and D. E. Koditschek. Sensor-Based Reactive Symbolic Planning in Partially Known Environments. In *IEEE International Conference on Robotics and Automation*, pages 5683–5690, 2018.

[204] V. Vasilopoulos, G. Pavlakos, K. Schmeckpeper, K. Daniilidis, and D. E. Koditschek. Reactive Navigation in Partially Familiar Planar Environments Using Semantic Perceptual Feedback. *Under review, arXiv pre-print: 2002.08946*, 2019.

[205] V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek. Reactive Semantic Planning in Unexplored Semantic Environments Using Deep Perceptual Feedback. *IEEE Robotics and Automation Letters*, 5(3):4455–4462, July 2020.

[206] V. Vasilopoulos, Y. Kantaros, G. J. Pappas, and D. E. Koditschek. Reactive Planning for Mobile Manipulation Tasks in Unexplored Semantic Environments. In *IEEE International Conference on Robotics and Automation*, 2021.

[207] VectorNav Technologies. VN-100, 2019. URL `https://www.vectornav.com/products/vn-100`.

[208] W. Vega-Brown and N. Roy. Asymptotically optimal planning under piecewise-analytic constraints. In *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[209] W. Vega-Brown and N. Roy. Anchoring abstractions for near-optimal task and motion planning. In *RSS Workshop on Integrated Task and Motion Planning*, 2017.

[210] W. Vega-Brown and N. Roy. Admissible abstractions for near-optimal task and motion planning. In *International Joint Conference on Artificial Intelligence*, 2018.

[211] M. Vendittelli, J.-P. Laumond, and B. Mishra. *Decidability of Robot Manipulation Planning: Three Disks in the Plane*, pages 641–657. Springer Tracts in Advanced Robotics. Springer, Cham, 2015. ISBN 978-3-319-16594-3. doi: 10.1007/978-3-319-16595-0_37. URL `https://link.springer.com/chapter/10.1007/978-3-319-16595-0_37`.

[212] Vicon. Vantage V16, 2020. URL `https://www.vicon.com/hardware/cameras/vantage/`.

[213] P. Vlantis, C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos. Robot navigation in complex workspaces using harmonic maps. In *IEEE International Conference on Robotics and Automation*, pages 1726–1731, 2018.

[214] J. Wang and E. Olson. AprilTag 2: Efficient and robust fiducial detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4193–4198, 2016.

[215] L. L. Whitcomb, D. E. Koditschek, and J. B. D. Cabrera. Toward the Automatic Control of Robot Assembly Tasks via Potential Functions: The Case of 2-D Sphere Assemblies. In *IEEE International Conference on Robotics and Automation*, pages 2186–2192, 1992.

[216] J. Wolfe, B. Marthi, and S. Russell. Combined task and motion planning for mobile manipulation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2010.

[217] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert. Autonomous navigation for BigDog. In *IEEE International Conference on Robotics and Automation*, pages 4736–4741, 2010.

[218] M. Zucker, S. Joo, M. X. Grey, C. Rasmussen, E. Huang, M. Stilman, and A. Bobick. A General-purpose System for Teleoperation of the DRC-HUBO Humanoid Robot. *Journal of Field Robotics*, 32(3):336–351, 2015.