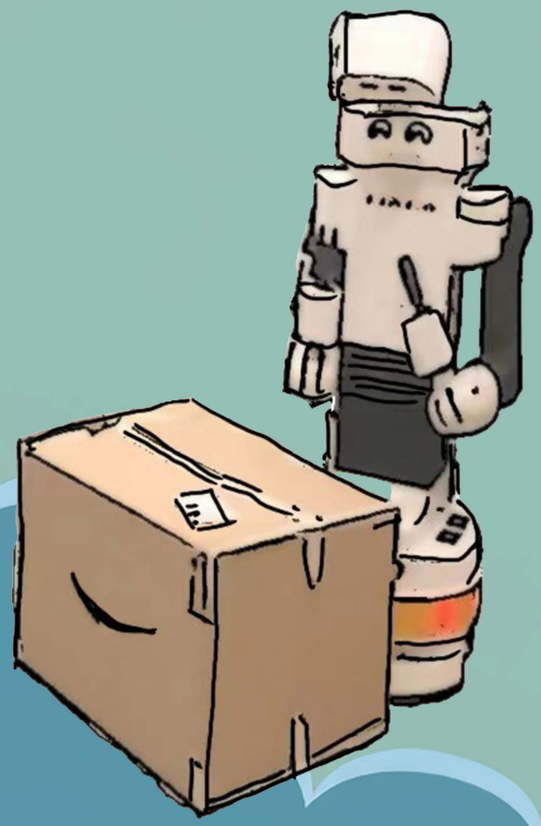
Unwieldy Object Delivery with Mobile Robots

Yujie Tang



Unwieldy Object Delivery with Mobile Robots

Yujie Tang

Unwieldy Object Delivery with Mobile Robots

Yujie Tang
唐玉洁

Unwieldy Object Delivery with Mobile Robots

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Monday 31 March 2025 at 17:30 o'clock

by

Yujie TANG

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof.dr. W. Wisse	Delft University of Technology, promotor
Dr. W. Pan	TU Delft / University of Manchester, UK, copromotor
Independent members:	
Prof. G.C.H.E. de Croon	Delft University of Technology
Prof. Dr. Ing. H. Vallery	RWTH Aachen & TU Delft & Erasmus MC
Prof. J. Xiao	Worcester Polytechnic Institute
Prof. K. Lynch	Northwestern University
Reserve member:	
Dr.ir. R. Happee	Delft University of Technology

The research described in this thesis was supported by the China Scholarship Council (CSC) grant 202006890020 and the Cognitive Robotics department at Delft University of Technology.

Published and distributed by: Yujie Tang
E-mail: yujietangtang@gmail.com

Keywords:	Motion planning, robot manipulation, nonprehensile manipulation, robot control, trajectory planning, state estimation, contact modeling
Front & Back:	Yujie Tang, Hai Zhu & ChatGPT
Printed by:	ProefschriftMaken

ISBN: 978-94-6510-558-1

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

The important thing is not to stop questioning. Curiosity has its own reason for existence.
— Albert Einstein

Contents

Summary

This thesis presents a comprehensive exploration of unwieldy object delivery using mobile robots, focusing on the challenges and advancements in Navigation Among Movable Objects (NAMO). The research addresses critical issues in robotic manipulation, particularly nonprehensile techniques such as pushing, which are essential for handling objects that are difficult to grasp.

The study is structured around four key research questions guiding the investigation into effective pushing manipulation strategies:

1. How to perform pushing manipulation with limited or inaccurate state estimation:

   We employ the concept of "stable pushing," ensuring the robot can always "catch" the object during delivery. The stable pushing control problem is simplified as an optimization problem with a concise linear constraint. Experiments show that this approach outperforms reactive pushing strategies, reducing the robot's traveled distance by 23.8% and time by 77.4%.

2. How to improve pushing manipulation when accurate state estimation is available:

   We enhance the stable pushing approach to a more maneuverable "free pushing" method, allowing contact point changes to improve manipulation mobility. This approach achieves an average success rate of 83% with an accuracy of 0.085m when pushing to selected goals, demonstrating improved agility and efficiency compared to stable pushing.

3. How to achieve efficient, real-time global trajectory optimization for contact-rich pushing:

   By investigating the differential flatness property of the pushing system, we simplify the pushing planning problem, significantly reducing computational complexity. This transformation allows for a simpler contact-implicit planning task that is easy to design, fast to solve, and robust to uncertainties.

4. How to perform robust state estimation through sensor fusion:

   We integrate data from multiple sensors and improve the robustness of the classic Kalman Filter using a deep reinforcement learning (DRL) algorithm. This novel DRL-based orientation estimation method guarantees bounded estimation errors without the need for hyperparameter tuning. Experiments demonstrate its superior performance compared to conventional methods, particularly in challenging scenarios with inaccurate initial state estimates, imprecise filter gains, and non-Gaussian noise environments.

The key contributions of this thesis include:

- A stable pushing approach that simplifies the optimization problem for nonholonomic mobile bases.
- A maneuverable free pushing method that enhances agility while maintaining contact.
- A reactive manipulation strategy leveraging differential flatness for efficient trajectory planning.
- A reinforcement learning-based approach to improve state estimation accuracy.

In conclusion, this research significantly advances the capabilities of mobile robots in handling unwieldy objects, bridging the gap between theoretical navigation planning and practical applications in complex environments. The findings pave the way for future research and broader applications of mobile robots in various domains, including logistics, search and rescue, and autonomous inspections.

Samenvatting

Dit proefschrift presenteert een diepgaande onderzoek naar de levering van onhandelbare objecten met behulp van mobiele robots, met de nadruk op de uitdagingen en vooruitgang in Navigatie Tussen Verplaatsbare Objecten (NTVO). Het onderzoek richt zich op kritieke vraagstukken in robotmanipulatie, met name technieken als duwen, die essentieel zijn voor het hanteren van objecten die moeilijk te grijpen zijn.

Het onderzoek is gestructureerd rond vier belangrijke onderzoeksvragen rondom effectieve manipulatie bij duwbewegingen:

1. Hoe kunnen manipulatie door duwbewegingen uitgevoerd worden met beperkte en onnauwkeurige bepalingen van de huidige configuratie:

   We introduceren het concept van “stabiel duwen,” waarbij we ervoor zorgen dat de robot het object altijd kan “vangen” tijdens transport. Het probleem van stabiel duwen wordt vereenvoudigd tot een optimalisatieprobleem met een lineare voorwaarde. Experimenten tonen aan dat deze aanpak beter presteert dan reactieve duwstrategieën, waarbij de afgelegde afstand van de robot met 23,8% en de tijd met 77,4% wordt verminderd.

2. Hoe verbeter je duwmanipulatie wanneer het mogelijk is om de configuratie van de robot nauwkeurig te bepalen:

   We verbeteren de stabiele duwmethode tot een "vrije duwmethode"die meer manouevres toestaat, waarbij veranderingen in de contactpunten worden toegestaan om de bewegelijkheid van de manipulatie te verbeteren. Deze aanpak bereikt een gemiddeld slagingspercentage van 83% met een nauwkeurigheid van 0,085m bij het duwen naar geselecteerde doelen, wat een verbeterde wendbaarheid en efficiëntie aantoont in vergelijking met stabiel duwen.

3. Hoe kan efficiente en realtime trajectoptimalisatie bereikt worden voor contactrijk duwen:

   Door de differentiële vlakheidseigenschap van het systeem te onderzoeken, wordt het oorspronkelijke probleem betreffende het duwen van objecten vereenvoudigd, waardoor de computationele complexiteit aanzienlijk wordt verminderd. Deze transformatie zorgt voor een doeltreffende contact-impliciete planning die eenvoudig te ontwerpen en op te lossen is, en robuust is tegen onzekerheden.

4. Is het mogelijk door middel van het fuseren van informatie uit verschillende sensoren, een stabiele schatting te maken van de huidige configuratie:

We integreren gegevens van meerdere sensoren en verbeteren de stabiliteit van de klassieke Kalman Filter met behulp van een deep reinforcement learning (DRL) algoritme. Deze nieuwe DRL-gebaseerde oriëntatieschattingsmethode garandeert gelimiteerde schattingsonnauwkeurigheden zonder dat handmatige hyperparametrizatie bepaling nodig is. Experimenten tonen de superieure prestaties aan in vergelijking met conventionele methoden, vooral in uitdagende scenario's met onnauwkeurige initiële configuratiesbepalingen, onnauwkeurige filterwinsten en niet-Gaussische ruisomgevingen.

De belangrijkste bijdragen van dit proefschrift zijn:

- Een stabiele duwaanpak die het optimalisatieprobleem voor niet-holonome mobiele bases vereenvoudigt.
- Een manoeuvreerbare vrije duwmethode die de wendbaarheid verbetert terwijl het contact behouden blijft.
- Een reactieve manipulatiestrategie die gebruik maakt van differentiële vlakheid voor efficiënte trajectplanning.
- Een op reinforcement learning gebaseerde benadering om de nauwkeurigheid van configuratiesbepalingen te verbeteren.

Concluderend, dit onderzoek verbetert de mogelijkheden van mobiele robots bij het hanteren van moeilijk hanteerbare objecten aanzienlijk, en overbrugt de kloof tussen theoretische navigatieplanning en praktische toepassingen in complexe omgevingen. De bevindingen banen de weg voor toekomstig onderzoek en bredere toepassingen van mobiele robots in verschillende domeinen, waaronder logistiek, zoek- en reddingsacties, en autonome inspecties.

Acknowledgments

Pursuing a PhD in a foreign country was a brave decision I made. At the start, I had doubts about completing such an ambitious project in the Netherlands. However, I am proud to have persevered and to now be approaching the finish line. It has been a long journey enriched by the tremendous support of those around me, and I am grateful to have met so many wonderful people who have made this journey both interesting and memorable.

First and foremost, I would like to thank my supervisors. Throughout these four years, I experienced cycles of confidence and self-doubt, facing critical comments both from myself and reviewers. I am deeply grateful to have had such a supportive promoter, Martijn, always by my side. Thank you for your consistent willingness to help, your comprehensive suggestions, and for giving me the freedom to make my own choices. Your most frequent saying, "This is your PhD. You make your own choice," introduced me to a professor-student relationship unlike any I had previously known. I would also like to thank my supervisor Wei Pan for giving me the opportunity to start my PhD at TU Delft. Your fascinating ideas never failed to inspire me. Thank you for teaching me reinforcement learning hands-on in my first year and for helping me revise my manuscripts until the last minute. The early publication under your supervision significantly reduced my pressure in later years. Finally, I would like to thank Prof. Yangmin Xie and Prof. Hang Shi, who supervised my master's studies. You opened the door to academia for me and provided excellent academic training, building the foundation for my PhD research.

Beyond my supervisors, I am grateful for the senior researchers in our sections who have been incredibly supportive. Chris, your positive, optimistic, and cheerful attitude has always been contagious and uplifting. Thank you for your endless encouragement. Thanks to Max Spahn, and Giovanni for the inspiring research discussions. Thanks to Guopeng for the valuable suggestions during my job search. Thanks to Chadi, Arend-Jan, and Kseniia for helping with my experiments, and to Lasse for assistance with Julia.

Beginning my PhD during the pandemic made me especially appreciate the time working in the office. I would like to express my gratitude to my officemates for their companionship. Luzia, who started her PhD around the same time as me, became not just a colleague but a dear friend. We were surprised to discover we lived in the same building, and this proximity nurtured our friendship over four years. Thank you for your patience, even when I accidentally gave your key to a stranger, and for always encouraging me to leave my comfortable sofa, not being a sofa potato. I appreciate you explaining European traditions and helping me rephrase my thoughts when cultural misunderstandings arose.

When we returned to normal office work, I met Elia, Maximilian, and Anna. I'm grateful for experiencing many exciting activities with you, from go-karting to bouldering and skiing.

Later, Saray and Khaled joined us. Saray, thank you for teaching us to bake Limburgse vlaai and for your nice dinner invitations. Khaled, thanks for being my bouldering partner and practicing the green routes together. Special thanks to our additional officemate, cake master Julian, for all your delicious bakes and initiating the wonderful Monday cake breaks.

I must acknowledge other wonderful colleagues in the department: Tasos and Jelle for their support during our Swiss hiking adventure; Tomás and Rodrigo for the memorable times in Japan; and Dennis, Lorenzo, Gustavo, Ashwin, Manuel, Andrea and Mariano for the pleasant conversations in the social room. Thanks to my Chinese colleagues (Zhaochong, Jingyue, Chuhan, Cong, Zhaoting, Gang, Jiatao, Renchi) for the board game sessions and shared Chinese meals!

Outside the faculty, I was fortunate to have wonderful neighbors. Xiaohuan, Jing, Yifei, Desong, and Tianlong, thank you for your company during my first year. Our trips exploring this new country together helped prevent any homesickness. Yiru and Ziqing, I cherished our museum visits and art appreciation sessions. Our time-to-time corridor chats made living alone feel less lonely.

I'm grateful to Sihao, Sherry, and Lianlian for organizing wonderful dinners and board game evenings. Your garden barbecues are particularly missed. Thanks to Chen, my downstairs neighbor, for sharing interesting "Haiguitang" stories and installing my fancy ceiling light. Thanks to Yingfu for your encouragement and rescue on the black slope in Stubai – the photo you took captures that "memorable" moment.

Finally, Delft gave me not just a PhD degree but also a lovely boyfriend. Though maintaining a long-distance relationship has been challenging, thank you for not giving up and for encouraging me to complete my studies. To my parents, thank you for your unwavering support. I'm deeply grateful for your understanding and encouragement, which have allowed me to explore this vast world.

Yujie Tang
Delft, October 2024

1

Introduction

1

1.1 Motivation

Mobile robots have become integral to various autonomous tasks, including inspections, monitoring, logistics, and search and rescue operations. Traditional autonomous navigation approaches focus on safe movement and collision avoidance, but real-world scenarios often present more complex challenges. Robots may need to navigate through cluttered rooms or debris-blocked tunnels, requiring more than simple obstacle avoidance. These situations demand the ability to interact with and reconfigure the environment (as shown in Fig. 1.1), giving rise to the field of Navigation Among Movable Objects (NAMO) [1].

Current NAMO research primarily focuses on optimizing navigation costs by determining which objects to move and where to place them. Recent studies have explored various aspects of NAMO optimization. Zhang et al. [2] focused on minimizing total time cost in route planning. Considering social factors, Renault et al. [3] incorporated strategies to reduce disturbance to humans during object placement. Additionally, researchers like Wang et al. [4] and Meng et al. [5] investigated object properties such as movability and affordance to inform decision-making processes in NAMO scenarios. However, a crucial aspect is overlooked: how robots should interact with these objects and precisely deliver them to the planned place.

In an attempt to address this gap, Wang et al. [6] combined navigation and interaction planning in NAMO through Reinforcement Learning. However, learning a policy for NAMO is a long-horizon task with sparse rewards which makes the NAMO policy learning challenging. To simplify planning, Zhang et al. [2] proposed simple representations for interaction actions, only assuming basic attach and detach actions. This approach, however, ignores the complexities of object graspability, which depends on the object's mass and shape, as noted by Scholz et al. [7].

Obstacles encountered during navigation are often too heavy and unwieldy for effective grasping. The limited payload capacity of robot arms presents another challenge. For instance, the widely used Franka Panda has a maximum payload of just 3 kg. Therefore, Ellis et al. [8] and Zherdev et al. [9] propose a method to push the object to the goal instead of grasping them. Compared to grasping, pushing with the mobile base is more versatile and cost-effective. However, these methods often consider only simple straight-forward pushing, limiting object placement options and preventing optimization.

This research aims to tackle the key challenge in NAMO: how to effectively manipulate obstacles. Our study explores a wide range of object manipulation techniques, focusing particularly on efficiently pushing large, heavy objects using mobile robot bases. This approach overcomes the payload limitations of robot arms, which often struggle with heavy objects. Furthermore, it targets to expand the possibilities for object placement and trajectory optimization beyond simple straight-line pushing. The completion of this research will contribute to the NAMO field by enhancing mobile robots' capabilities in handling unwieldy obstacles. Consequently, this work will bridge the gap between theoretical navigation planning and its practical application in complex environments with movable obstacles, paving the way for broader mobile robot applications.

Figure 1.1: A Husky robot navigating a supermarket environment encounters baskets blocking its path. To clear the way, it attempts to push the baskets aside before reversing to realign with its predefined path.

1.2 Research questions

This thesis focuses on unwieldy object delivery using mobile bases, addressing the "how" problem in Navigation Among Movable Objects (NAMO). Specifically, we explore nonprehensile manipulation techniques, particularly pushing, which involve manipulating objects without grasping them. This approach is more cost-effective and flexible than traditional grasping methods, making it particularly suitable for mobile robots handling unwieldy objects.

In robotic manipulation, particularly nonprehensile manipulation where objects are not firmly grasped, accurate state estimation is crucial for effective action. This is especially relevant for mobile base pushing, where the robot must continuously adjust its position relative to the object, requiring a sequence of precise interactions. Our work addresses the challenges of pushing manipulation with mobile bases by exploring four key research questions:

1. How can we perform pushing manipulation with mobile bases when state estimation information is limited or inaccurate?

Our first research question addresses the challenge of performing pushing manipulation with a mobile base when state estimation is limited or unreliable. In such cases, ensuring consistent and effective contact between the robot and the object is crucial. Mason [10] introduced the concept of "stable pushing", which maintains constant robot-object contact to minimize repositioning and reduce dependence on state estimation. However, stable pushing relies on the motion constraint at the contact point. Most commonly used mobile bases are differentially-driven, inherently subject to nonholonomic constraints. When these constraints are combined with those of stable pushing, they result in a complex set of constraints in the optimization problem. So, how can we design a stable pushing method

1

for nonholonomic mobile robots considering its nonholonomic constraint?

2. How can we improve pushing manipulation when accurate state estimation is available?

Our second research question focuses on improving pushing manipulation when accurate state estimation is available. While stable pushing benefits from reduced reliance on state estimation during the process, it suffers from limited object mobility due to the need to maintain stiff contact. To enhance pushing agility, planners that allow relative sliding between the robot and object [11–14] have been proposed. These methods utilize varying contact modes (sticking, right-slide, and left-slide) to apply the necessary pushing force on the object. However, executing these planned pushes on real robot platforms may sometimes prove unreachable due to the robot's kinematic limitations [11]. So, how can we design a more maneuverable pushing approach while still ensuring the feasibility of the plans?

3. How can we achieve efficient, real-time global trajectory optimization for contact-rich pushing?

The underactuated nature of nonprehensile manipulation systems often leads local controllers to become trapped in suboptimal solutions, necessitating global trajectory planning. Unlike stable pushing with its fixed contact point, more maneuverable approaches allow for dynamic contact point changes during manipulation. To plan for the contact changes, existing methods frequently rely on computationally intensive techniques [15] or are limited to short-horizon tasks [14], rendering them impractical for real-time applications. The complexity is primarily due to the highly nonlinear dynamics involved, giving rise to our third research question: How can we simplify the planning for contact-rich pushing to overcome these limitations and enable efficient, real-time global trajectory optimization?

4. How can we estimate the states robustly through sensor fusion?

The fundamental challenge of state estimation in nonprehensile manipulation remains crucial for ensuring effective robot-object interactions. This challenge is twofold, necessitating accurate estimation of both the robot's and the object's state. To enhance estimation accuracy, we often employ sensor fusion, integrating data from multiple sensors. This critical aspect leads to our fourth research question: How can we achieve robust state estimation through the effective fusion of data from diverse sensor sources to improve the overall performance and reliability of nonprehensile manipulation systems?

1.3 Contributions

To solve the research questions proposed in Section. 1.2, several contributions are made in this thesis:

A Stable Pushing Approach for Unwieldy Object Delivery with Nonholonomic Mobile Base: We propose a stable pushing method for the nonholonomic mobile base pushing problem that maintains stiff contact between the robot and the object to minimize repositioning actions and reduce the reliance on state estimation. We prove that a line contact, rather than a single point contact, is necessary for nonholonomic robots to achieve stable pushing. We also show that the stable pushing constraint and the nonholonomic constraint of the robot can be simplified as a concise linear motion constraint. Then, the pushing planning problem can be formulated as a constrained optimization problem using nonlinear model predictive control (NMPC). According to the experiments, our NMPC-based planner outperforms a reactive pushing strategy in terms of efficiency, reducing the robot's traveled distance by 23.8% and time by 77.4%. Furthermore, our method requires four fewer hyperparameters and decision variables than the Linear Time-Varying (LTV) MPC approach, making it easier to implement. Real-world experiments are carried out to validate the proposed method with two differential-drive robots, Husky and Boxer, under different friction conditions.

A Maneuverable Free Pushing Approach with Contact Point Changing: A more maneuverable push approach, "free pushing", is proposed that allows the transportation of objects using mobile bases while considering the feasibility of the plans. Unlike previous *stable pushing* methods, which maintain a stiff robot-object contact, our approach uses the state estimation information of the object and allows the robot to maneuver around the object while pushing it. It aims to execute continuous pushes without losing contact for improved pushing maneuverability. Additionally, to ensure the feasibility of the planned pushes, a robot-object contact model is developed to account for the shape and kinematics of the robot in pushing modeling and planning. A Model Predictive Controller solves the pushing planning problem in real-time. Experimental results show that the proposed method achieves an average success rate of 83% with an accuracy of 0.085m when pushing to the selected goals. Compared to the stable pushing method, this approach improves the agility and efficiency of mobile pushers. Furthermore, it is robust in achieving the task while tolerating modeling errors.

Nonprehensile Planar Manipulation via Differential Flatness: The planning and control of nonprehensile manipulation are notoriously difficult due to the hybrid nature of contact dynamics. Instead of solving the planning problem directly, we exploit the differential flatness property of the pushing system to model its dynamics in a simplified form. With the differential flatness property, we are able to transform the manipulation problem into a simpler trajectory planning task, thus achieving significant reductions in computational complexity. The proposed method has the following advantages compared to the other methods: 1) It is simple to design, fast to solve, and robust to uncertainties. 2) Unlike other approaches, it has the advantage of simplicity with only one parameter to tune. No learning or extra tactile sensors are required. 3) The global planner extends the

usage of the free-pushing controller even in cluttered environments.

Reinforcement Learning Compensated Extended Kalman Filter for State Estimation: To achieve better state estimation results under nonprehensile manipulation, we get the state estimation by fusing the observation from multiple sensors. Furthermore, we improve the robustness of the classic Kalman Filter with a deep reinforcement learning (DRL) algorithm. The convergence of state estimation errors is proved with Lyapunov's method in control theory. Based on the theoretical results, the estimator gains and a Lyapunov function are parametrized by deep neural networks and learned from samples. However, in this thesis, the proposed method is only tested in an attitude estimation task using Inertial Measurement Units (IMU) sensors. The extension to the state estimation task in nonprehensile manipulation is left as our future work. The proposed DRL estimator is compared with three well-known orientation estimation methods on both numerical simulations and real datasets. The results show that the proposed algorithm is superior for arbitrary estimation initialization and can adapt to very large velocities for which other algorithms can hardly be applicable. To the best of our knowledge, this is the first DRL-based orientation estimation method with an estimation error boundedness guarantee.

1.4 Outline

Figure 1.2 outlines the structure of the thesis. Chapter 2 reviews related works in unwieldy object delivery using mobile robots and sensor fusion techniques. Chapter 3 presents a stable pushing approach for scenarios where state estimation information is incomplete or unavailable. Chapter 4 introduces a more maneuverable free pushing method, which demonstrates improved pushing agility but requires constant, accurate state estimation of the object. Chapter 5 extends the free pushing method by exploiting its differential flatness property, simplifying the complex manipulation planning problem into a trajectory optimization problem for the object. Chapter 6 showcases improved state estimation results, proposing and testing a Deep Reinforcement Learning (DRL)-based state estimation method for attitude estimation fusing IMU sensor data. Finally, Chapter 7 concludes the thesis and provides recommendations for future research.

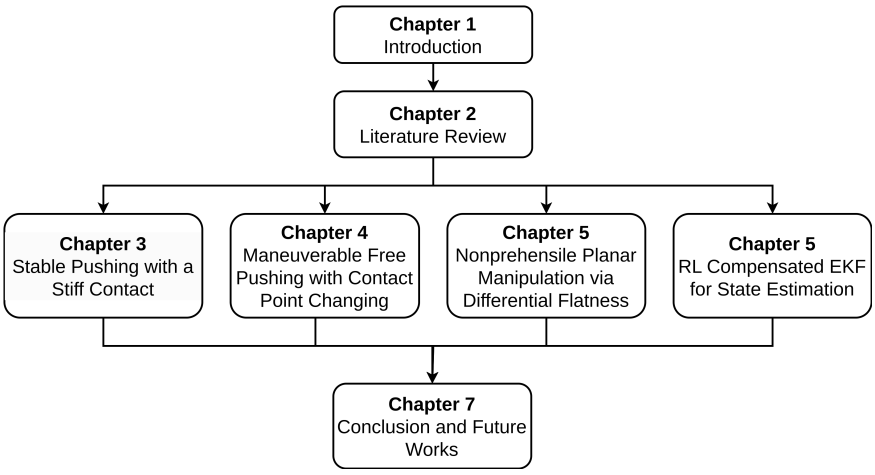


Figure 1.2: Structure of this thesis.

2

Literature review

In this chapter, we summarize related works in two main research fields connected to this thesis topic: nonprehensile pushing manipulation and sensor fusion for state estimation.

2

2.1 Nonprehensile pushing manipulation

In the class of nonprehensile manipulation, pushing received the most attention for its high flexibility and efficiency in completing a task [16–18]. Pushing with robot arms has been extensively studied [13, 19, 20]. However, the delivered object may sometimes be either too heavy or too large for the robot arm to grasp. In this case, one option is to manipulate the object by pushing it with the robot arm [21]. Alternatively, the robot base can push the object. Although there are some works that have studied pushing with robot arms, research on pushing with mobile robots is still limited. This thesis focuses on the nonprehensile pushing manipulation for unwieldy object delivery with mobile bases.

2.1.1 Reactive manipulation

Considering the difficulty in modeling the contacts, early research uses some reactive manipulation approaches. Nieuwenhuisen et al. investigated reactively push objects along the environment boundaries in [18, 22]. Compliance was used as an aid to push the object and to compensate for uncertainty under contact. This approach also simplified the problem of pushing in cluttered environments, eliminating the need to find feasible paths in free space. However, the method was limited to disk-shaped pushers and objects and could only be applied in environments with smooth boundaries, which are rare in the real world.

To achieve practical pushing performance, Krivić et al. proposed new reactive controllers [23, 24]. Their basic idea was to keep the robot, object, and goal in a line to push the object toward the goal. Adaptive proportional controllers were employed to deliver the object along these pre-defined trajectories. Nevertheless, the method was limited to pushing small-sized objects with circular or point-sized robots, making it easy to reposition around the object to change the pushing direction. The method was improved to deal with obstacle avoidance while delivering the object by proposing a pushing corridor [25]. However, the physical feasibility of the planned trajectory was not ensured. Furthermore, their numerous parameters and inherent model uncertainties made tuning difficult.

Alternatively, manipulation can be managed by reactively sampling actions according to predictive models [26–28]. Experimental models are first obtained regarding how pushable real-world objects with complex 3D structures move in response to various pushing actions. Then, a rapidly-exploring random tree (RRT) based planner uses past pushing experiences to construct achievable and collision-free pushing plans [26]. Besides experimental models, Pezzato et al. [27] directly use the physical simulator as the predictive model and introduce a sampling-based MPC method that samples input sequences for varied pushing sequences. While these sampling-based approaches mitigate optimization challenges with non-convex constraints and discontinuous dynamics, they require substantial computational resources due to parallel sampling of numerous trajectories.

To avoid heavy computational burden, sampling methods are also used to sample

contact modes to achieve trajectories with longer horizons [29, 30]. Doshi et al. [15] developed a search tree with a predefined depth to handle a fixed number of contact mode switches. However, as the number of potential contact modes increases, so does the search complexity. These methods often focus on basic manipulation motions, such as pushing centrally or forward, leading to frequent failures and inefficient trajectories due to numerous repositioning actions. Zito et al. [31] expanded the search tree by applying random control inputs, guiding the system towards randomly selected points until the goal is reached. Despite its flexibility, this approach does not guarantee plan optimality and depends on the discretization level of control inputs, requiring more extensive searches with finer discretization.

Recent efforts have applied deep learning to nonprehensile planar manipulation, aiming to learn the physics of manipulation [26, 32]. These approaches help manage irregular objects with complex dynamics but struggle to generalize across different objects, necessitating retraining for even minor differences. Reinforcement learning methods have been employed to develop manipulation policies through trial and error without pre-learned dynamics models [33]. These are particularly useful for cluttered environments with location-based attention mechanisms [34]. However, the extensive training required, along with the need for retraining when the scene changes, limits their practicality. Additionally, slight variations in objects or environments can significantly impact the robustness of these methods.

However, all the aforementioned research assumes the use of omnidirectional mobile robots, which can freely move around the object to achieve the planned pushing actions. For widely-used differential drive robots, limited research has been conducted, as the nonholonomic constraint hinders their ability to smoothly push around the object, making pushing planning more complex. This will be the main research focus in Chapters 3 and 4.

2.1.2 Pushing with a stiff contact

In addition to the motion constraints, a significant challenge in robot pushing is the uncertainty about the object's pose after each action [35]. The methods discussed above rely on reactive actions taken after observing the resulting motion of the pushed object. The robot pusher and the object strive to maintain an equilibrium configuration to continue moving together, resembling a "catching" action during navigation [36]. Thus, the crucial aspect of designing a push/navigation controller is ensuring the stability of this "catching" action.

The concept of stable pushing, which establishes a predictable stiff contact between the robot and the object, was proposed based on the mechanics of planar sliding [37]. This idea has been widely used in the field of pushing manipulation, as demonstrated in [12, 38]. In Chapter 3, we also adopt the concept of stable pushing and propose a method that enables a differential-drive robot to push an object without losing contact.

The most related methodology is proposed by Bertoncelli et al. [39], where a Linear Time-Varying (LTV) MPC is used for mobile robots to push an object along a given path. Stable pushing is achieved by optimizing for both the pushing force and the robot control

inputs, which explicitly imposes friction cone constraints. However, we found that it was computationally expensive to solve this optimization problem due to the additional decision variables and constraints. Furthermore, it is not even solvable with commercial solvers such as ACADOS [40]. To address this, a reference trajectory and supplementary linearization are essential components in the solution process. In contrast, our proposed method in Section 3 implicitly constrains the stiff robot-object contact by deriving a concise motion constraint for the robot control input, making it easier to implement. The validation of the proposed method is also shown in both simulation and real-world experiments.

2.1.3 Model predictive control

Although pushing with a stiff contact can reduce uncertainty under manipulation and thus simplify planning for pushing, the fixed contact also limits the range of pushing forces that can be transmitted, thereby restricting the system's maneuverability. It is worth exploring the pushing mechanics and designing a more flexible pushing approach to adapt to more complicated application scenarios.

The challenges in modeling pushing and implementing it within a controller arise from its inherent discontinuities, characterized by (1) transitions between different contact modes (sliding up, sticking, and sliding down) caused by friction, and (2) sudden changes in system dynamics upon contact.

Various solutions have been proposed in the literature to address the problem posed by its discontinuous feature. One approach uses a hybrid model to represent pushing dynamics in different contact modes [13] and employs a mixed-integer program that optimizes both the discrete contact mode sequence and continuous control actions. On the contrary, contact is represented as a complementarity constraint, implicitly planning for contact modes [14, 41]. However, solving a mixed-integer optimization problem or finding a solution for trajectory optimization with complementarity constraints are computationally challenging, especially as the size of the discrete decision variables increases. Therefore, continuous relaxations are often used to simplify the problem. For example, continuous pushes are planned without losing contact with the object in [13], while [41] assumes frictionless contact between the robot and the object.

In addition to using continuous approximations, it is recognized that even accurate analytical dynamics models for pushing are inherently unstable because physics parameters such as inertia and friction can only be approximated [42]. Furthermore, friction parameters may gradually vary over large surfaces, which is hard to model [43]. Thus, simplified models are commonly used instead of seeking an accurate dynamics model, accompanied by real-time control strategies to fill the gaps in model simplification. For instance, ellipsoidal approximation of the limit surface together with the quasi-static interaction assumption has been widely used in robot-pushing applications [12, 15, 44]. This approximation is relatively accurate in determining whether slippage will occur [45], but it cannot quantify the relative motion between the robot and the object. In contrast, a simplified object dynamics model is used to predict the relative motion in [46]. While it plans for the optimal contact force, contact point, and robot control input using two separate model predictive control frameworks, it does not guarantee that the planned contact points are reachable.

A dynamics model for pushing manipulation is preferred to be able to predict the relative motion between the robot and the object such that feasible push plans can be predicted. Considering pushing with robots of different sizes results in distinct object motions and contact point trajectories. Previous work, such as [47, 48], has studied high-level path planning for large-sized disk-shaped robots and objects but has not provided a low-level control approach to achieve the planned path. As Rigo concluded in [46], following the pushing trajectory is equivalent to following the contact point sequence. However, the nonholonomic constraint of a differential-drive robot limits its ability to reach planned contact points. Therefore, planning achievable contact point sequences is critical to the success of push manipulation, which will be researched in Chapter 4.

2.2 Sensor fusion for state estimation

State estimation is crucial in robotics and human motion analysis [49–51]. Recent advancements in sensor technology have significantly improved state estimation accuracy. Sensor fusion, which combines data from multiple sensors, is commonly used to estimate object states more robustly. Various sensor combinations have been explored, such as inertial measurement units (IMU) with magnetometers [52–54], magnetometers with cameras [55], and IMUs with visual sensors [56, 57]. This thesis focuses on orientation estimation using inertial sensors and magnetometers.

Orientation estimation algorithms can be summarised into three categories: (1) Bayesian estimation, (2) optimisation and (3) deep learning. In Bayesian estimation, the well-known extended Kalman filter (EKF) and the unscented Kalman filter (UKF), were used to estimate the orientation [52, 53, 58]. The key idea is to approximate the states by a Gaussian distribution based on the linearisation technique and the deterministic sampling technique, respectively. Furthermore, the complementary filter was developed based on the EKF, which exploits the complementary characteristics of gyroscopes and that of accelerometer and magnetometer at different time scales [54]. In optimisation, the orientation estimation is obtained based on gradient-based optimisation algorithms [59, 60]. Until recently, deep learning was introduced to estimate the orientation [61], in which a deep neural network is trained to mimic the noise distribution of gyroscopes such that accurate orientation estimates can be obtained by open-loop integration of the noise-free gyro measurements. These algorithms showed superior estimation performance empirically. However, the performance can not be theoretically guaranteed, i.e, the orientation estimate error never diverge. In this chapter, we will employ Lyapunov's method in control theory to prove the estimation error boundedness guarantee using samples. Based on the theoretical result, we will develop a reinforcement learning (RL) based algorithm to learn the estimator from samples.

RL was first applied for state estimation in [62]. Motivated by this work, we plan to develop an RL algorithm to learn the estimator gain using samples while the orientation estimator remains the structure of conventional EKF. The key idea is, the estimator gain will be approximated by a deep neural network (DNN) as a function of the sequence of estimate errors. Different from other popular RL algorithms [63–65], the value function will be treated as a Lyapunov function used to guarantee the estimation performance.

Lyapunov's method has been widely used as a basic tool for stability analysis in control theory [66]. To analyze the stability, the key is to find a scalar "energy-like" Lyapunov function for the considered system such that the derivative/difference of Lyapunov function along the state trajectory is semi-negative definite. Nonetheless, the construction/learning of the Lyapunov function is not trivial. In [67], a straightforward approach is proposed to construct the Lyapunov function for nonlinear systems using DNNs. Recently, the asymptotic stability in model-free RL is given for robotic control tasks in [68]. Inspired by the works [67, 68], we will also parameterise the Lyapunov function as a DNN and learn the parameters from samples. Thereafter, a soft actor-critic (SAC) like algorithm [65] that incorporates the Lyapunov boundedness condition in the objective function to be optimised is proposed. By using the learned estimator gain, the estimate error of the orientation estimator is guaranteed to be bounded all the time.

In this thesis, we combine Lyapunov's method and DRL to design a state estimator with estimation error boundedness guarantee for orientation estimation. Furthermore, we try to combine reinforcement learning (RL) with the classic EKF to design a tuning-free estimation algorithm insensitive to inaccurate initial state estimate and filter gain. It leverages the merits of both data-driven RL and model-based probabilistic methods instead of only using RL to train the filter gain [69].

3

A stable pushing approach for unwieldy object delivery with nonholonomic mobile base

Parts of this chapter appeared in:

- **Y. Tang**, H. Zhu, S. Potters, M. Wisse, W. Pan, "Unwieldy Object Delivery With Nonholonomic Mobile Base: A Stable Pushing Approach," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7727-7734, Nov. 2023.

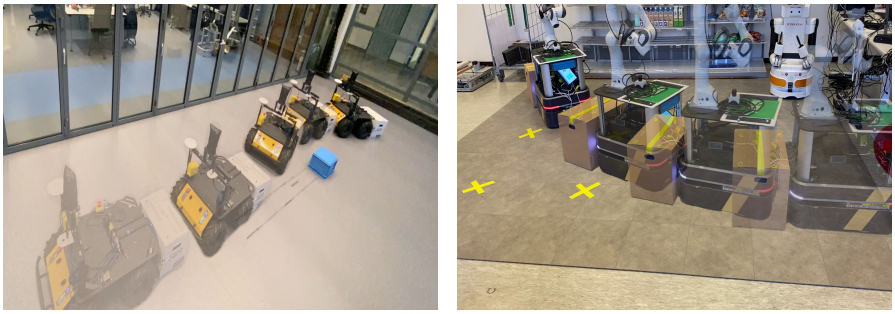
In this chapter, we present a novel stable pushing approach for mobile robots to deliver objects while maintaining a stiff contact between the robot and the object. This method significantly reduces the risk of losing the object during transport, thereby decreasing reliance on robust state estimation. The key challenges we address are twofold: 1) formulating a constraint within the planning optimization problem that ensures a stable contact, and 2) accounting for the nonholonomic constraints of the mobile base. Building upon extensive research in pushing mechanics, we derive a concise linear motion constraint for stable pushing with differential-drive mobile robots. Our approach is easily implementable within existing robot navigation frameworks and achieves a high success rate in real-world applications.

3.1 Introduction

With mobile robots increasingly being used, there are various scenarios in which the robots are expected to perform additional delivery tasks while maneuvering, for example, a robot conveying a package in a warehouse. In this regard, mobile robots equipped with robot arms have become progressively popular. However, the delivered object may be sometimes unwieldy, either too heavy or too large, for the robot arm to grasp. In this case, one option is to manipulate the object by pushing it with the robot arm [21]. Alternatively, the robot can push the object, as shown in Fig. 3.1. Without a robot arm, pushing with the robot expands its manipulation repertoire, making it not just a mobile base. Moreover, it reduces the cost, space, and payload by eliminating the robot arm [70].

Research on pushing with mobile robots is still limited, though pushing with robot arms has been extensively studied [13, 19, 20]. Mobile robots have nonholonomic constraints that restrict their ability to freely reach various planned contact points. As a result, the pushed object is prone to sliding away, requiring time-consuming and effort-consuming repositioning actions to restart pushing. To address this challenge, [36] proposed stable pushing, which involves maintaining a stiff robot-object contact to prevent frequent repositions. The object is effectively rigidly relative to the pusher, and the push is called a stable push [71]. This approach can reduce the risk of losing control over the object resulting in improved efficiency.

As concluded in [44], stable pushing with a single-point contact can be reducible to the Dubins car problem, where the sticking contact constraint is translated to bounded curvatures of the object's trajectory, represented as a motion cone for the object. However, we extend this conclusion by proving that stable pushing is not achievable for a differential-drive mobile robot pushing with a single-point contact, due to the limited friction cone and the nonholonomic constraint of the robot. It can not provide enough friction force to maintain a stiff robot-object contact. As a follow-up study to [44], we introduce a line contact to make stable pushing possible where a larger friction cone can be provided. Based on it, we prove that the stable pushing constraint and robot nonholonomic constraint can be combined as a linear motion constraint on the robot's control input, which greatly simplifies the pushing planning problem compared to [39], as the stable pushing can be guaranteed implicitly with the control constraint.



(a) (b)

Figure 3.1: The wheeled mobile robots (Clearpath Husky and Boxer) push a paper box with and without collision avoidance to the goal position. The blue box in (a) indicates the obstacle. Transparency of the robots and box indicates their movement.

We formulate the goal-conditioned stable pushing problem as a constrained optimization problem by employing Nonlinear Model Predictive Control (NMPC). Our NMPC planner with the concluded motion constraint guarantees that the object's motion is within the motion cone for stable pushing and the physical limitation of the robot is met. Additionally, it can be readily adapted to an obstacle-aware planner by including collision avoidance constraints (see the blue box in Fig. 3.1a), which do not introduce additional complexity.

The main contributions of this chapter can be summarized as follows:

- We first propose a stable pushing approach for nonholonomic mobile robots that maintains a stiff robot-object contact so that the need for frequent repositioning actions can be minimized.
- We then derive a concise linear motion constraint to simplify the stable pushing one in [39] and develop an algorithm that is easier to be implemented with commercial solvers.
- Lastly, we evaluate the proposed method through real-world experiments using wheeled mobile robots (Clearpath Husky and Boxer) that showed significant reductions in traveled distance and time.

3.2 Preliminaries

Throughout this chapter, scalars are denoted by italic lowercase letters, e.g., x, vectors by bold lowercase, e.g., $\mathbf{x}$, matrices by plain uppercase, e.g., A, and sets by calligraphic uppercase, e.g., $\mathcal{C}$. The superscript $\mathbf{x}^\top$ or $A^\top$ denotes the transpose of a vector $\mathbf{x}$ or a matrix A. Denote by $\{\mathcal{W}\}$, $\{\mathcal{R}\}$, and $\{\mathcal{O}\}$, the global world frame, the robot body frame, and the object body frame, respectively.

3.2.1 Robot dynamics model

Consider a nonholonomic differential-drive robot. Let $\mathbf{x}_{\mathrm{r\_e}} = [x_\mathrm{r}, y_\mathrm{r}, \theta_\mathrm{r}, v_\mathrm{r}, \omega_\mathrm{r}]^\top \in \mathbb{R}^5$ denote the extended robot state vector, where $\mathbf{p}_\mathrm{r} = [x_\mathrm{r}, y_\mathrm{r}]^\top$ represents the robot position (the geometric center of its four wheels) in the world frame $\{\mathcal{W}\}$, θ_r its orientation and v_r and ω_r its linear and angular velocities referring to the world frame, as shown in Fig. 3.4. Denote by $\mathbf{u}_\mathrm{r} = [a_\mathrm{r}, \xi_\mathrm{r}]^\top \in \mathbb{R}^2$ the robot's control input vector, in which a_r and ξ_r are its linear and angular accelerations, respectively. The robot dynamics are described by the following nonlinear differential equations [72]:

$$\begin{bmatrix} \dot{x}_\mathrm{r} \\ \dot{y}_\mathrm{r} \\ \dot{\theta}_\mathrm{r} \\ \dot{v}_\mathrm{r} \\ \dot{\omega}_\mathrm{r} \end{bmatrix} = \begin{bmatrix} v_\mathrm{r}\cos\theta_\mathrm{r} \\ v_\mathrm{r}\sin\theta_\mathrm{r} \\ \omega_\mathrm{r} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_\mathrm{r} \\ \xi_\mathrm{r} \end{bmatrix}, \tag{3.1}$$

which can further be written in a nonlinear discrete form $\mathbf{x}_{\mathrm{r\_e}}^{t+1} = \mathbf{f}_\mathrm{r}(\mathbf{x}_{\mathrm{r\_e}}^{t}, \mathbf{u}_\mathrm{r}^{t})$, where $t \in \mathbb{N}$ denotes the time step.

The robot velocity expressed in the robot frame is ${}^{\mathcal{R}}\mathbf{v}_\mathrm{r} = [v_\mathrm{r}, 0]^\top$. By transforming it into the world frame, we can achieve

$${}^{\mathcal{W}}\mathbf{v}_\mathrm{r} = {}^{\mathcal{W}}R_{\mathcal{R}}\,{}^{\mathcal{R}}\mathbf{v}_\mathrm{r} = \begin{bmatrix} \cos\theta_\mathrm{r} & -\sin\theta_\mathrm{r} \\ \sin\theta_\mathrm{r} & \cos\theta_\mathrm{r} \end{bmatrix} \begin{bmatrix} v_\mathrm{r} \\ 0 \end{bmatrix}, \tag{3.2}$$

where ${}^{\mathcal{W}}R_{\mathcal{R}}$ represents the rotation matrix that transforms from the robot frame, $\mathcal{R}$, to the world frame, $\mathcal{W}$.

3.2.2 Quasi-static pushing

Pushed by the mobile robot, the object slides with friction interaction with both the ground and the robot. The friction interaction is assumed to conform to Coulomb's law. A quasi-static assumption is made here that the motion of the system is slow and the wrenches are balanced with negligible inertia effects. Then, a force-motion mapping can be given according to the Limit Surface theory proposed in [73]. All the possible static and sliding friction wrenches form a convex set whose boundary is called limit surface. Under the uniform pressure distribution, the limit surface is a closed convex surface and can be approximated by an ellipsoid [74]. In this case, the applied push wrench that quasi-statically balances the friction wrench has:

$${}^{\mathcal{O}}\mathbf{w}_\mathrm{p}^\top H\,{}^{\mathcal{O}}\mathbf{w}_\mathrm{p} = 1, \tag{3.3}$$

in which $H = \mathrm{diag}(\frac{1}{(\mu_\mathrm{g} N_\mathrm{o})^2}, \frac{1}{(\mu_\mathrm{g} N_\mathrm{o})^2}, \frac{\gamma_\mathrm{g}^2}{(\mu_\mathrm{g} N_\mathrm{o})^2})$, where ${}^{\mathcal{O}}\mathbf{w}_\mathrm{p} = [{}^{\mathcal{O}}f_{\mathrm{p},x}, {}^{\mathcal{O}}f_{\mathrm{p},y}, {}^{\mathcal{O}}\tau_\mathrm{p}]^\top \in \mathbb{R}^3$ denotes the wrench applied by the pusher that quasi-statically balances the friction wrench exerted by the ground planar surface, the left super-script ${}^{\mathcal{O}}\cdot$ represents variables in the object body frame. The friction coefficient between the object and the ground planar surface is denoted

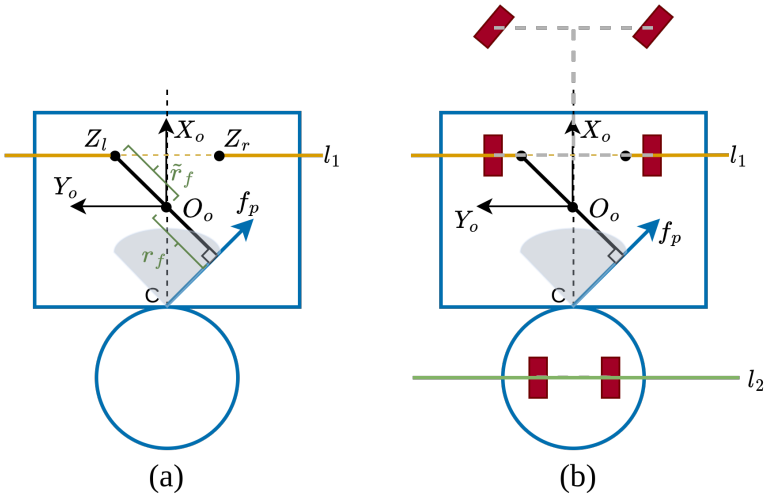


Figure 3.2: Illustration of the possible center of rotation. The circle and the rectangle represent the robot and the object in a 2D plane. The grey area and the blue arrows respectively indicate the friction cone and its edges. The orange line indicates the set of rotation centers of the object under stable push. While the green line is the robot's common left and right wheel axis and the line of its possible rotation centers. In (a), the object is pushed by an omni-directional pusher with a point contact. The set of its possible rotation centers lies on the orange line. In (b), the object is pushed by a nonholonomic robot with its rotation centers on the green line. However, there is no overlap between the possible rotation center of the wheeled robot and the object under this contact configuration. The red bricks, connected by grey dashed lines, represent the wheels of a car model in Dubin's car problem.

as μ_g, N_o represents the gravity of the object, and γ_g is an integration constant related to the contact surface area [1].

The friction wrench is a point on the limit surface when the object is sliding. Moreover, the direction of the object's twist ${}^{\mathcal{O}}\mathbf{v}_o = [{}^{\mathcal{O}}v_{o,x}, {}^{\mathcal{O}}v_{o,y}, {}^{\mathcal{O}}\omega_o]^\top \in \mathbb{R}^3$ is given by the normal to the limit surface at that point [73]. Hence, there is:

$$ {}^{\mathcal{O}}\mathbf{v}_o \propto \frac{\partial}{\partial {}^{\mathcal{O}}\mathbf{w}_p}({}^{\mathcal{O}}\mathbf{w}_p^\top H {}^{\mathcal{O}}\mathbf{w}_p) \propto H {}^{\mathcal{O}}\mathbf{w}_p. \tag{3.4} $$

3.2.3 Dubins car model with a single-point contact pusher

As concluded in [44], stable pushing with a single-point contact can be reducible to the Dubins car problem [75]. As shown in Fig. 3.2a, a round pusher pushes a rectangle shaped object at point C with a pushing force ${}^{\mathcal{O}}\mathbf{f}_p = [f_{p,x}, f_{p,y}]$, which is limited within the friction cone. The resulted twist of the object, ${}^{\mathcal{O}}\mathbf{v}_o$, can be represented as an instantaneous center of rotation $IRC = [{}^{\mathcal{O}}v_{o,x}/{}^{\mathcal{O}}\omega_o, {}^{\mathcal{O}}v_{o,y}/{}^{\mathcal{O}}\omega_o]$.

[1] $\gamma_g = \frac{A(S_g)}{\iint_{S_g} \sqrt{x^2+y^2}\,dx\,dy}$, where S_g is the contact patch between the object and the ground planar surface, and $A(S_g)$ its area.

Given a pushing force ${}^{\mathcal{O}}\mathbf{f}_{\mathrm{p}}$ at contact point C, the distance from the object frame origin O_o to the line of force is $r_{\mathrm{f}} = \frac{|{}^{\mathcal{O}}x_{\mathrm{c}} f_{\mathrm{p,x}}|}{\sqrt{f_{\mathrm{p,x}}^2 + f_{\mathrm{p,y}}^2}}$. According to the limit surface theory, distance from the center of rotation to the origin is inverse-proportional to r_{f}, that is, $\tilde{r}_{\mathrm{f}} = \sqrt{\frac{{}^{\mathcal{O}}v_{\mathrm{o},x}^2 + {}^{\mathcal{O}}v_{\mathrm{o},y}^2}{{}^{\mathcal{O}}v_{\mathrm{o},\omega}^2}} = \frac{\gamma_{\mathrm{g}}^2}{r_{\mathrm{f}}}$.

It is demonstrated in [44], as in Projective Geometry, the dual of the line of pushing force $\mathbf{f}_{\mathrm{p}}$ about the origin O_o is the instantaneous center of rotation, IRC. So the dual of $\mathbf{f}_p$ in all directions forms a line, as a set of all the possible instantaneous rotation centers, which is perpendicular to the vector from the origin to the contact point, represented as the dashed orange line, l_1, in Fig. 3.2a. But due to the friction cone constraint, the rotation center will not be positioned on the line segment $Z_l Z_r$ whose two vertices correspond to the pushing force along the edge of the friction cone.

In other words, the stable pushing constraint is translated to a bounded curvature of the object, which makes the stable pushing planning a Dubins car problem, as depicted in Fig. 3.2b. However, [44] only considers the omnidirectional pushers. If we take a differential-drive wheeled robot as the pusher, the robot can only rotate about a point that lies along its common left and right wheel axis [76], as shown in Fig. 3.2b. There comes the contradiction that the shared rotation center of the robot and the object can only be the intersection of l_1 and l_2, which means the robot and the object can only move together straightly forward or rotate around the intersection point of the two lines of rotation centers to maintain stable pushing.

3.3 Sticking contact constraint

As shown in Section.3.2.3, the maneuverability of the pushing system with a single-point contact is greatly restricted by using a nonholonomic rectangular mobile base. We focus on pushing with line contact to improve maneuverability under stable pushing. Due to the complexity of directly imposing the friction cone constraint, we instead derive a simplified linear motion constraint tailored for the differential drive robot. This approach allows us to solve the stable pushing problem effectively. The Clearpath Husky and Boxer robot are used here, as shown in Fig. 3.1. The schematic of the pusher-slider system can be found in Fig. 3.4.

3.3.1 Graphical derivation

Building upon the derivation for point contact based on the graphical approach presented in Section.3.2.3, we extend it to the line contact case, as depicted in Fig. 3.3. The line contact can be simplified as two point contacts at the extreme points [77], ${}^{\mathcal{O}}C_{\mathrm{i}} = [-W_{\mathrm{o}}/2, d_{\mathrm{i}}], i \in \{1, 2\}$. The pushing force at contact points is denoted by $\mathbf{f}_{\mathrm{p},i} = [f_{\mathrm{p},i}^{\mathrm{L}}, f_{\mathrm{p},i}^{\mathrm{R}}]^T \in \mathbb{R}^2$, including two components along the two edges of the friction cone. To ensure stiff contact between the robot and the object, the pushing forces, $\mathbf{f}_{\mathrm{p},i}$, are limited within the friction cone.

A total generalized force, $\mathbf{f}_{\mathrm{p}} = [f_{\mathrm{p}}^{\mathrm{L}}, f_{\mathrm{p}}^{\mathrm{R}}] \in \mathbb{R}^2$, and a corresponding generalized contact

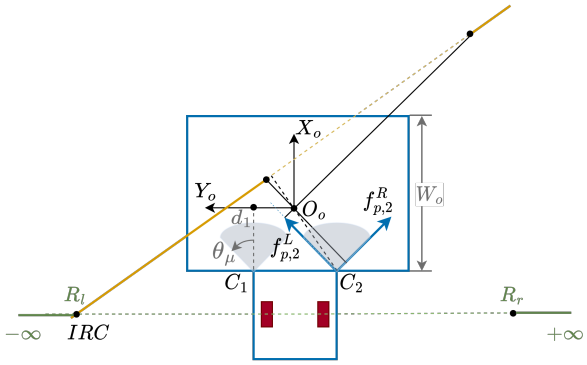


Figure 3.3: Graphical demonstration for the sticking contact constraint. The intersection of the robot's and object's possible rotation center lies on the green lines.

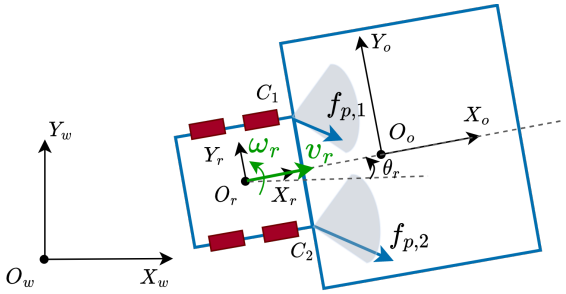


Figure 3.4: Schematic of the robot-object pushing system.

point, ${}^{\mathcal{O}}C = [-W_o/2, d], d \in [-\frac{L_o}{2}, \frac{L_o}{2}]$, can be found, which are equivalent to the two pushing forces, $\mathbf{f}_{p,i}, i \in 1, 2$, ensuring that the contact wrench exerted by the generalized force, ${}^{\mathcal{O}}\mathbf{w}_p$, matches that of the pushing forces, ${}^{\mathcal{O}}\mathbf{w}_{p,1}$ and ${}^{\mathcal{O}}\mathbf{w}_{p,2}$: ${}^{\mathcal{O}}\mathbf{w}_p = {}^{\mathcal{O}}\mathbf{w}_{p,1} + {}^{\mathcal{O}}\mathbf{w}_{p,2}$.

The generalized contact point shifts on the line segment C_1C_2, causing a tilt in the line of rotation centers l (for details, please refer to [44]). Consequently, this tilted l intersects with the wheel axis of the robot, as illustrated in Fig. 3.3. Under the friction cone constraint, all the possible intersections form the line segment $[-\infty, R_l]$ and $[R_r, +\infty]$. Obviously, the sticking constraint is transformed to a constrained motion set for the robot-object system.

3.3.2 Algebraic derivation

Now we derive the constrained motion set boundary using an algebraic approach.

The friction cone of the pushing force is

$$\mathcal{F}_{p,i} = \{\mathbf{f}_{p,i} \in \mathbb{R}^2 \mid f^L_{p,i} > 0, f^R_{p,i} > 0\}, \; i = 1, 2. \tag{3.5}$$

Equivalently, the friction cone on $\mathbf{f}_{p,i}$ can be written in a form of $\mathbf{f}_{p,i} = \lambda_{1,i}\begin{bmatrix}1\\0\end{bmatrix} + \lambda_{2,i}\begin{bmatrix}0\\1\end{bmatrix} \mid \lambda_{1,i}, \lambda_{2,i} > 0$

where $\lambda_{1,i}, \lambda_{2,i}$ are non-negative real numbers [78]. For each feasible friction force $\mathbf{f}_{\mathrm{p},i} \in \mathcal{F}_{\mathrm{p},i}$, it generates a wrench ${}^{\mathcal{O}}\mathbf{w}_{\mathrm{p},i} = J_{\mathrm{p},i}\mathbf{f}_{\mathrm{p},i}$ with $J_{\mathrm{p},i}$ the matrix that maps the contact friction force to a pusher wrench in the object's body frame.

$$J_{\mathrm{p},i} = \begin{bmatrix} \cos(\theta_\mu) & \cos(\theta_\mu) \\ \sin(\theta_\mu) & -\sin(\theta_\mu) \\ d_i\cos(\theta_\mu) + \frac{1}{2}W_{\mathrm{o}}\sin(\theta_\mu) & d_i\cos(\theta_\mu) - \frac{1}{2}W_{\mathrm{o}}\sin(\theta_\mu) \end{bmatrix} \tag{3.6}$$

3

The friction cones on the contact points lead to the wrench cone. For each friction cone $\mathcal{F}_{\mathrm{p},i}, i = 1,2$, pusher wrenches ${}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},i}$ and ${}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},i}$ corresponding to the two-unit edges $\mathbf{f}^{\mathrm{L}}_{\mathrm{p},i} = [0,1]^\top$ and $\mathbf{f}^{\mathrm{R}}_{\mathrm{p},i} = [1,0]^\top$, gives the edges of the wrench cones, as shown in Fig. 3.5b.

$${}^{\mathcal{O}}\mathcal{W}_{\mathrm{p},i} = \{{}^{\mathcal{O}}\mathbf{w}_{\mathrm{p},i} = J_{\mathrm{p},i}\mathbf{f}_{\mathrm{p},i} \mid \mathbf{f}_{\mathrm{p},i} \in \mathcal{F}_{\mathrm{p},i}\},\ i = 1,2. \tag{3.7}$$

where

$$\begin{aligned} {}^{\mathcal{O}}\mathbf{w}_{\mathrm{p},i} &= \lambda_{1,i}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},i} + \lambda_{2,i}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},i} \\ &= \begin{bmatrix} \lambda_{1,i}\cos(\theta_\mu) + \lambda_{2,i}\cos(\theta_\mu), \\ \lambda_{1,i}\sin(\theta_\mu) - \lambda_{2,i}\sin(\theta_\mu), \\ \lambda_{1,i}(d_i\cos(\theta_\mu) + \frac{1}{2}W_{\mathrm{o}}\sin(\theta_\mu)) + \lambda_{2,i}(d_i\cos(\theta_\mu) - \frac{1}{2}W_{\mathrm{o}}\sin(\theta_\mu)) \end{bmatrix}. \end{aligned} \tag{3.8}$$

Then the generalized wrench of the two pushing forces is

$$\begin{aligned} {}^{\mathcal{O}}\mathbf{w}_{\mathrm{p}} &= \lambda_3{}^{\mathcal{O}}\mathbf{w}_{\mathrm{p},1} + \lambda_4{}^{\mathcal{O}}\mathbf{w}_{\mathrm{p},2} \\ &= \lambda_3(\lambda_{1,1}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},1} + \lambda_{2,1}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},1}) \\ &\quad + \lambda_4(\lambda_{1,2}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},2} + \lambda_{2,2}{}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},2}) \end{aligned} \tag{3.9}$$

where $\lambda_j > 0,\ j = 3,4$. Since $\lambda_{1,i}\lambda_j > 0$, the feasible set of the generalized wrench in Eq. (3.9) can be represented as a convex hull ${}^{\mathcal{O}}\mathcal{W}_{\mathrm{p}}$, as shown in Fig. 3.5c.

$${}^{\mathcal{O}}\mathcal{W}_{\mathrm{p}} = \mathbf{cvx\_hull}({}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},1}, {}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},1}, {}^{\mathcal{O}}\mathbf{w}^{\mathrm{L}}_{\mathrm{p},2}, {}^{\mathcal{O}}\mathbf{w}^{\mathrm{R}}_{\mathrm{p},2}) \tag{3.10}$$

As mentioned in Eq. (3.4), the limit surface theory gives the mapping of the pushing force and the resulting object sliding motion. The direction of the object's twist is parallel to $H{}^{\mathcal{O}}\mathbf{w}_{\mathrm{p}}$. Combining with Eq. (3.7), we can write all possible twists ${}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}} = [{}^{\mathcal{O}}v_{\mathrm{o},x}, {}^{\mathcal{O}}v_{\mathrm{o},y}, {}^{\mathcal{O}}\omega_{\mathrm{o}}]^\top$ of the object as:

$${}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}} = \{k_{\mathrm{o}}H{}^{\mathcal{O}}\mathbf{w}_{\mathrm{p}} \mid {}^{\mathcal{O}}\mathbf{w}_{\mathrm{p}} \in {}^{\mathcal{O}}\mathcal{W}_{\mathrm{p}},\ k_{\mathrm{o}} \in \mathbb{R}^+\}, \tag{3.11}$$

where k_{o} is a magnitude parameter.

For all pusher wrenches ${}^{\mathcal{O}}\mathbf{w}_{\mathrm{p}} \in {}^{\mathcal{O}}\mathcal{W}_{\mathrm{p}}$ that are on the ellipsoidal limit surface, the set of mapped object twists ${}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}}$ is also a polyhedral cone since the mapping in Eq. (3.11) is linear. Thus, we can compute the motion cone ${}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}}$ by computing its edges, as shown in Fig. 3.5d.

Additionally, since the object is pushed by the robot, which has a linear velocity v_{r} and angular velocity ω_{r}, without losing or sliding the contact, we have the object velocity

$${}^{\mathcal{W}}\mathbf{v}_{\mathrm{o}} = {}^{\mathcal{W}}\mathbf{v}_{\mathrm{r}} + {}^{\mathcal{W}}R_{\mathcal{R}} \cdot (\boldsymbol{\omega}_{\mathbf{r}} \times {}^{\mathcal{R}}\mathbf{x}_{\mathrm{o}})_{(1:2)} \tag{3.12}$$

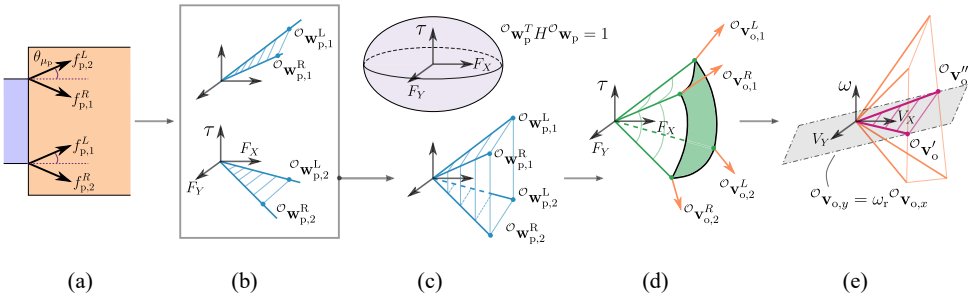


Figure 3.5: Illustration of the motion cone construction for planar pushing using a nonholonomic robot. (a) Friction cones. (b) Individual generalized friction cones. (c) Convex hull of the individual generalized friction cones (blue region) and the limit surface (light purple ellipsoid). (d) Feasible pusher wrenches (on the green surface) and force-motion model (orange vectors). (e) Motion cone of the object (area marked red).

where ${}^{\mathcal{R}}\mathbf{x}_{\mathrm{o}} = [d_{\mathrm{ro}}, {}^{\mathcal{R}}y_{\mathrm{o}}, 0]^{\top}$ denotes the object position in the robot frame and $\boldsymbol{\omega}_{\mathbf{r}} = [0, 0, \omega_{\mathrm{r}}]^{\top}$ corresponds to the pure rotation velocity vector of the robot. The subscript (1:2) indicates taking the first two dimensions of the vector.

After substituting Eq. (3.2) in Eq. (3.12), the velocity of the object expressed in the object frame can be achieved by multiplying ${}^{\mathcal{W}}R_{\mathcal{R}}^{-1}$ at both sides of Eq. (3.12), which yields:

$$ {}^{\mathcal{O}}v_{\mathrm{o},x} = v_{\mathrm{r}} - \omega_{\mathrm{r}}\,{}^{\mathcal{R}}y_{\mathrm{o}}, \quad {}^{\mathcal{O}}v_{\mathrm{o},y} = \omega_{\mathrm{r}} d_{\mathrm{ro}}, \quad {}^{\mathcal{O}}\omega_{\mathrm{o}} = \omega_{\mathrm{r}}. \tag{3.13} $$

It can be observed that Eq. (3.13) describes a plane ${}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}}$ crossing the origin in the $\mathcal{O}_{x-y-\omega}$ space: ${}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}} = \{{}^{\mathcal{O}}\mathbf{v}_{\mathrm{o}} \mid {}^{\mathcal{O}}v_{\mathrm{o},y} - d_{\mathrm{ro}}\,{}^{\mathcal{O}}\omega_{\mathrm{o}} = 0\}$, as shown in Fig. 3.5e. Combining Eq. (3.11)-(3.13), we can obtain the final possible twists of the object, known as the object motion cone, as the intersection of the set ${}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}}$ and the plane ${}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}}$: ${}^{\mathcal{O}}\bar{\mathcal{V}}_{\mathrm{o}} = \{{}^{\mathcal{O}}\mathbf{v}_{\mathrm{o}} \mid {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o}} \in {}^{\mathcal{O}}\mathcal{V}_{\mathrm{o}},\ {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o}} \in {}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}}\}$. The edges of the motion cone are computed as the intersection between the planes ${}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},1}^{R} - {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},2}^{R}$, ${}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},1}^{L} - {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},2}^{L}$ and the plane ${}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}}$, which results in two edge vectors, ${}^{\mathcal{O}}\mathbf{v}'_{\mathrm{o}}$ and ${}^{\mathcal{O}}\mathbf{v}''_{\mathrm{o}}$.

$$ \begin{aligned} {}^{\mathcal{O}}\mathbf{v}'_{\mathrm{o}} &= ({}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},1}^{L} \times {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},2}^{L}) \times \vec{\mathbf{n}} = k_o \begin{bmatrix} -d_{\mathrm{ro}}\cos(\theta_\mu) \\ -d_{\mathrm{ro}}\sin(\theta_\mu) \\ -\sin(\theta_\mu) \end{bmatrix} \\ {}^{\mathcal{O}}\mathbf{v}''_{\mathrm{o}} &= ({}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},1}^{R} \times {}^{\mathcal{O}}\mathbf{v}_{\mathrm{o},2}^{R}) \times \vec{\mathbf{n}} = k_o \begin{bmatrix} -d_{\mathrm{ro}}\cos(\theta_\mu) \\ d_{\mathrm{ro}}\sin(\theta_\mu) \\ \sin(\theta_\mu) \end{bmatrix} \end{aligned} \tag{3.14} $$

where $\vec{\mathbf{n}} = [0, 1, -d_{\mathrm{ro}}]$ is the normal vector to plane ${}^{\mathcal{O}}\mathcal{P}_{\mathrm{o}}$.

The object motion cone can then be written as ${}^{\mathcal{O}}\bar{\mathcal{V}}_{\mathrm{o}} = \lambda_5\,{}^{\mathcal{O}}\mathbf{v}'_{\mathrm{o}} + \lambda_6\,{}^{\mathcal{O}}\mathbf{v}''_{\mathrm{o}} \mid \lambda_5, \lambda_6 \in \mathbb{R}_{\geq 0}$. According to Eq. (3.13), we can achieve the corresponding motion cone for the robot, $\mathcal{V}_{\mathrm{r}}$, with a linear mapping $\begin{bmatrix} v_{\mathrm{r}} \\ w_{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & {}^{\mathcal{R}}y_{\mathrm{o}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{\mathrm{o,x}} \\ v_{\mathrm{o,y}} \\ w_{\mathrm{o}} \end{bmatrix} \mid \begin{bmatrix} v_{\mathrm{o,x}} \\ v_{\mathrm{o,y}} \\ w_{\mathrm{o}} \end{bmatrix} \in {}^{\mathcal{O}}\bar{\mathcal{V}}_{\mathrm{o}}$. Expressing the robot

motion cone as a conical combination,

$$\begin{aligned}\begin{bmatrix} v_\mathrm{r} \\ w_\mathrm{r} \end{bmatrix} =& \lambda_5 \begin{bmatrix} -{}^{\mathcal{R}}y_\mathrm{o}\sin(\theta_\mu) - d_\mathrm{ro}\cos(\theta_\mu) \\ -\sin(\theta_\mu) \end{bmatrix} + \\ & \lambda_6 \begin{bmatrix} {}^{\mathcal{R}}y_\mathrm{o}\sin(\theta_\mu) - d_\mathrm{ro}\cos(\theta_\mu) \\ \sin(\theta_\mu) \end{bmatrix}\end{aligned} \tag{3.15}$$

from which we achieve the motion constraint on the robot input by finding the boundary of $w_\mathrm{r}/v_\mathrm{r}$

$$k'' v_\mathrm{r}^t \leq \omega_\mathrm{r}^t \leq k' v_\mathrm{r}^t \tag{3.16}$$

where $v_\mathrm{r} \geq 0$, $k'' = \frac{\sin(\theta_\mu)}{{}^{\mathcal{R}}y_\mathrm{o}\sin(\theta_\mu) - d_\mathrm{ro}\cos(\theta_\mu)}$, $k' = \frac{\sin(\theta_\mu)}{{}^{\mathcal{R}}y_\mathrm{o}\sin(\theta_\mu) + d_\mathrm{ro}\cos(\theta_\mu)}$. It can also be regarded as a constraint on the curvature of the robot's trajectory, k. For simplification, we only plan for the pushes at the middle of the contact surface, where ${}^{\mathcal{O}}y_\mathrm{o} = 0$.

3.4 Planning for robot pushing

With the motion constraint derived in Eq. (3.16), we now present a motion planner for robot pushing that keeps the object to be within its motion cone based on NMPC.

NMPC formulation

We formulate a receding horizon optimization problem with N time steps and planning horizon $N\Delta t$:

$$\min_{\mathbf{x}_\mathrm{r\_e}^{1:N}, \mathbf{u}_\mathrm{r}^{0:N-1}} \quad \sum_{t=0}^{N-1} J^t(\mathbf{x}_\mathrm{r\_e}^t, \mathbf{u}_\mathrm{r}^t) + J^N(\mathbf{x}_\mathrm{r}^N) \tag{3.17a}$$

$$\text{s.t.} \quad \mathbf{x}_\mathrm{r\_e}^0 = \mathbf{x}_\mathrm{r\_e}(t_0), \tag{3.17b}$$

$$\mathbf{x}_\mathrm{r\_e}^t = \mathbf{f}_\mathrm{r}(\mathbf{x}_\mathrm{r\_e}^{t-1}, \mathbf{u}_\mathrm{r}^{t-1}), \tag{3.17c}$$

$$\mathbf{h}_\mathrm{pushing}(\mathbf{x}_\mathrm{r\_e}^t) \leq 0, \tag{3.17d}$$

$$\mathbf{h}_\mathrm{avoidance}(\mathbf{x}_\mathrm{r\_e}^t) \leq 0, \tag{3.17e}$$

$$\mathbf{u}_\mathrm{r}^{t-1} \in \mathcal{U}_\mathrm{r},\ \forall t \in \{1, \ldots, N\},$$

where Δt is the sampling time, J^t denotes the cost term at stage t and J^N denotes the terminal cost, $\mathbf{x}_\mathrm{r\_e}(t_0)$ is the initial extended state of the robot, $\mathbf{f}_\mathrm{r}$ is the robot dynamics model, $\mathcal{U}_\mathrm{r}$ represents the robot's acceleration and angular acceleration limits. $\mathbf{h}_\mathrm{pushing}$ and $\mathbf{h}_\mathrm{avoidance}$ respectively represent the path constraints for stable pushing and obstacle avoidance, which will be described in detail in the following.

Cost functions

Let $\mathbf{p}_\mathrm{o}^\mathrm{g}$ be the goal location that the object needs to be pushed to. We minimize the displacement between the object's terminal position with this goal. To this end, the terminal cost is defined as: $J^N(\mathbf{x}_\mathrm{r\_e}^N) = q_\mathrm{goal} \left\| \mathbf{p}_\mathrm{o}^N - \mathbf{p}_\mathrm{o}^\mathrm{g} \right\|$, where the object's terminal position is $\mathbf{p}_\mathrm{o}^N = \mathbf{p}_\mathrm{r}^N + R(\theta_\mathrm{r}^N)[d_\mathrm{ro}, 0]^\mathrm{T}$ with $R(\cdot)$ the two-dimensional rotation matrix. q_goal is a tuning

weight. The stage cost is to minimize the robot's linear and angular velocities to render it not to move too fast:$J^t(\mathbf{x}_{\mathrm{r\_e}}^t, u_{\mathrm{r}}^t) = q_{\mathrm{v}}(v_{\mathrm{r}}^t)^2 + q_{\omega}(\omega_{\mathrm{r}}^t)^2$, where q_{v} and q_{θ} are tuning weights.

Pushing constraints

To make the robot keep contact with the object while pushing, the object's motion has to be within its motion cone at each time step. By combining the computed motion cone in Eq. (3.16) with the continuous pushing constraint, the sticking contact constraints can be derived as follows:

$$v_{\mathrm{r}}^t \geq 0, \quad k'' v_{\mathrm{r}}^t \leq \omega_{\mathrm{r}}^t \leq k' v_{\mathrm{r}}^t, \tag{3.18}$$

It indicates that the robot has to push forward the object, but its angular velocity should be within a motion cone related to the forward speed, which formulates the stable pushing constraints $\mathbf{h}_{\mathrm{pushing}}$.

3.5 Experimental results

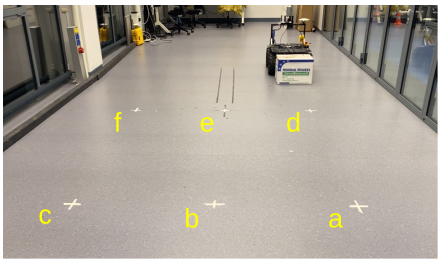
To validate the efficacy of our proposed method, we performed experiments using two robots, Clearpath Husky and Boxer, to test the stable pushing performance (Fig. 3.6 and 3.10). Both the Husky and Boxer robots were differential-drive wheeled robots with rectangular shapes, respectively sized 0.97×0.67 m and 0.75×0.55 m. Our experimental results demonstrated a 100% stiff contact when applying the proposed concise stable pushing constraint. Additionally, we compared the proposed method with state-of-the-art pushing baselines to showcase the conciseness of our proposed constraint and the efficiency of stable pushing by effectively controlling object motion.

3.5.1 Real-world experiments using Husky and boxer

We carried out real-world experiments with two robots to demonstrate the efficacy of our proposed sticking contact constraint when stably pushing paper boxes. Our experiments utilized a motion capture system (OptiTrack) and a Kalman filter to collect information on robots, objects, and obstacles that operate at 120Hz. Control commands were calculated using our NMPC-based method on a laptop and sent to robots through WiFi and ROS, which operate at a frequency of 20Hz. We use the open source solver ACADOS [40] to solve the NMPC problem, with a sampling time of $\Delta t = 0.1$ seconds, a planning horizon of $N = 20$ and tuning weights $q_{\mathrm{goal}} = 1, q_{\mathrm{v}} = q_{\omega} = 0.1$.

The Husky robot was equipped with a line bumper in the front, which acts as a pushing effector. It was used to push a large paper box measuring $0.32 \times 0.48 \times 0.48$ meters and weighing 2.8 kilograms. At the beginning of the push, the box was placed in contact with the robot center at a distance of $d_{\mathrm{ro}} = 0.66$ meters. The angle of the friction cone was set to $\theta_{\mu} = 12.00$ degrees. It is estimated by measuring the force which could pull the box at a constant speed, such that the pulling force is equal to the friction force: $F_{\mathrm{pull}} = F_f = \tan\theta_{\mu} \cdot m_{\mathrm{o}} g$. Then θ_{μ} can be achieved as $\arctan(\frac{F_{\mathrm{pull}}}{m_{\mathrm{o}} g})$.

Using the above setup, the limits of the robot trajectory curvature are calculated as



(a)

(b)

(c)

Figure 3.6: Evaluation of stable pushing performance with different goal positions. (a) Six predefined pushing goals, represented by white crosses on the floor. The corresponding goal-oriented pushing results are shown in Fig. 3.7 (a–f). (b) and (c) show the experimental results of robot pushing without and with the sticking contact constraints (Eq.(3.18) *), respectively. The transparency of the robot and box indicates their movement over time.*

$k' = 0.32$ and $k'' = -0.32$. Due to the size limitation of the motion capture system, we selected six pushing goals with coordinates (2,1), (2,0), (2,-1), (0,1), (0,0), and (0,-1) to evaluate the stable pushing performance, as shown in Fig. 3.6a. Starting from the initial position (-2,1), the Husky robot was tasked with pushing the paper box to the designated goal positions, as shown in Fig. 3.7 (a-f). The robot successfully maintained sticking contact with the object in all cases. Compared to trajectories without the stiff contact constraint (Fig. 3.7 (h-j)), the object easily slides away while the robot moves (intuitive comparison can be found in Fig. 3.6b and 3.6c). However, the contact constraint also limited the maneuverability of the pushing system, so that the maximum curvature of the planned trajectory was bounded. Fig. 3.8 illustrates the relationship between maneuverability and motion cone. As a result, some pushing targets (e.g., Goal c in Fig. 3.7) were unattainable within a limited time with the local NMPC planner. Reposition actions are required, so a global pushing planner will be the focus of our future research. Additionally, the proposed method can be easily extended to an obstacle-aware case, as shown in Fig. 3.1 and Fig. 3.9. A static obstacle is placed in front of the robot, and the object's goal location is behind it. The robot can successfully avoid the obstacle by maintaining both the stiff contact and obstacle avoidance constraints while pushing the object to the goal location.

Furthermore, we aimed to comprehensively validate the effectiveness of our proposed stable pushing method under varying friction conditions using the Boxer robot within a distinct environment. A series of experiments were conducted to this end. In the initial phase, we conducted ablation studies to assess the effectiveness of the sticking contact constraint with box sized 0.39×0.59 m. Three pushing targets were selected, with five pushing trials conducted for each target. The outcomes of these ablation experiments are illustrated in Fig. 3.10, demonstrating an impressive 100% success rate across all trials. Subsequently, we tried a new box sized 0.32×0.48 m and proceeded to an experiment where the robot pushed an object around the room. The implementation of the stiff contact constraint ensured that the robot maintained stiff contact with the object throughout the process. This strategic approach significantly reduced the need for frequent repositioning actions and requires only two designed switches. To further gauge the stability and robustness of our method, we designed a path tracking experiment. In this setup, the robot meticulously followed a predefined path while engaging in stable pushing. Both sets of experimental results are depicted in Fig. 3.11 (shown in the attached video as well), illustrating the method's consistent performance across diverse scenarios.

Overall, the outcomes of these comprehensive experiments demonstrate the robustness and efficacy of our proposed method across different friction conditions and robot platforms, underscoring its potential for real-world applications in robotics.

3.5.2 Comparison with the baseline approaches

What's more, to assess the performance of our proposed stable pushing method, we compared it to two existing baseline approaches, namely the reactive pushing strategy [25] and a Linear Time-Varying Model Predictive Control (LTV MPC) based stable pushing approach [39]. The comparison results are presented in Table 3.1.

During the pushing process, the reactive pushing strategy attempts to minimize the

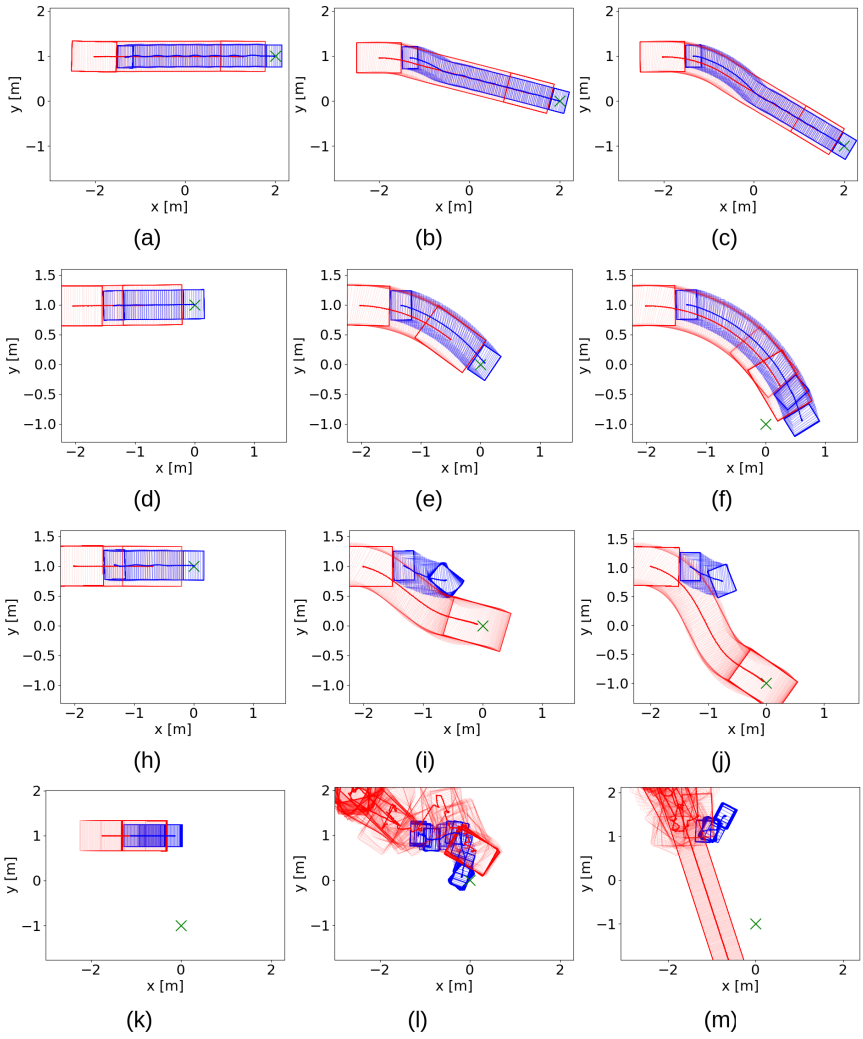


Figure 3.7: Experiments were conducted to evaluate the stable pushing performance with different goal positions, comparing the results with and without the sticking contact constraint. Additionally, the proposed approach is compared to a baseline method. Panels (a-f) show the stable pushing outcomes for the six selected goals, as depicted in Fig. 3.6a. For goals (d), (e), and (f), Fig. (h-j) displays the pushing paths without the sticking contact constraint, while Fig. (k-m) highlights the performance of the reactive pushing strategy.

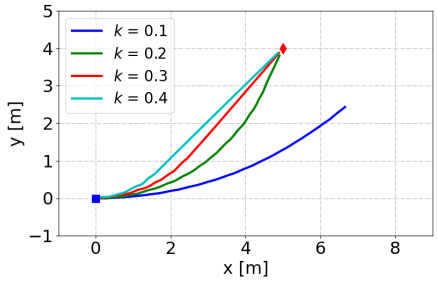


Figure 3.8: Robot trajectories for pushing considering various limits of the robot trajectory curvature, where $k = k' = -k''$. The blue square and the red diamond represent the start and the goal locations, respectively. The smaller the motion cone, the maneuverability of the robot is more limited.

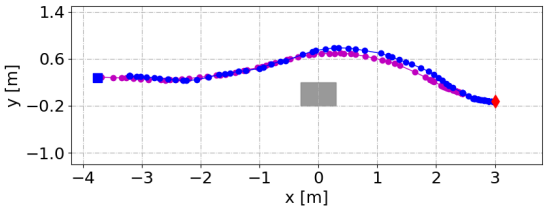
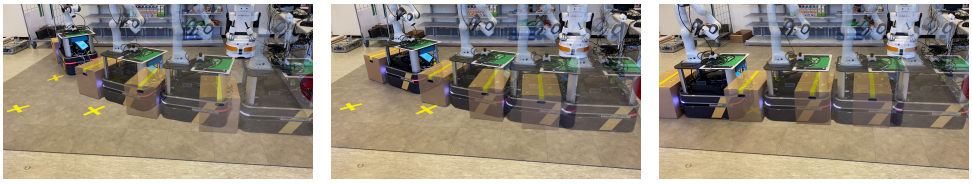
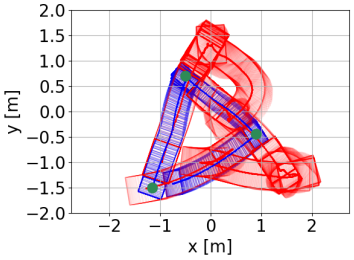


Figure 3.9: Experimental results of obstacle-aware robot pushing. The red and blue curves with dots represent the trajectories of the robot and the pushed object, respectively. The obstacle is marked in gray.

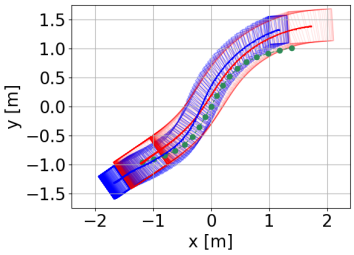


(a) (b) (c)

Figure 3.10: Goal-targeted stable pushing with Boxer. Stiff contact is successfully maintained under the sticking contact constraint.



(a) Push an object around the room.



(b) Path following while engaging in stable pushing.

Figure 3.11: Stable pushing across different scenarios. The red and blue curves represent the trajectories of the robot and the pushed object, respectively. The reference waypoints are marked in green. In (b), A sponge sheet is sticked to the box to augment friction in the robot-object interaction, where $k' = -k'' = 0.4$. For detailed information, we direct readers to our accompanying video.

	Number of hyper-parameters	Success rate (For Goal 1, 2, 3)	Decision variables in MPC (at time t)	Solvable with commercial solver
Proposed approach	1	100%, 100%, 0%	7	Yes
Reactive pushing	5	100%, 60%, 0%	-	-
LTV MPC	5	-	11	No

Table 3.1: Comparison to the baselines

angle between the object's movement direction and its direction toward the goal location. As a result, the robot must maneuver around the object to adjust its angle and sometimes reposition itself when the robot-object contact is lost. However, the core of the controller is a Proportional-Integral-Derivative (PID) controller, which is challenging to tune for optimal performance. Due to safety concerns, we tested this approach only in simulation. As shown in Fig. 3.7 (k-m), the robot often loses contact with the object, requiring time-consuming repositioning actions. Moreover, since the approach was originally designed for omnidirectional robots, it does not account for the motion constraints of nonholonomic robots. The robot sometimes bumps into the object while repositioning, adversely affecting pushing performance. In contrast, our proposed approach has demonstrated superior efficiency and pushing success rate for all three goals while maintaining a higher pushing success rate. The reactive pushing approach only achieves high success rates when the goal position is directly in front of the robot and is close to the initial position. To achieve

the goals d, e, f, it has an average distance traveled by the robot and a time of 8.53 m and 58.4 s, respectively, while our proposed approach only takes 6.53 m and 13.2 s which saves 23.8% and 77.4% in these metrics.

The LTV MPC-based pushing method shares the same motivation and mechanics as our proposed approach which is to add the friction cone constraint to guarantee stable pushing. However, the LTV MPC approach directly adds the stiff contact constraint to the optimization problem without any preprocessing. Consequently, it has four additional independent decision variables and four more hyperparameters to tune in the MPC formulation. We utilized the open source ACADOS solver to solve the MPC problem proposed in LTV MPC, which is unsolvable due to extra independent variables. Compared to other models, our concise stiff contact constraint requires only one hyperparameter ($k' = -k''$) to tune and can be easily added to MPC-based navigation controllers.

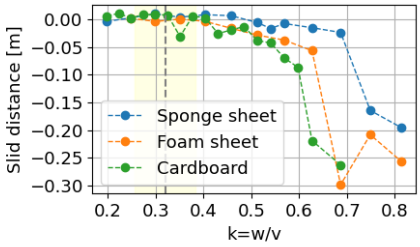
3.5.3 Sensitivity analysis

Recognizing the inherent challenges in accurately measuring friction coefficients, we conducted a comprehensive sensitivity analysis. The primary goal was to determine the parameter k without prior knowledge of the friction coefficient between the robot and the object. Additionally, we sought to comprehend how variations in the estimation of k would impact the effectiveness of stable pushing. Subsequently, we assessed stable pushing performance for objects with distinct surface characteristics, including sponge sheet, foam sheet, and cardboard. Furthermore, recognizing the common occurrence of non-uniform mass distribution in unwieldy objects, we conducted experiments involving the rearrangement of the same set of objects within the box, thus achieving diverse mass distributions. This enabled us to investigate the method's robustness in scenarios where the assumption of uniform mass distribution is not perfectly upheld.

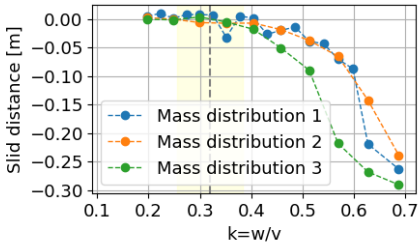
Because k represents the limit of the robot trajectory curvature, our experimental setup entailed pushing various objects at a uniform speed of 0.1 m/s around a predetermined rotation center for a duration of 4 seconds. This rotation center, in turn, determines moving along a certain trajectory with curvature $k = w_r/v_r$. By measuring the displacement of the object's position in the robot frame at both the start and end of the trajectory, we quantified the cumulative slid distance of the object at different k. The outcomes of the experiments are depicted in Fig. 3.12. Notably, when $k < k' = 0.32$ (for ${}^{\mathcal{O}}y_o = 0$, where changing direction represents a symmetry case that we omit here), the object's slid distance remains at zero such that stable pushing is attainable. Conversely, when $k > k' = 0.32$, the assurance of stable pushing diminishes where the object slides. This observed trend persists across all tested friction conditions and mass distributions, underscoring the approach's capacity for generalization. Even when k deviates by as much as ±20%, the slid distance remains constrained to within 0.05 m.

3.5.4 Discussion

The proposed approach introduces a simple analytical stable pushing constraint, ensuring pushing stability under the line contact between the robot and the object. It is well-suited



(a) Slid distance for pushes along different curvatures, k, across different friction conditions which indicates the effectiveness of stable pushing.



(b) Slid distance for pushes along different curvatures, k, across different mass distributions which indicates the effectiveness of stable pushing.

Figure 3.12: Validate the effectiveness of stable pushing under different conditions. The grey dashed line corresponds to $k = 0.32$*. The yellow shading represents the range of* $k \in [0.8k, 1.2k]$*. Notably, the results illustrate that stable pushing—where the sliding distance is less than 0.05 m—can be realized whenever* $k < 0.32$*, irrespective of alterations in friction conditions or mass distribution. Additionally, even when the hyperparameter in the stiff pushing constraint deviates by up to ±20%, stable pushing remains intact.*

for objects with uniform mass distributions, and it can potentially be extended to handle cases with slightly nonuniform mass distributions and indeterminate anisotropic friction. Its simplicity is a notable feature, with only one hyperparameter requiring approximation. However, stable pushing imposes limitations on maximum trajectory curvature, which is decided by the friction condition between the robot-object interaction. Adding high friction coating will help to improve system maneuverability.

In contrast, there are widely-used learning-based pushing controllers utilize data-based pushing dynamics models, which do not consider the shape or mass distribution of the object [11, 20, 26]. However, data-driven methods are known for their data dependency, challenges in generalization, and susceptibility to Model Drift. Moreover, they neglect pushing stability, resulting in frequent object sliding and the need for time-consuming repositioning actions, especially problematic for nonholonomic mobile robots with limited maneuverability.

The choice between stable pushing for regular-shaped objects and intermittent pushing for complex objects should be made based on the specific application's requirements and the characteristics of the objects involved.

3.6 Conclusion

This chapter addresses the problem of using a differential-drive mobile robot to push an object to a goal location. We start by revisiting the pushing mechanics and highlighting the nonholonomic robot's challenges. To overcome the challenge, we propose a stable pushing approach that maintains a stiff line contact between the robot and the object, controlled by a stable pushing constraint. As a key contribution of this work, we provide an algorithm to simplify this constraint as a concise motion constraint for the robot. An NMPC-based planner is presented for stable pushing by considering the motion constraint. Our proposed method is more efficient than reactive pushing strategies, with a 23. 8% reduction in the traveled trajectory length and a 77.4% reduction in time. Furthermore, our method is more concise than the LTV MPC-based stable pushing method, making it easier to implement. We validate our proposed method through real-world experiments with Husky and Boxer robots under different friction conditions. However, the stable pushing method has limitations in maneuverability. Our future research aims to design global policies that can further switch between contact surfaces to improve maneuverability.

4

Unwieldy object delivery with nonholonomic mobile base: A free pushing approach

Parts of this chapter appeared in:

- **Y. Tang**, M. Wisse, W. Pan, "Unwieldy Object Delivery With Nonholonomic Mobile Base: A free Pushing Approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8991-8998, Sep. 2024.
- **Y. Tang**, M. Wisse, "Pushing manipulation with a Nonholonomic Mobile Base" *ECCOMAS Thematic Conference on Multibody Dynamics*, July 24 - 28, 2023, Lisbon, Portugal.

In Chapter 3, we address the challenge of unwieldy object delivery using a stable pushing approach, which focuses on maintaining stiff contact between the robot and the object, thereby reducing reliance on robust state estimation. However, the motion constraints inherent in stable pushing limit the system's maneuverability. To overcome this limitation, we propose a free pushing approach in this chapter. This method allows the robot to slide relative to the object, enabling it to maneuver around the object while pushing. Although this approach requires more accurate object state estimation, it offers more flexible pushing motions and better adaptability to confined spaces.

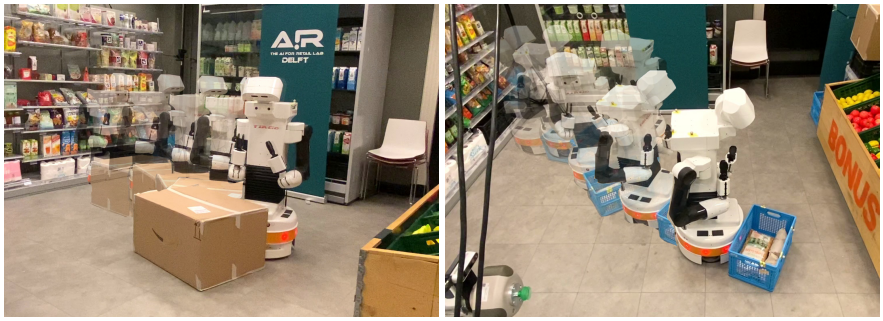
4.1 Introduction

Mobile manipulators hold great potential for practical applications, such as logistics operations, where they may need to move objects that are unwieldy, either too heavy or too large, for their manipulator arm(s) to grasp. A viable solution is to equip the robot with the ability to push the object, an essential motion primitive for handling objects of varying sizes. In this chapter, we concentrate on pushing using the basic mobile base [25, 39, 79], in contrast to recent studies that focus on pushing with a manipulator's end effector [80].

In the field of pushing with robot arms, *stable pushing* approach is widely used. It assumes a stiff contact while pushing [19, 39, 79]. However, for this assumption to be valid, the contact forces imposed by the robot on the object must remain within the friction cone, causing nonholonomic constraint on the motion of the pushed object [44, 81]. As a result, the mobility of the pushed object is limited. Especially when pushed with mobile base pushers, which are mostly nonholonomic wheeled robots, stable pushing would further undermine the maneuverability of the robot-object system. In the case of a cylinder-shaped nonholonomic robot, stable pushing only allows it to push the object straight forward or rotate with a fixed turning radius [82].

To improve pushing agility, pushing planners which allow for relative sliding between the robot and object are proposed [11–14]. In these methods, the robot-object contact is switched between sticking, right-slide, and left-slide modes to provide the required pushing force on the object. But these pushing motions are planned under the assumption that the robot can freely access any point on the object, which may be true for a robot arm but not necessarily for a nonholonomic robot. For example, differential-drive robots can not move sideways to reach the planned contact point. Executing the planned pushes is always difficult on a real robot platform due to the limitations of the robot's kinematics [11].

This chapter aims to develop a maneuverable push approach that allows the transportation of objects using powerful mobile base pushers while considering the feasibility of the plans. The proposed approach allows the object to slide relative to the pusher, hence called *free pushing*, in contrast to the *stable pushing* methods. To ensure the feasibility of the planned pushes, planning is conducted for the motion of the robot pusher and the resulting object movement. It is achieved through the development of a robot-object contact model by modeling the robot and the object as a unified multibody system employing Differential Algebraic Equations (DAE) such that the motion of the object relative to the robot can be predicted. Unlike existing pushing models, the relative motion is modeled with careful



(a) (b)

Figure 4.1: Pushing (a) a $0.565 \times 0.755 \times 0.425$ m^3 *sized package, and (b) an 8 kg weighed basket, with a differential-drive mobile base. Tiago, a mobile service robot from PAL Robotics [83], is used in this work. Transparency in the image indicates the movements.*

consideration of the robot's shape and kinematics. To the best of our knowledge, this is the first paper that addresses the shape and kinematics of the robot in pushing modeling and planning. A Model Predictive Controller (MPC) solves the Push Planning Problem in real time.

The contributions of our paper can be summarized as follows:

- We propose a free pushing approach for nonholonomic mobile robot which enables the robot to maneuver around the object while pushing it, allowing for improved pushing maneuverability compared to the stable pushing counterpart.
- We develop a robot-object contact model that takes into account the robot shape and kinematics in pushing modeling and planning, ensuring efficient continuous pushing and the feasibility of the planned pushes.
- We evaluate the proposed method through real-world experiments using Tiago (Fig. 4.1), a mobile service robot, demonstrating an average success rate of 83% with an accuracy of 0.085m.

4.2 Preliminaries

We consider a pushing planning problem where a cylinder-shaped differential-drive robot pushes a rectangular object toward a goal position. Fig. 4.2 provides a visualization of the problem setup. The pushing system consists of the cylinder-shaped robot, which can be controlled directly, and the rectangular object, which moves in response to the contact forces. We assume that the contact between the robot and the object is frictionless while using a continuous friction model for the contact between the object and the ground. In the following, we will introduce the friction-less contact model and continuous friction model respectively. Before going into details, we first provide the definitions for the variables used throughout the paper.

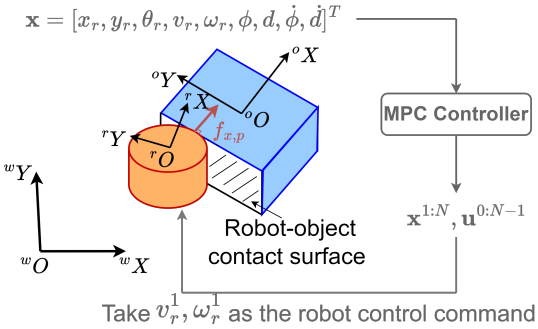


Figure 4.2: The configuration of the coordinates and frames, as well as the demonstration of the control framework.

- $\mathbf{x}_\mathrm{r}$, the robot state $[x_\mathrm{r}, y_\mathrm{r}, \theta_\mathrm{r}]^\mathrm{T}$ in the world frame, where the origin of the robot frame is defined as the geometric center of the cylinder-shaped robot base;
- $\mathbf{x}_\mathrm{o}$, the object state $[x_\mathrm{o}, y_\mathrm{o}, \theta_\mathrm{o}]^\mathrm{T}$ in the world frame, where the origin of the object frame is defined as the geometric center of the cuboid basket;
- ϕ, the angle between the object and the robot frame as shown in Fig. 4.3a
- d, the y coordinate of robot center in the object frame;
- v_r, ω_r, the linear and angular velocities of the robot;
- a_r, ξ_r, the linear and angular accelerations of the robot;
- ${}^{\mathcal{O}}f_{\mathrm{x,p}}$, the x-component of the push force in the object frame[1];
- $\mathbf{w}_\mathrm{g} = [f_{\mathrm{x,g}}, f_{\mathrm{y,g}}, \tau_\mathrm{g}]^\mathrm{T}$ is the friction wrench exerted by the ground on the object, where $f_{\mathrm{x,g}}, f_{\mathrm{y,g}}$ are the x and y component of the friction force, τ_g is the friction torch;
- $\tilde{\mathbf{w}}_\mathrm{g} = [\tilde{f}_{\mathrm{x,g}}, \tilde{f}_{\mathrm{y,g}}, \tilde{\tau}_\mathrm{g}]^\mathrm{T}$, the simplified friction wrench exerted by the ground to the object;
- $M_\mathrm{o} = \mathrm{diag}[m_\mathrm{o}, m_\mathrm{o}, I_\mathrm{o}]$, the inertia matrix of the object, where $m_\mathrm{o}, I_\mathrm{o}$ denote the mass and moment of inertia of the object;
- $W_\mathrm{o}, L_\mathrm{o}$, width and length of the object;
- r_r, radius of the robot.

4.2.1 Friction-less robot-object contact model

With a point contact between the robot and the object, the object can rotate around that contact point or slide along the contact surface, which can be seen in Fig. 4.3a.

[1]The superscript $\mathcal{R}$ denotes the robot frame. The object and the world frames are represented as $\mathcal{O}$ and $\mathcal{W}$. $\mathcal{W}$ is always omitted for simplicity.

Instead of dividing the contact modes into $\{sticking, right-sliding, left-sliding, separation\}$ states [12, 84, 85], we use a frictionless contact assumption to simplify the problem such that there is a continuous sliding contact. Since friction forces only play a minor role in our pushing tasks, this assumption helps reduce computational time at the expense of a slight loss in physical accuracy [41].

In addition, we introduce contact constraints to ensure continuous contact between the robot and the object, avoiding separation contact modes during the pushing process. Specifically, these constraints require that (4.1a) the robot does not move backward, (4.1b) the robot does not decelerate abruptly at a rate greater than the object's deceleration due to friction with the ground, (4.1c) the object remains in front of the robot, and (4.1d) the robot maintains contact with the object at the same plane. The contact constraints are summarized as:

$$v_{\mathrm{r}} \geq 0 \tag{4.1a}$$

$$|a_{\mathrm{r}}| < a_{\mathrm{r,\,max}} \tag{4.1b}$$

$$-90^{\circ} < \phi < 90^{\circ} \tag{4.1c}$$

$$-\frac{1}{2}L_o < d < \frac{1}{2}L_o \tag{4.1d}$$

where $a_{\mathrm{r,\,max}}$ is the maximum acceleration of the robot.

4.2.2 Continuous object-ground contact friction model

While the contact between the robot and the object is assumed to be frictionless in our pushing planning problem, we model the contact between the object and the floor using a continuous friction model. In most previous studies on pushing planning, the Coulomb friction model has been used, and the friction cone has been employed to control the transition between different contact modes. However, the Coulomb model is discontinuous at zero relative tangential contact velocities, which can lead to instability in the numerical simulation of the sliding motion [86]. This characteristic makes the design of controllers more complicated.

Thus we point out the key modeling decision in this chapter which is to use a continuous friction model instead of the traditional Coulomb friction model [87]. This choice enables us to model the contact dynamics in a continuous manner, which streamlines the controller design process. To obtain the continuous friction model, we mathematically approximate the Coulomb model using a Sigmoid function. It should be noted that we assume a constant coefficient of friction, μ_{g}, for the friction interaction between the object and the ground at the contact surface. The friction force at position ${}^{\mathcal{W}}\mathrm{p}$ is expressed as

$$\mathbf{f}_{\mathrm{g,p}}({}^{\mathcal{W}}\mathbf{p}) = -\mu_{\mathrm{g}} \cdot (\gamma_1 \mathrm{sig}(\mathbf{v}_{\mathrm{p}}) - [\gamma_2, \gamma_2]^{\mathrm{T}}) \cdot f_{\mathrm{n}} \tag{4.2}$$

where $\gamma_1 = 2, \gamma_2 = 1$ are scaling factors, f_{n} is the normal force at ${}^{\mathcal{W}}\mathrm{p}$, $\mathbf{v}_{\mathrm{p}} = [\dot{x}_{\mathrm{p}}, \dot{y}_{\mathrm{p}}]^{\mathrm{T}}$ the velocity of ${}^{\mathcal{W}}\mathrm{p}$ expressed in the world frame, and $\mathrm{sig}(\cdot)$ is the sigmoid function that operates element-wise on the input vector, such that $\mathrm{sig}(\mathbf{x})$ yields a vector $\mathbf{y}$, where each element y_i is computed as $y_i = \frac{1}{1+\exp(-(x_i))}$, for $i = 1, 2, ..., n$, where n is the length of the input vector $\mathbf{x}$.

4.3 Modelling

In this Section, we present the dynamics model for the pushing system. First, we introduce the model for the differential-drive robot pusher. Then we will explain the contact transition dynamics between the robot and the object such that the physical interaction is modeled with a 2D (i.e., top-view) constrained multi-body system.

4.3.1 Robot dynamics model

Given the full robot state $[x_\mathrm{r}, y_\mathrm{r}, \theta_\mathrm{r}, v_\mathrm{r}, \omega_\mathrm{r}]^\top$ and its control input $\mathbf{u}_\mathrm{r} = [a_\mathrm{r}, \xi_\mathrm{r}]^\top \in \mathbb{R}^2$, the dynamics model of robot can be written as:

$$\begin{bmatrix} \dot{x}_\mathrm{r} \\ \dot{y}_\mathrm{r} \\ \dot{\theta}_\mathrm{r} \\ \dot{v}_\mathrm{r} \\ \dot{\omega}_\mathrm{r} \end{bmatrix} = \begin{bmatrix} v_\mathrm{r}\cos(\theta_\mathrm{r}) \\ v_\mathrm{r}\sin(\theta_\mathrm{r}) \\ \omega_\mathrm{r} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_\mathrm{r} \\ \xi_\mathrm{r} \end{bmatrix} \tag{4.3}$$

4.3.2 Dynamics model of the pushing system

Different from the widely-used point contact model [12, 84, 85] where the robot is a pole and its shape is ignored, we model the relative motion between the robot and the object with careful consideration of the shape of the robot. Since we constrain that the contact between the robot and the object is carefully maintained, the robot-object system is modeled as two rigid bodies connected by an idealized joint. Furthermore, we assume that the mobile robot is controlled in a closed loop to achieve the desired movement, so its motion will not be influenced by the reaction force exerted by the object.

For the pushed object, its motion is a result of the contact interaction with the robot, as shown in Fig. 4.3. The dynamics of the object can be achieved by a combination of the Newton-Euler equations of motion and a contact constraint equation in the form of Differential Algebraic Equations.

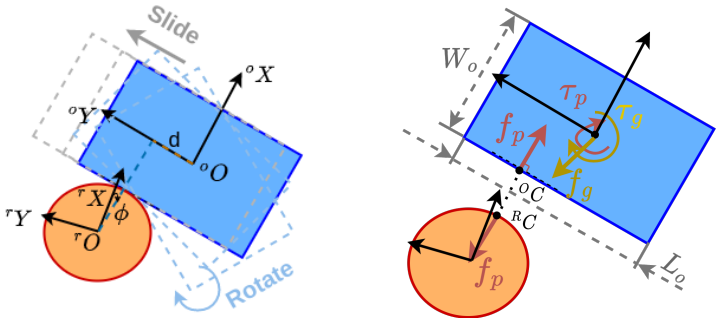
The Newton Euler equations of motion for the object are

$$\begin{cases} \cos(\theta_o)\,{}^{\mathcal{O}}f_{\mathrm{x,p}} + f_{\mathrm{x,g}} = m_\mathrm{o}\ddot{x}_\mathrm{o} \\ \sin(\theta_o)\,{}^{\mathcal{O}}f_{\mathrm{x,p}} + f_{\mathrm{y,g}} = m_\mathrm{o}\ddot{y}_\mathrm{o} \\ -d\,{}^{\mathcal{O}}f_{\mathrm{x,p}} + \tau_\mathrm{g} = I_\mathrm{o}\ddot{\theta}_\mathrm{o} \end{cases} \tag{4.4}$$

where $d = -\sin(\theta_o)\cdot(x_r - x_o) + \cos(\theta_o)\cdot(y_r - y_o)$ is the position of the contact point in y direction of the Object frame.

To solve for the three unknown accelerations of the object as well as the push force in Eq. (4.4), we need one more equation. Since the successive interaction restricts the separate movement of the two bodies at the contact point, we have the contact constraint:

$${}^{\mathcal{O}}x_\mathrm{r} = \cos(\theta_\mathrm{o})(x_\mathrm{r} - x_\mathrm{o}) + \sin(\theta_\mathrm{o})(y_\mathrm{r} - y_\mathrm{o}) = -r_\mathrm{r} - \frac{1}{2}W_\mathrm{o} \tag{4.5}$$



(a) (b)

Figure 4.3: The robot and the object are assumed to be connected with a virtual sliding joint around the contour of the mobile base. (a) shows the possible robot-object relative motion which not only includes sliding but also rotating around the contact point. In (b), the robot and the object are taken as two free bodies where the contact joint between them is cut. Then the Newton-Euler equations of motion of the object can be achieved where the joint is "glued" with the constraint equation.

By differentiating Eq. (4.5) twice with respect to time, we achieve the constraint on the object accelerations:

$$C_{\mathbf{x}}\ddot{\mathbf{x}}_{\mathrm{r}} + g_{\mathbf{x}} = 0 \tag{4.6}$$

with $g_{\mathbf{x}} = -\cos(\theta_{\mathrm{o}})\dot{\theta}_{\mathrm{o}}\dot{\theta}_{\mathrm{o}}(x_{\mathrm{r}}-x_{\mathrm{o}}) + \cos(\theta_{\mathrm{o}})\ddot{x}_{\mathrm{r}} + \sin(\theta_{\mathrm{o}})\ddot{y}_{\mathrm{r}} - 2\sin(\theta_{\mathrm{o}})\dot{\theta}_{\mathrm{o}}(\dot{x}_{\mathrm{r}}-\dot{x}_{\mathrm{o}}) - \sin(\theta_{\mathrm{o}})\dot{\theta}_{\mathrm{o}}\dot{\theta}_{\mathrm{o}}(y_{\mathrm{r}} - y_{\mathrm{o}}) + 2\cos(\theta_{\mathrm{o}})\dot{\theta}_{\mathrm{o}}(\dot{y}_{\mathrm{r}} - \dot{y}_{\mathrm{o}})$, and $C_{\mathbf{x}} = [-\cos(\theta_{\mathrm{o}}), -\sin(\theta_{\mathrm{o}}), -\sin(\theta_{\mathrm{o}})(x_{\mathrm{r}} - x_{\mathrm{o}}) + \cos(\theta_{\mathrm{o}})(y_{\mathrm{r}} - y_{\mathrm{o}})]$.

Combining the equations of motion (Eq. (4.4)) and the contact constraint equation (Eq. (4.6)) leads to the full set of DAEs:

$$\begin{bmatrix} M_{\mathrm{o}} & C_{\mathbf{x}}^{\mathrm{T}} \\ C_{\mathbf{x}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_{\mathrm{o}} \\ {}^{\mathcal{O}}f_{\mathrm{x,p}} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_{\mathrm{g}} \\ g_{\mathbf{x}} \end{bmatrix} \tag{4.7}$$

where $\mathbf{w}_{\mathrm{g}}$ is calculated by integrating the force over the contact patch A:

$$\mathbf{f}_{\mathrm{g}} = \int_{R} -\mu_{\mathrm{g}}(2\mathrm{sig}(\mathbf{v}_{\mathrm{p}}) - [1,1]^{\mathrm{T}})p({}^{\mathcal{O}}\mathbf{p})dA, \tag{4.8}$$

$p({}^{\mathcal{O}}\mathbf{p})dA$ is the normal force at ${}^{\mathcal{O}}\mathbf{p}$ under pressure distribution $p(\cdot)$, ${}^{\mathcal{O}}\mathbf{p}$ is the position expressed in the object frame. The corresponding frictional moment is $\tau_{\mathrm{g}} = [{}^{\mathcal{O}}\mathbf{p} \otimes \mathbf{f}_{\mathrm{g}}]_{(3)}$, with $\otimes$ cross product, $[\cdot]_{(3)}$ indicates the third element in the vector.

4.3.3 Model simplification

However, the precision of the double integration depends on the number of segmented contact patches. Integration with more contact patches promises a more accurate approximation to the integral but slow computation. For computational efficiency, we simplified

the friction model of the object-ground contact by only summing for the force at the four corners of the polygonal object, instead of integrating over the whole contact region. Assuming a uniform object density when projected onto the horizontal plane, the normal force at four corners is $\frac{1}{4}m_o g$, where g refers to the acceleration of gravity. Using Eq. (4.2), the simplified friction force is ${}^{\mathcal{W}}\tilde{\mathbf{f}}_{\mathrm{g}} = \sum {}^{\mathcal{W}}\mathbf{f}_{\mathrm{g,\,i}}({}^{\mathcal{O}}\mathbf{p}_{\mathrm{c,i}})$ where $i = \{0,1,2,3\}$ for a rectangle object, ${}^{\mathcal{O}}\mathbf{p}_{\mathrm{c,i}}$ is the coordinate of the corner at contact patch A. The corresponding simplified frictional moment is $\tilde{\tau}_{\mathrm{g}} = \sum[{}^{\mathcal{O}}\mathbf{p}_{\mathrm{c,i}} \otimes {}^{\mathcal{W}}\tilde{\mathbf{f}}_{\mathrm{g,\,i}}]_{(3)}$.

4.4 Planning for robot pushing

By predicting the motion of the pushed object using Eq. (4.7), we can solve the constrained pushing planning problem using a Model Predictive Controller. It ensures the feasibility of the MPC-planned pushes by planning for both the path of the robot and the motion of the pushed object. First, we will discuss how to reformulate the DAE model to simplify the optimization problem in MPC. Afterward, we will present the Optimal Control Problem (OCP) and the cost function.

4.4.1 Dynamics model in generalized coordinates with implicit constraints

To solve the pushing control problem in an MPC formulation, a system dynamic model is required. The robot-pusher system is modelled as a constrained multi-body system whose dynamics is derived with a set of equations of motion and an algebraic constraint equation in Eq. (4.7). However, it is known that many numerical discretization schemes, such as the implicit Runge–Kutta (IRK) methods, fail to converge or exhibit an order reduction when applied to DAEs [88]. The algebraic constraints for the system state make the forward prediction problem significantly hard.

Removing the algebraic constraint in Eq. (4.6), obviously can help by making the problem less complex. Thus we reformulate the DAEs to simplify the optimal control problem. By choosing a set of independent generalized coordinates, we derive the system dynamics where the algebraic contact constraint can be imposed implicitly.

The object only has two degrees of freedom with respect to the robot, as shown in Fig. 4.3a. Therefore, the independent generalized coordinates are chosen as $\mathbf{q} = [\phi, d]$. With the contact constraint between the robot and the object during pushing, the coordinates of the object in the robot frame can be written as a function of the generalized coordinates and their derivatives:

$$\begin{bmatrix} {}^{\mathcal{R}}x_{\mathrm{o}} \\ {}^{\mathcal{R}}y_{\mathrm{o}} \\ {}^{\mathcal{R}}\theta_{\mathrm{o}} \end{bmatrix} = \begin{bmatrix} \cos(\phi)(W_{\mathrm{o}}/2 + r_{\mathrm{r}}) + d\sin(\phi) \\ \sin(\phi)(W_{\mathrm{o}}/2 + r_{\mathrm{r}}) - d\cos(\phi) \\ \phi \end{bmatrix} \tag{4.9}$$

We differentiate Eq. (4.9) twice:

$$^{\mathcal{R}}\dot{\mathbf{x}}_{\mathrm{o}} = T\dot{\mathbf{q}} \tag{4.10a}$$

$$^{\mathcal{R}}\ddot{\mathbf{x}}_{\mathrm{o}} = T\ddot{\mathbf{q}} + \mathbf{g} \tag{4.10b}$$

where

$$^{\mathcal{R}}\ddot{\mathbf{x}}_{\mathrm{o}} = [^{\mathcal{R}}\ddot{x}_{\mathrm{o}}, {}^{\mathcal{R}}\ddot{y}_{\mathrm{o}}, {}^{\mathcal{R}}\ddot{\theta}_{\mathrm{o}}]^{\mathrm{T}}$$

$$\mathbf{g} = \begin{bmatrix} -\cos(\phi)\dot{\phi}^2(w_{\mathrm{o}}/2 + r_{\mathrm{r}}) + \dot{d}\cos(\phi)\dot{\phi} - d\sin(\phi)\dot{\phi}^2 + \dot{d}\cos(\phi)\dot{\phi}, \\ -\sin(\phi)\dot{\phi}^2(w_{\mathrm{o}}/2 + r_{\mathrm{r}}) + \dot{d}\sin(\phi)\dot{\phi} + d\cos(\phi)\dot{\phi}^2 + \dot{d}\sin(\phi)\dot{\phi}, \\ 0 \end{bmatrix}$$

$$T = \begin{bmatrix} -\sin(\phi)(w_{\mathrm{o}}/2 + r_{\mathrm{r}}) + d\cos(\phi) & \sin(\phi) \\ \sin(\phi)(w_{\mathrm{o}}/2 + r_{\mathrm{r}}) + d\sin(\phi) & -\cos(\phi) \\ 1 & 0 \end{bmatrix}$$

$^{\mathcal{R}}\ddot{\mathbf{x}}_{\mathrm{o}}$ can be achieved by reformulating the dynamics in Eq. (4.4) in the non-inertia reference frame (the Robot frame):

$$M_{\mathrm{o}}{}^{\mathcal{R}}\ddot{\mathbf{x}}_{\mathrm{o}} = {}^{\mathcal{R}}\mathbf{w}_{\mathrm{p}} + {}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}} \tag{4.11}$$

where $^{\mathcal{R}}\mathbf{w}_{\mathrm{p}} \in \mathbb{R}^3$ denotes the push wrench exerted by the robot, $^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}} = 2m_{\mathrm{o}}\dot{\boldsymbol{\theta}}_{\mathrm{r}} \times {}^{\mathcal{W}}\dot{\mathbf{p}}_{\mathrm{r}} + m_{\mathrm{o}}{}^{\mathcal{W}}\ddot{\mathbf{p}}_{\mathrm{r}} + m_{\mathrm{o}}\dot{\boldsymbol{\theta}}_{\mathrm{r}} \times (\dot{\boldsymbol{\theta}}_{\mathrm{r}} \times {}^{\mathcal{W}}\mathbf{p}_{\mathrm{r}}) + I_{\mathrm{o}}\ddot{\boldsymbol{\theta}}_{\mathrm{r}} \times {}^{\mathcal{W}}\mathbf{p}_{\mathrm{r}}$ is the introduced additional fictitious wrenches with $\dot{\boldsymbol{\theta}}_{\mathrm{r}} = [0, 0, \omega_{\mathrm{r}}]$, ${}^{\mathcal{W}}\mathbf{p}_{\mathrm{r}} = [x_{\mathrm{r}}, y_{\mathrm{r}}, 0]$.

To achieve the object dynamics with generalized coordinates, the TMT method proposed in [89], is used with the application of the principle of virtual power. Based on the DAE achieved in Section.IV, the TMT method simply derives the unconstrained equations of motion [90].

Since we have introduced the generalized coordinates, we also have a new set of corresponding generalized force $\mathbf{f}_q$. Given the wrench of generalized force as $\mathbf{w}_q$, the total virtual power of $^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}}$, $^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}}$ and $\mathbf{w}_q$ is

$$\delta P = \delta^{\mathcal{R}}\dot{\mathbf{x}}_{\mathrm{o}}^{\mathrm{T}}({}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - M_{\mathrm{o}}{}^{\mathcal{R}}\ddot{\mathbf{x}}_{\mathrm{o}} - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}}) + \delta\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{w}_q \tag{4.12}$$

The virtual velocities and the accelerations which satisfy the constraints are shown in Eq. (4.10a)-(4.10b). Substitution in Eq. (4.12) yields the virtual power expressed in generalized coordinates and their derivatives,

$$\delta P = (T\delta\dot{\mathbf{q}})^{\mathrm{T}}({}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - M_{\mathrm{o}}(T\ddot{\mathbf{q}} + \mathbf{g}) - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}}) + \delta\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{w}_q \tag{4.13}$$

Because the system is in dynamic equilibrium where $\delta P = 0$, which results in

$$T^{\mathrm{T}}({}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - M_{\mathrm{o}} T\ddot{\mathbf{q}} - M_{\mathrm{o}}\mathbf{g} - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}}) + \mathbf{w}_q = 0 \tag{4.14}$$

$\mathbf{f}_q$ equals the friction force between robot and object, which is omitted because of the friction-less assumption in Section. 4.2.1. Then we have $\mathbf{w}_q = 0$. Rearranging the terms in Eq. (4.14) gives us contact dynamics in terms of generalized coordinates.

$$\ddot{\mathbf{q}} = T^{\mathrm{T}} M_{\mathrm{o}}({}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}} - M_{\mathrm{o}}\mathbf{g}) \tag{4.15}$$

Hence, the dynamics of the whole robot-object system is the combination of Eq. (4.3) and Eq. (4.15).

$$\begin{bmatrix} \dot{x}_{\mathrm{r}} \\ \dot{y}_{\mathrm{r}} \\ \dot{\theta}_{\mathrm{r}} \\ \dot{v}_{\mathrm{r}} \\ \dot{\omega}_{\mathrm{r}} \\ \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{r}}\cos(\theta_{\mathrm{r}}) \\ v_{\mathrm{r}}\sin(\theta_{\mathrm{r}}) \\ \omega_{\mathrm{r}} \\ a_{\mathrm{r}} \\ \xi_{\mathrm{r}} \\ \dot{\mathbf{q}} \\ T^{\mathrm{T}} M_{\mathrm{o}}({}^{\mathcal{R}}\tilde{\mathbf{w}}_{\mathrm{g}} - {}^{\mathcal{R}}\mathbf{w}_{\mathrm{fic}} - M_{\mathrm{o}}\mathbf{g}) \end{bmatrix} \tag{4.16}$$

4.4.2 Optimal control problem (OCP) formulation

With the simplified motion model, a desired controller must be able to recover from the perturbation and be fast enough to do re-planning online. Additionally, it is important to obey the contact constraint in Eq. (4.1) so that the derived contact dynamics is applied. To satisfy these requirements, we formulate an OCP where the goal is to minimize the finite-horizon cost-to-go function subject to the constraints and the dynamics of the system at every control cycle.

We first define the system state vector $\mathbf{x}$ and control input $\mathbf{u}$ as $\mathbf{x} = [x_{\mathrm{r}}, y_{\mathrm{r}}, \theta_{\mathrm{r}}, v_{\mathrm{r}}, \omega_{\mathrm{r}}, \phi, d, \dot{\phi}, \dot{d}]^{\mathrm{T}}$ and $\mathbf{u} = [a_{\mathrm{r}}, \xi_{\mathrm{r}}]^{\mathrm{T}}$. Then the cost-to-go for N time steps is set as:

$$J^{\mathrm{N}}(\mathbf{x}^{\mathrm{N}}) = ({}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{\mathrm{N}} - {}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{G}) Q ({}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{\mathrm{N}} - {}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{G})^{\mathrm{T}} \tag{4.17}$$

where Q denotes the weight matrix associated with the object state, ${}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{\mathrm{N}} = [{}^{\mathcal{W}}x_{\mathrm{o}}^{\mathrm{N}}, {}^{\mathcal{W}}y_{\mathrm{o}}^{\mathrm{N}}, {}^{\mathcal{W}}\theta_{\mathrm{o}}^{\mathrm{N}}]^{\mathrm{T}}$ is the predicted object pose in the world frame, ${}^{\mathcal{W}}\mathbf{p}_{\mathrm{o}}^{G}$ is the target object pose.

The final OCP is defined to minimize the distance between the object pose and the target pose at the end of the overall horizon, while satisfying the contact constraints in

Eq. (4.1), the state constraints and the robot dynamics:

$$\min_{\mathbf{x}^{1:N},\mathbf{u}^{0:N-1}} \quad J^{\mathrm{N}}(\mathbf{x}^{\mathrm{N}}) \tag{4.18a}$$

$$\text{s.t.} \quad \mathbf{x}^0 = \mathbf{x}^{t_0} \tag{4.18b}$$

$$\mathbf{x}^t = \mathbf{f}(\mathbf{x}^{t-1}, \mathbf{u}_{\mathrm{r}}^{t-1}) \tag{4.18c}$$

$$v_{\mathrm{r}}^t > 0 \tag{4.18d}$$

$$-90^\circ < \phi^t < 90^\circ \tag{4.18e}$$

$$|a_{\mathrm{r}}^{t-1}| < a_{\mathrm{r,max}} \tag{4.18f}$$

$$\mathbf{u}^{t-1} \in \mathcal{U}_{\mathrm{r}},\ \forall t \in \{1,\dots,N\}$$

It should be noted that $\mathbf{x}^t = \mathbf{f}(\mathbf{x}^{t-1}, \mathbf{u}_{\mathrm{r}}^{t-1})$ is the forward model in Eq. (4.16), and $\mathcal{U}_{\mathrm{r}}$ represents the robot's acceleration and angular acceleration limits. Every time step, the OCP is solved online.

4.5 Experimental results

This Section presents an evaluation and validation of the proposed pushing model and controller. The Tiago service robot [83] (see Fig. 4.1), which has a cylinder-shaped mobile base with a radius of 27cm, was used to push several rectangular boxes with varying physical properties, including mass, size, and texture. The results indicate that our simplified pushing model predicts the object's motion under pushing with an accuracy better than 0.075m. With this prediction, the proposed controller plans feasible trajectories by adapting the contact with the object while pushing to the goal. Although the pushing success rate varies depending on the target's position, the proposed method can achieve an average success rate of 83% with a pushing accuracy of 0.085m for the selected goals, distributed equally in the test room. Compared to the stable pushing controller [82], the proposed approach improves the agility and efficiency of mobile manipulators and is robust in achieving the task while tolerating modeling errors.

4.5.1 Evaluation of the contact model

We evaluate the precision of the contact dynamics model. The Tiago robot is controlled to push a paper box that has a dimension of $0.386 \times 0.585 \times 0.4$ m^3 and a weight of 1.5 kg. The friction coefficient of the ground-box contact is $\mu_{\mathrm{g}} = 0.4$. Two primary sources of prediction errors were identified: the mismatch between the actual shape of the robot and its modeled counterpart and the slippage of the wheeled robot during movement.

We first evaluate the accuracy of the proposed contact model under different contact configurations. To exclude the influence of the robot slippage, we conduct object pushing experiments with a fixed speed as $v_{\mathrm{r}} = 0.1$m/s, $w_{\mathrm{r}} = 0$ rad/s. 380 trails of the pushing trajectory are collected for 20 timestamps, with a sampling time of $\delta t = 0.1s$. The average error of the motion prediction by the DAE model (Eq. 4.7) and the simplified Generalized Coordinate Model (GCM, Eq. 4.16) is presented in Fig. 4.5a and 4.5c, respectively. We found that the overall prediction error was under 0.025m for the DAE model and 0.04m for the

simplified GCM. However, we observed that the prediction accuracy was lower in cases where the object was in front of the robot at the center ($d = 0, \phi = 0$) due to a mismatch between the real robot and our model. Specifically, there is a boss on the front of the mobile base serving as the charging connector, which causes discontinuous contact dynamics, as shown in Fig. 4.4.

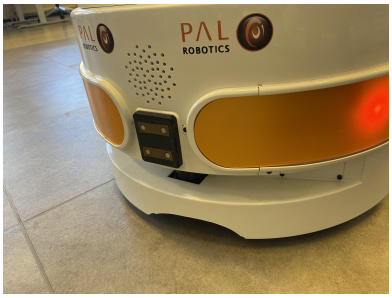


Figure 4.4: Boss on the robot.

Furthermore, we evaluate the prediction error under different pushing speed by conducting at least 5 trials of pushing trajectory for each speed configuration, with increments of 0.1m/s and 0.05rad/s for linear and rotation speed, respectively. The results are shown in Fig. 4.5b and 4.5d. It demonstrates an increasing trend in prediction error as pushing speed increases. This could be attributed to the mobile robot's wheel slippage worsening at higher speeds, leading to poorer object motion prediction during pushing. Under all the speed configurations, the DAE model had the highest prediction error of 0.06m, while the simplified GCM had a prediction error of 0.075m.

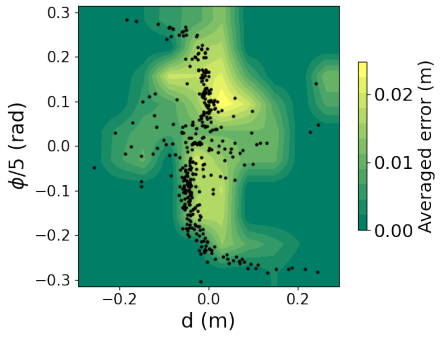
Despite the simplified model's inherent lower prediction accuracy compared to the DAE model, the error resulting from the different contact/speed configurations seems more significant, as shown in Fig. 4.5. Consequently, a reactive controller is deemed necessary to bridge the gap in the modeling process and tolerate modeling errors.

4.5.2 Validation of the controller

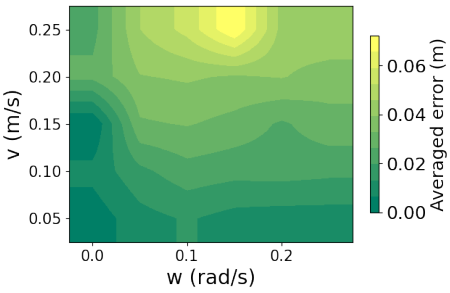
Simulation results

Using the same setup as in Section 4.5.1, we conducted simulations in which the robot pushed the object to different goal positions with varying initial contact configurations. The MPC planner is configured with a prediction horizon of 30 and a sampling time of $\delta t = 0.1s$. As a goal-reaching controller, the weight matrix is set as $Q = \text{diag}(1, 1, 0)$. The solver we used is IPOPT. The key parameters for both the simulation experiments and real-world experiments are concluded in Table. 4.1.

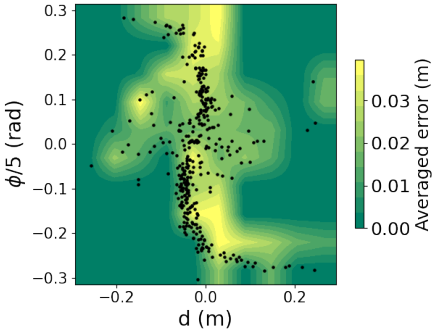
Three different scenarios were defined, and the proposed free pushing approach is compared with the stable pushing method in [44]. As the simulation results shown in Fig. 4.6, the robot changes its contact with the object during the pushing process in the free



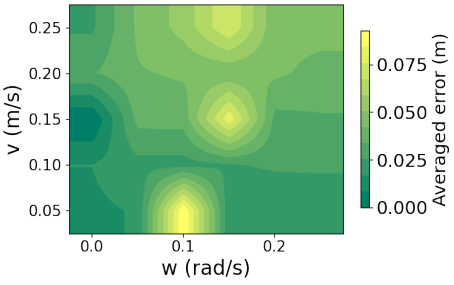
(a) Prediction accuracy of DAE under different contact configurations



(b) Prediction accuracy of DAE at different pushing speeds



(c) Prediction accuracy of GCM under different contact configurations



(d) Prediction accuracy of GCM at different pushing speeds

Figure 4.5: The prediction accuracy of the DAE model and the simplified GCM evaluated by their averaged prediction error. The black dots in (a), (b) represent the sampled contact configurations. The prediction accuracy is determined by comparing the predicted object position with its actual position recorded by the motion capture system (OptiTrack) after a prediction horizon of 20 timestamps.

pushing mode, while it maintains a sticking contact with the object in the stable pushing cases. To maintain the sticking contact in stable pushing, the object is modeled as a Dubin's car with its maximum curvature limited [44], as shown in Fig. 4.6b. However, it does not consider the kinematic constraints of the nonholonomic robot, which results in infeasible push plans, as seen in Fig. 4.6d, where the robot is unable to follow the trajectory to push the object sideways. Furthermore, in cases where the robot and the pushing target are initially positioned on the left of the object, the robot is capable of pushing the object towards the goal by adapting the contact configuration during pushing, as depicted in Fig. 4.7. But it is not possible with the stable pushing controller where a longer pushing trajectory with a large curvature is required. In conclusion, the free pushing method plans for feasible trajectories for the robot by considering its nonholonomic constraint. It is also more agile and efficient in completing pushing tasks due to its ability to change contact points while pushing.

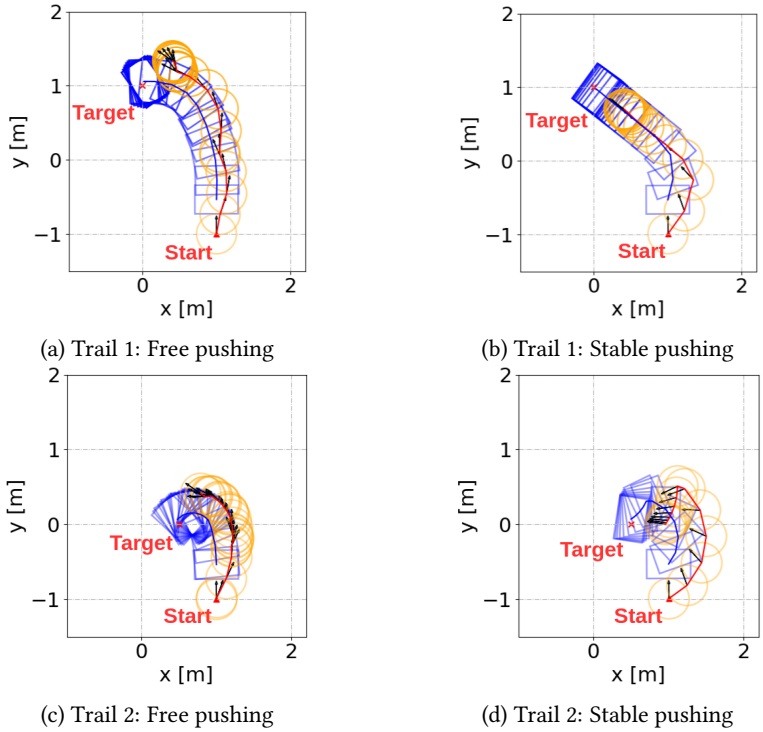
Real-world experimental results

The proposed controller is tested in physical experiments using (1) the same paper box in Section. 4.5.1 and (2) a plastic basket which sizes $0.345 \times 0.522 \times 0.278$ m^3 and weighs 8 kg. The robot and object states are provided by a motion capture system (OptiTrack) and a Kalman filter, which runs at 120Hz. Control commands are computed on a laptop using our proposed MPC-based method and are sent to the Tiago robot via WiFi and ROS, which runs at 10Hz. The average solution time of the MPC controller is 60ms. According to the setup of Tiago, we send the optimized v_r^1 and ω_r^1 form Eq. 4.18 as its control commands (as demonstrated in Fig. 4.2).

Ablation experiments are conducted to push the box towards 6 different targets (as shown in Fig. 4.8), distributed evenly in the motion capture room, for 10 times per target. We defined pushing accuracy as the square root of the average of the squared errors between the goal and the actual final position of the pushed object. We considered cases with pushing accuracy below 0.25m as failures. The results are summarized in Table 4.2, where we observed an average reaching accuracy below 0.120m and a delivery success rate higher than 70%. Since the object movement is very sensitive to the small differences in the

Table 4.1: Key parameter values used in both the simulation and real-world experiments.

	Parameter	**Value**
MPC	Prediction Horizon	30
	Sampling Time	0.1s
	Weight, Q, in cost (Eq. 4.17)	diag(1,1, 0).
	Solver	IPOPT
	Average Solution Time	60ms
Pushed objects 1. Paper Box	Size	$0.386 \times 0.585 \times 0.4m^3$
	Mass	1.5 kg
Pushed objects 2. Plastic Basket	Size	$0.345 \times 0.522 \times 0.278m^3$
	Mass	8 kg



(a) Trail 1: Free pushing

(b) Trail 1: Stable pushing

(c) Trail 2: Free pushing

(d) Trail 2: Stable pushing

Figure 4.6: Comparison of the pushing performance with a free and stable pushing controller in simulation. The orange circles and blue rectangles represent the trajectory of the robot and object, respectively. The black arrows indicate the direction of the robot. Trails (a), and (b) share the same target of (0, 1), while panels (c) and (d) share the target of (0.5, 0). All panels start with the same initial contact configuration of $d = 0$ and $\phi = 0$.

contacts and initial states, the robot reacts to unforeseen changes by continuously adapting its contact with the object, resulting in differences in travel distance, even for the same goal. In Fig. 4.9, we present several repeated pushing trials targeting goal 4. Starting at position $(0, -1)$ with the contact configuration $d = 0, \phi = 0$, the resulting object trajectories varied significantly. Furthermore, as the object got closer to the target, there were instances where the robot's control inputs were too small to articulate the pushing, leading to wheel slippage and failure, as shown in Fig. 4.9c. This observation motivates our future research on compliant robot pushing.

As we already discussed in our previous work [82], maintaining contact between the robot and the object is crucial for pushing with nonholonomic robots. Because frequent repositioning actions are time-consuming. Additionally, planning these actions while avoiding collisions with the object simultaneously is challenging. As an improvement of [82], we will not explain it repeatedly due to space limitation. However, we give experiment results of tracking predefined trajectories with the free pushing method (Fig. 4.10). These

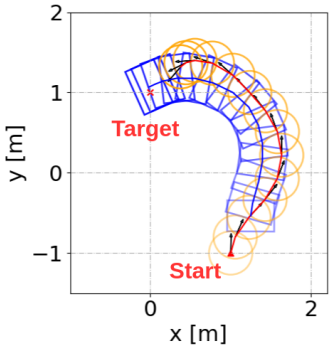


Figure 4.7: Free pushing trail 3 with an initial contact configuration of $d = 0.25$ m and $\phi = 0$ and target of (0, 1). The stable pushing is infeasible using the setup in Fig. 4.6.

results illustrate how the proposed method can continuously push the object along the given path without losing contact. Additionally, the free pushing controller is more maneuverable than [82] to track curly paths without curvature limitation.

Furthermore, we attempted a more challenging task, i.e., pushing a heavy basket filled with random products (Fig. 4.1b). The basket's mass distribution is difficult to measure, complicating accurate system modeling. This test was performed to evaluate the controller's robustness and determine whether it could compensate for small model mismatches. The results of the pushing experiment are shown in Fig. 4.11, where the controller still successfully achieved the pushing task despite the modeling errors. However, it resulted in a longer pushing distance and more changes in the contact configuration. Our results indicate that the mobile base is able to deliver large and heavy objects that cannot be grasped. However, its pushing performance is still limited by motor capabilities and wheel slippage. It is noted that objects that are too heavy may degrade the pushing performance, resulting in jerky motions.

Table 4.2: Pushing performance with the real robot.

Goal	Averaged pushing accuracy (RMSE, m)	Averaged travel distance (m)	Success rate	Number of trails
1	0.074	2.324	0.7	10
2	0.054	1.427	1	10
3	0.120	2.510	0.7	10
4	0.105	3.242	0.8	10
5	0.064	2.957	1	10
6	0.098	4.186	0.8	10

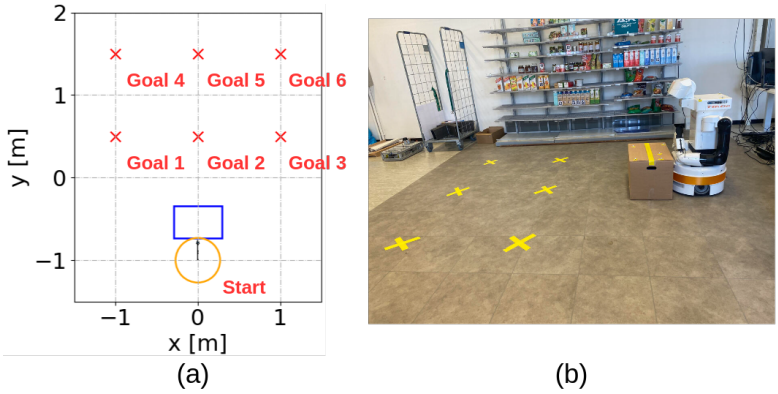


Figure 4.8: The proposed controller is tested in physical experiments where a Tiago robot is used to push a paper box to specified positions. The selected pushing targets are shown in the figure.

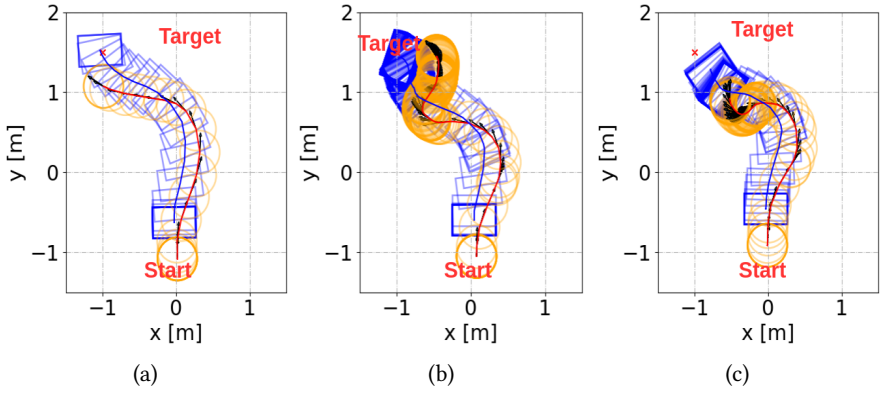


Figure 4.9: Pushing trails with the same goal. The object motion is sensitive to the small difference in the control inputs and the initial states. But the robot achieves the push success by continuously adapting the contact configuration.

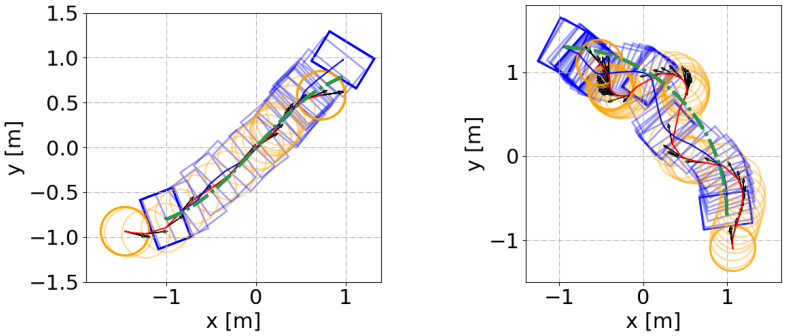


Figure 4.10: Tracking an S-curve (left) and a 1/4 circle (right) while engaging in free pushing. The green line represents the reference path.

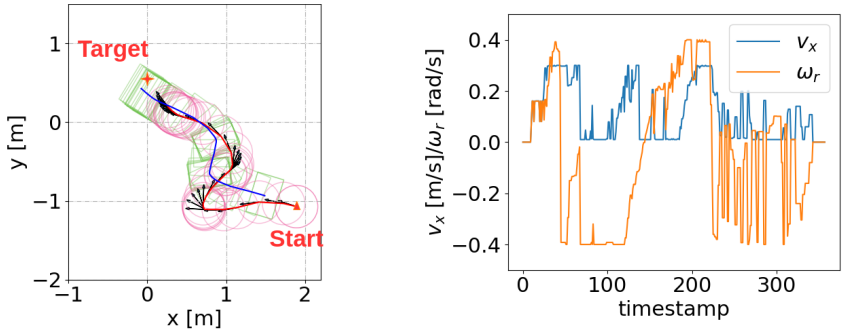


Figure 4.11: The robot is able to successfully push a heavy plastic basket to the target position with the tolerance of modeling error. Pictures of the experiment can be seen in Fig. 4.1b. (Left) the recorded trajectories, (right) the control commands.

4.6 Conclusion

In this chapter, we address the challenge of unwieldy object delivery using a differential-drive mobile robot with a free pushing method. Unlike previous approaches focusing on contact point trajectories, we introduce a continuous contact model to predict pusher-slider dynamics and directly plan robot control inputs. This method ensures feasible pushes and minimizes time and distance by continuously pushing the object without relocations. However, it is limited to regular-shaped objects like rectangles.

In contrast, data-driven controllers are suitable for irregular objects but require extensive data collection for tuning. Training a model for each object is impractical. Given that most unwieldy objects in logistics are regular-shaped, we prefer to make an analytical reactive controller, as introduced. An enhancement to the proposed method could involve identifying parameters in the model during the pushing process.

5

Nonprehensile planar manipulation via differential flatness

Parts of this chapter appeared in:

- **Y. Tang**, M. Wisse, W. Pan, "Reactive Nonprehensile Planar Manipulation via Differential Flatness," submitted to *IEEE Robotics and Automation Letters*, under review.

Chapters 3 and 4 propose two object manipulation methods with mobile bases, focusing on controller design. However, for underactuated nonprehensile manipulation, long-horizon planning remains challenging. This challenge is compounded by the highly nonlinear dynamics and discontinuities in the manipulation process, which motivated the research presented in Chapter 5. In this chapter, we investigate the differential flatness property of the pushing system and utilize it to simplify planning for nonprehensile manipulation. Leveraging the simplicity and computational efficiency of the proposed method, we develop a reactive pushing manipulation approach.

5.1 Introduction

In addition to typical pick-and-place operations, nonprehensile manipulations, which require extensive contact with the physical world, are widely used. However, planning and control for contact-rich manipulations are recognized as particularly challenging, especially for long-horizon tasks.

The difficulty of contact-rich manipulation lies in its complex dynamics, characterized by multimodality under varying contact conditions. Additionally, it requires decisions on when to initiate or break contact, making the decision variables a mix of continuous control inputs and integer contact modes, thus formulating a hybrid optimization problem. [13] addresses the problem using mixed-integer programming, but due to computational complexity, it necessitates an additional offline phase to pre-plan a sequence of contacts. More recently, [14] solves the nonprehensile planar manipulation control problem as a trajectory optimization problem with complementarity constraints. However, this approach can only be solved online for sufficiently short horizons due to computational demands and requires the use of a specialized commercial solver. Even though these methods precisely address the planning and control for planar manipulation, they are limited to scenarios with predefined contact trajectories and short horizons.

To avoid excessive mode enumeration and reduce computational complexity, stable manipulation is widely explored to simplify manipulation planning [19, 39, 79]. This approach emphasizes limiting the applied force through the contact point within the bounds of the friction cone, thereby maintaining contact stability without switching to other contact modes. [44] demonstrate that the robot-object system under stable contact is differentially flat, reducing the manipulation planning problem to that of a Dubins car. Consequently, planning for planar manipulation becomes as straightforward as computing a Dubins curve, which greatly simplifies the problem and offers the benefit of time optimality as well.

Although stable manipulation makes nonprehensile planar manipulation easier, the fixed contact point under stable pushing can only transmit a limited set of forces, thus restricting the system's maneuverability. Our previous research [91] presented a free-pushing approach that approximates the robot-object contact as a sliding joint, smoothing transitions across different contact points. Compared to stable manipulation, the free-pushing approach exhibits highly reactive manipulation behavior, freely maneuvering around the object, like reactively catching the object during manipulation. However, the

Figure 5.1: Manipulating object to the goal with a low-cost mobile robot: Mirte Master. The transparency of the robots and boxes indicates their movement.

underactuated nature of the nonprehensile manipulation system poses the problem of the controller sometimes getting trapped in local optima, necessitating a global planner to plan a global trajectory beforehand. Optimizing such a global trajectory for contact-based manipulation is challenging due to the highly nonlinear dynamics, prompting us to investigate how we can simplify the planning of free pushing.

The key contribution of this chapter is that we demonstrate that the free-pushing system is also differentially flat like the one with stable contact. This means the manipulation planning problem can be simplified as a trajectory planning problem for the object. As a result, we can derive the robot trajectory directly from the object trajectory. This makes it easy to plan a reaching trajectory where the start of the robot's pushing trajectory is where it plans to initiate contact. The entire manipulation plan, including reaching and manipulating, only takes an average of 2 milliseconds to solve. With the benefit of fast calculations, our method offers robustness, ensuring quick recovery from disturbances or uncertainties during manipulation. Furthermore, our method can easily incorporate obstacle avoidance using the planned trajectory, effectively handling manipulation in cluttered environments.

To conclude, the proposed method has the following advantages compared to the other methods:

- It is simple to design, fast to solve, and robust to uncertainties.
- Unlike other approaches, it has the advantage of simplicity with only one parameter to tune. No learning or extra tactile sensors are required.
- The global planner extends the usage of the free-pushing controller even in cluttered environments.

5.2 Preliminaries

5.2.1 Problem definition

In this chapter, we address the problem of reactive nonprehensile planar manipulation. The significant amount of contact during manipulation makes modeling and planning challenging, particularly in cluttered environments where obstacle avoidance is necessary. Additionally, uncertainties arising from contact often lead to manipulation failures, necessitating a method that can adapt reactively to changes. To achieve this task, we need to solve the following problems:

1. How to plan for a feasible and collision-free trajectory for the planar manipulation?
2. How to reactively handle disturbances and recover during the manipulation?

We simplify the planning problem by proving that the nonprehensile planar manipulation system is differentially flat: the trajectory of the object can fully determine the system behavior, as explained in Section 5.3. This reduces manipulation planning to a trajectory planning problem for the object (Section 5.4). Furthermore, our reactive approach addresses manipulation failures caused by uncertainties or external disturbances. It unifies reaching and pushing tasks into a single trajectory optimization framework, ensuring fast computation and robustness to disturbances, as detailed in Section 5.5.

5.2.2 Quasi-static assumption

Due to the complexities in modeling impacts and the typically low speeds involved in manipulation, the contacts in this work are assumed to be quasi-static. This means the object's motion is slow enough that inertial forces can be neglected compared to other forces acting on the system [92]. This assumption simplifies the motion model of the object by only considering a balance between the pushing and friction forces. Although the quasi-static assumption may not ensure an accurate model in some dynamic cases, it is a powerful tool for simplifying the analysis of planar manipulation tasks. We rely on our reactive design to compensate for any imprecision in the model.

5.3 Differential flatness

Differential flatness is a concept in control theory that simplifies controlling and planning the trajectories of complex dynamical systems. Many systems have nonlinear dynamics, making control and planning challenging. In a differentially flat system, we can use a simpler set of variables (flat outputs) to "flatten" the system's behavior, transforming a complicated planning problem into a simpler one.

A system is said to be differentially flat if a set of differentially independent variables, known as flat outputs $\mathbf{y}$, can be found such that (1) The flat outputs $\mathbf{y}$ can be expressed as functions of the system's state variables $\mathbf{x} \in \mathbb{R}^n$, control inputs $\mathbf{u} \in \mathbb{R}^m$, and a finite number of their derivatives. (2) Conversely, all the state variables and control inputs of the system

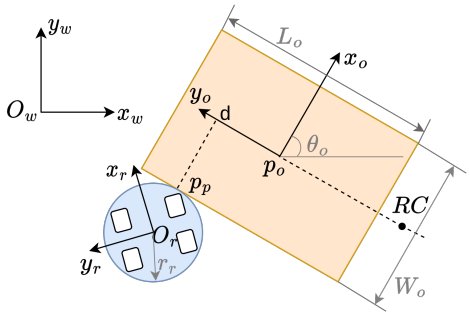


Figure 5.2: Coordinate and frame configuration. The blue circle represents the robot, while the orange rectangle denotes the object. Key local frames are illustrated.

can be expressed as functions of the flat outputs $\mathbf{y}$ and a finite number of their derivatives [93]. From a mathematics perspective, a nonlinear system in the form of $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ can be stated as differentially flat if there exists a set of flat outputs $\mathbf{y} \in \mathbb{R}^m$ such that

$$\begin{aligned} \mathbf{y} &= f_1(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, ...) \\ \mathbf{x} &= f_2(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, ...) \\ \mathbf{u} &= f_3(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, ...) \end{aligned} \tag{5.1}$$

By finding the appropriate flat outputs, the complex task of trajectory planning can be reduced to a simpler problem involving these outputs and their derivatives. This property enables the trajectory of the flat output to determine the entire system behavior.

For the cylinder-shaped robot (the Mirte Master robot as shown in Fig. 5.1), its center is also the central point of the configuration of the four wheels. The robot has a radius r_r. We use a rectangular object as an example, with length L_o and width W_o. However, the proposed method can be generalized to other shapes that have a plane to contact and maneuver around. The key to proving the flatness of the system is identifying the flat output.

Proposition 1. *The position of the object's center of mass, $\mathbf{p}_o(t)$, is the flat output of the pusher-slider system.*

Assumption 1. *The contact under this work is assumed to be quasi-static [92, 94].*

Assumption 2. *The contact between the robot and the object is assumed to be friction-less [91].*

Proof. First, the dimension of $\mathbf{p}_o(t)$ matches that of the control input. Next, we demonstrate that the trajectory of the entire system can be derived (without integration) from the trajectory of the object. Given the object's position $\mathbf{p}_o(t) = [x_o(t), y_o(t)]$ along a geometric path $\tau(t)$, we can express its orientation θ_o, the path's tangent direction $T(t)$, and curvature

$\kappa(t)$ as (illustrated in Fig. 5.3b):

$$\begin{aligned} \theta_o &= \arctan\frac{\dot{y}_o}{\dot{x}_o}, \\ T(t) &= \frac{1}{\sqrt{\dot{x}_o^2+\dot{y}_o^2}}\begin{bmatrix}\dot{x}_o\\ \dot{y}_o\end{bmatrix} \\ \kappa(t) &= \frac{\dot{x}_o\ddot{y}_o-\dot{y}_o\ddot{x}_o}{(\dot{x}_o^2+\dot{y}_o^2)^{3/2}} \end{aligned} \tag{5.2}$$

The normal vector of the curve at $p_o(t)$ can be achieved from $T(t)$ by a counterclockwise rotation of $\frac{\pi}{2}$,

$$\begin{aligned} N(t) &= R(\frac{\pi}{2})\cdot T(t) \\ &= \frac{1}{\sqrt{\dot{x}_o^2+\dot{y}_o^2}}\begin{bmatrix}-\dot{y}_o\\ \dot{x}_o\end{bmatrix} \end{aligned} \tag{5.3}$$

where $R(\theta)$ denotes the rotation matrix.

The instantaneous rotation center of the curve, RC, is

$$\begin{aligned} \mathrm{RC}(t) &= \mathbf{p}_o(t)+\frac{1}{\kappa(t)}N(t) \\ &= \mathbf{p}_o(t)+\frac{\dot{x}_o^2+\dot{y}_o^2}{\dot{x}_o\ddot{y}_o-\dot{y}_o\ddot{x}_o}\begin{bmatrix}-\dot{y}_o\\ \dot{x}_o\end{bmatrix} \end{aligned} \tag{5.4}$$

According to [44], the distances from the object center to the pushing force and instantaneous rotation center are inversely perpendicular (Fig. 5.2). This geometric relationship allows us to express the trajectory curvature $\kappa(t)$ in terms of the contact point distance d:

$$\kappa(t) = c\cdot d \tag{5.5}$$

Since the contact point, $\mathbf{p}_p$, is opposite the object's origin O_o and located on the object's surface,

$$\begin{aligned} \mathbf{p}_p(t) &= \mathbf{p}_o(t)+c\kappa(t)N(t)-\frac{W_o}{2}T(t) \\ &= \mathbf{p}_o(t)+c\frac{\dot{x}_o\ddot{y}_o-\dot{y}_o\ddot{x}_o}{(\dot{x}_o^2+\dot{y}_o^2)^2}\begin{bmatrix}-\dot{y}_o\\ \dot{x}_o\end{bmatrix}-\frac{W_o}{2}\frac{1}{\sqrt{\dot{x}_o^2+\dot{y}_o^2}}\begin{bmatrix}\dot{x}_o\\ \dot{y}_o\end{bmatrix} \end{aligned} \tag{5.6}$$

The robot center is located r_r from the contact point in the inverse direction of $T(t)$.

$$\begin{aligned} \mathbf{p}_r(t) &= \mathbf{p}_p(t)-r_rT(t) \\ &= \mathbf{p}_o(t)+c\frac{\dot{x}_o\ddot{y}_o-\dot{y}_o\ddot{x}_o}{(\dot{x}_o^2+\dot{y}_o^2)^2}\begin{bmatrix}-\dot{y}_o\\ \dot{x}_o\end{bmatrix}- \\ &\quad(\frac{W_o}{2}+r_r)\frac{1}{\sqrt{\dot{x}_o^2+\dot{y}_o^2}}\begin{bmatrix}\dot{x}_o\\ \dot{y}_o\end{bmatrix} \end{aligned} \tag{5.7}$$

The control input of the system, $\mathbf{u}_\mathrm{r}$, is the desired velocity for the robot, which can be obtained as the derivative of $\mathbf{p}_\mathrm{r}(t)$:

$$\mathbf{u}_\mathrm{r} = \begin{bmatrix} v_\mathrm{r} \\ \omega_\mathrm{r} \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_r(t)^2 + \dot{y}_r(t)^2} \\ \arctan(\frac{\dot{y}_r(t)}{\dot{x}_r(t)}) \end{bmatrix} \tag{5.8}$$

where $\mathbf{p}_\mathrm{r} = [x_\mathrm{r}, y_\mathrm{r}]$ which indicates the position of the robot.

Now, we have the robot position and control input expressed in terms of the flat output (object position) and its derivatives. Thus, without calculating the pushing force and reasoning the resulting movement of the object, the robot's position and motion can be derived from the object's position and its derivatives. The object's trajectory determines the robot's trajectory and its motion. We can prove that the position of the object's origin is the flat output of the pusher-slider system.

5.4 Trajectory optimization

Since the flat outputs are differentially independent, there are no feasibility constraints on the trajectories in the flat space. Consequently, every trajectory in the flat space can be mapped to a feasible trajectory in the state-input space [95]. Trajectories planned in the flat space can directly yield control inputs without considering the system's complex dynamics, thereby simplifying the trajectory planning process.

Hereby, global planning of the nonprehensile planar manipulation is simplified as the trajectory planning problem for the object. Given the object's starting pose $[x_o^s, y_o^s, \theta_o^s]^\top$, goal pose $[x_o^g, y_o^g, \theta_o^g]^\top$, and the position of the obstacles, we can easily plan for a trajectory of the object that is pushed to the goal while avoiding the obstacle.

5.4.1 Representation of the trajectory

We use piecewise Bezier curves, $\tau = [\tau_x, \tau_y]$, to represent the object trajectory in each dimension $\mu \in x, y$. Each Bezier segment is parameterized to time t and defined over a fixed interval $[0, 1]$, consisting of a set of control points $\mathbf{p}^i$.

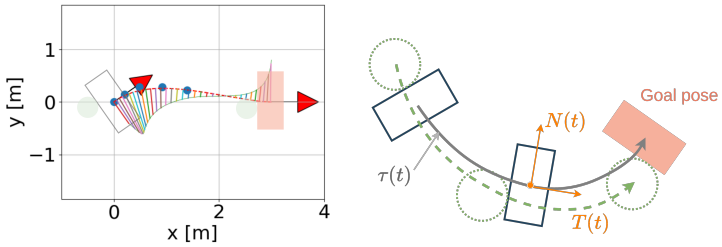
For example, the j^{th} segment of the Bezier curve in μ dimension, τ_μ, is defined as

$$\tau_{\mu j}(t) = \sum \mathbf{p}_{\mu j}^i b_n^i(t) \tag{5.9}$$

where $b_n^i(t)$ is the Bernstein polynomial basis with n being the degree of the basis and $\binom{n}{i}$ being the binomial coefficient. Here we use the cubic Bezier curve where $n = 3$. $\mathbf{p}_{\mu j}^i$ is the i^{th} control point of of the j^{th} segment of the trajectory.

Accordingly, we have a piece-wise curve of m segments:

$$\tau_\mu(t) = \begin{bmatrix} \tau_{\mu 1}, \\ \tau_{\mu 2}, \\ ..., \\ \tau_{\mu m} \end{bmatrix} = \begin{cases} \sum_{i=0}^n \mathbf{p}_{\mu 1}^i b_n^i(\frac{t-T_0}{T_1-T_0}), t \in [T_0, T_1] \\ \sum_{i=0}^n \mathbf{p}_{\mu 2}^i b_n^i(\frac{t-T_1}{T_2-T_1}), t \in [T_1, T_2] \\ ... \\ \sum_{i=0}^n \mathbf{p}_{\mu m}^i b_n^i(\frac{t-T_{m-1}}{T_m-T_{m-1}}), t \in [T_{m-1}, T_m] \end{cases} \tag{5.10}$$



(a) (b)

Figure 5.3: Example of the optimized Bezier curve. (a) Given the object's start and goal poses, a Bezier curve (dashed red line) is optimized to represent the object's trajectory. The blue dots are the control points of the first curve segment. The white and pink patches represent the object's start and goal poses, respectively. The colorful lines indicate the instantaneous rotation radius of the trajectory, while the green circles indicate the robot's positions. (b) The robot's trajectory is recovered from the optimized object trajectory, allowing the robot to maneuver and push the object along the planned path to reach the goal pose.

5

where $T_1, T_2, ..., T_m$ are the end times of each segment.

5.4.2 Objective

According to Eq. 5.5, the curvature of the object trajectory is proportional to the distance of the contact point to the x direction of the object frame. During the pushing process, we want to maintain contact between the robot and the object and avoid sliding away from the object. Therefore, we aim to minimize the curvature of the planned object trajectory to keep the contact point closer to the center of the object.

The curvature variation energy is given by $E_{cv}(P) = \int_0^l [\dot{\kappa}(s)]^2 dt$, where s is the arc parameter, l is the arc length of $\tau(t)$. Minimizing the curvature of the trajectory is challenging in real-time because the curvature involves the second derivatives of position, leading to a non-quadratic cost function, which complicates the optimization problem and can make it non-convex. According to [96, 97], this nonlinear energy function can be approximated by the quadratic jerk energy. The jerk energy of each Bezier curve segment is defined as:

$$\begin{aligned} E_{jerk}(\tau_{\mu j}) &= \int_{T_{j-1}}^{T_j} \|\dddot{\tau}_{\mu j}(t)\|^2 dt \\ &= \mathbf{p}_{\mu j}^\top Q_{\mu j} \mathbf{p}_{\mu j} \end{aligned} \tag{5.11}$$

where $\dddot{\tau}_{\mu j}(t) = 6(-p_{\mu j}^0 + 3p_{\mu j}^1 - 3p_{\mu j}^2 + p_{\mu j}^3)$, $\mathbf{p}_{\mu j} = [p_{\mu j}^0, p_{\mu j}^1, p_{\mu j}^2, p_{\mu j}^3]^\top$, $Q_{\mu j} = 36 * \begin{bmatrix} 1,-3,3,-1 \\ -3,9,-9,3 \\ 3,-9,9,-3 \\ -1,3,-3,1 \end{bmatrix}$

is the symmetric Hessian matrix.

Then the final cost function of the trajectory optimization problem is

$$J = \sum_{\mu \in x,y} \sum_{j=1,2,...,m} E_{jerk}(\tau_{\mu j}) \tag{5.12}$$

5.4.3 Enforcing constraints

A set of constraints is set to ensure the trajectory's smoothness and feasibility in generating a trajectory containing piecewise trajectory segments. Furthermore, safety constraints, start pose constraints, and goal pose constraints are also applied to ensure that we can achieve the desired path. For each piece of the Bezier curve, higher-order derivatives can be represented as a linear combination of corresponding lower-order control points, written as:

$$a_{\mu j}^{0,i} = p_{\mu j}^{i}, a_{\mu j}^{l,i} = \frac{n!}{(n-l)!} \cdot (a_{\mu j}^{l-1,i+1} - a_{\mu j}^{l-1,i}), l \geq 1, \tag{5.13}$$

where l is the order of the derivative, and n is the degree of the Bernstein basis.

Continuity constraints

The trajectory should be continuous at all derivatives at the connecting points of the segments. Therefore, the continuity constraint is applied to the consecutive curves:

$$a_{\mu j}^{\sigma,n} = a_{\mu j+1}^{\sigma,0}, 1 \leq j \leq m-1, \mu \in x, y, \tag{5.14}$$

where σ indicates the σ^{th} derivative, $0 \leq \sigma \leq n$.

Safety constraints

Bezier curves have the property that the curve lies completely within the convex hull of its control points. Thus, safety constraints are enforced on the control points to ensure that they are within a convex polygon without overlapping with the obstacle, which adds boundary limits on the control points:

$$\beta^{-} < p_{\mu j}^{i} < \beta^{+} \tag{5.15}$$

where $1 \leq j \leq m-1, \mu \in x, y, z, j \in 0, 1, ..., n$.

Start and goal pose constraints

The beginning and end points of the trajectory are defined by the start and goal points of the object, resulting in start and goal pose constraints:

$$\begin{aligned} \tau_{x0}(0) &= x_o^s \\ \tau_{y0}(0) &= y_o^s \\ \tau'_{x0}(0) &= \cos(\theta_o^s) \\ \tau'_{y0}(0) &= \sin(\theta_o^s) \\ \tau_{xm}(T_m) &= x_o^g \\ \tau_{ym}(T_m) &= y_o^g \\ \tau'_{xm}(T_m) &= \cos(\theta_o^g) \\ \tau'_{ym}(T_m) &= \sin(\theta_o^g) \end{aligned} \tag{5.16}$$

The relative position between the robot and the object at the beginning determines the starting curvature of the curve, $\kappa(0)$:

$$\kappa(0) = c/d(0) \tag{5.17}$$

where $d(0)$ is the distance from the object's origin to the line of the pushing force at the beginning, which will be optimized.

The starting contact point is limited at the contact patch such that $-W_o/2 < d(0) < W_o/2$. The continuity constraints and the start and goal pose constraints are formulated as linear equality constraints on the decision variables, denoted as $\boldsymbol{A}_{eq}\mathbf{c} = \mathbf{b}_{eq}$. The safety constraints form linear inequality constraints, written as $\boldsymbol{A}_{ie}\mathbf{c} = \mathbf{b}_{ie}$. The final trajectory optimization problem is formulated as a linear quadratic optimization problem as follows. An example of the optimized trajectory can be found in Figure. 5.3.

$$\min_{p^i_{\mu j}, d(0)} \quad \mathbf{p}_{\mu j}^\top Q_{\mu j} \mathbf{p}_{\mu j} \tag{5.18a}$$

$$\boldsymbol{A}_{eq}\mathbf{c} = \mathbf{b}_{eq}, \tag{5.18b}$$

$$\boldsymbol{A}_{ie}\mathbf{c} = \mathbf{b}_{ie}. \tag{5.18c}$$

where $p^i_{\mu j}$ are the control points of the Bezier curve segments.

5.5 Planning and control

In the previous section, we described how to plan for planar manipulation. By determining the trajectory of the object, we can infer the behavior of the entire system. However, this planning only addresses the phase when the robot and the object are in contact during manipulation. Maintaining continuous contact during nonprehensile manipulation is challenging due to uncertainties and disturbances. Thus, a robust approach must include the ability to recover from lost contact. Here, we introduce a reactive manipulation pipeline.

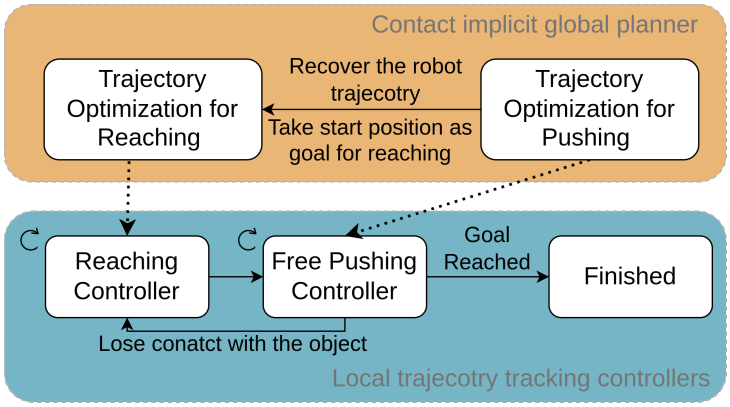


Figure 5.4: Planning and control pipeline. The dashed arrow indicates the transmission of the data flow, like planned trajectories. The open circle arrow indicates that the controller recursively follows the reference trajectory until the termination condition is met.

The schematic of the pipeline is shown in Fig. 5.4. It features a contact-implicit global planner that continuously plans an object trajectory under pushing and a robot trajectory to reach the object. Since the robot's pushing trajectory can be derived from the object's trajectory using the differential flatness property, the goal of the robot's reaching task is to position itself at the start of the pushing trajectory. Trajectory optimization for the robot reaching task follows the same principles as those for the object, as illustrated in the previous section.

When the robot is not in the initial pushing position, a reaching controller is employed to track the robot's reaching trajectory until it reaches the target point. Once the robot makes contact with the object, a free pushing controller [91] is used to follow the pushing trajectory until the object reaches its goal. The free pushing controller adapts to uncertainties by dynamically changing the contact point on the object. Additionally, it is always ready to switch back to the reaching controller if the robot loses contact with the object, making the pushing approach more reactive to uncertainties.

We use Model Predictive Controllers (MPC) to track the trajectories. For simplicity, the details of the controllers are not included here; please refer to our previous work [91] for comprehensive information. While we employ MPC controllers in this approach, other local controllers can also be used to track the planned trajectory.

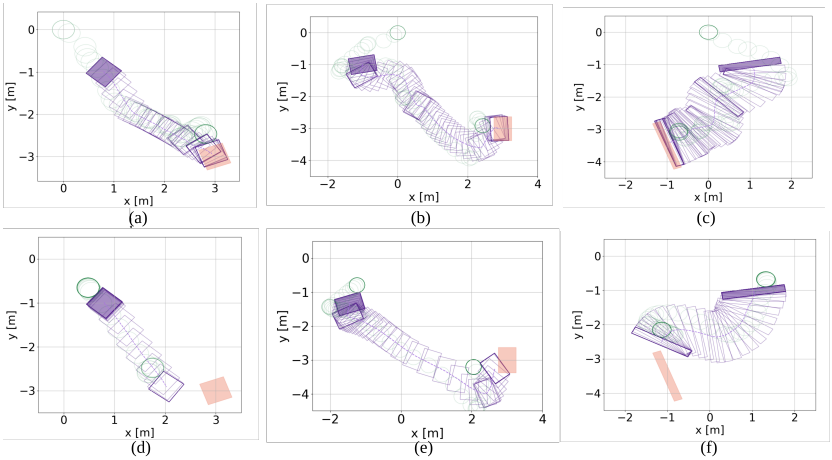


Figure 5.5: Goal reaching experiments. (a, b, c) Manipulate objects with different sizes with the proposed reactive manipulation method, including reaching and pushing phases. (d, e, f) Manipulate objects with different sizes with the free pushing controller in [91], without the global reference trajectory.

5.6 Experimental results

This section presents the experimental results performed in simulation and in the real world. The trajectory optimization problem is implemented using CasADi and solved with the IPOPT solver. Aside from the third-party solver, the code is implemented in Python and runs on a laptop with the processor i7-11800H. With the trajectory optimization method proposed in Section. 5.4, the robot can plan for the planar manipulation easier and faster, which only takes 2 ms in average to make one contact-implicit manipulation plan. Compared to manipulation without the global plan, it avoids going to the local optima and can recover from the infeasible configurations. Compared to the more reactive MPPI (Model Predictive Path Integral) controller, the proposed approach shows the benefits of producing more efficient manipulations and demonstrates significant simplicity. The real experiments show its ability to deal with unexpected disturbances during manipulation. Experiments are also shown in the supplementary video.

5.6.1 Simulation experiments

We use Isaac Gym as our simulation environment for its accurate contact dynamics and realistic physical interaction simulations. We present three examples of manipulating objects with different shapes, as shown in Fig. 5a-5c. The dimensions of the manipulated objects are 0.5x0.5 m, 0.5x0.75 m, and 0.2x1.5 m, each weighing 2 kg. The parameter c is set to 0.02.

Starting from the purple patch, the controller's goal is to transport the object to the pink patch. At the beginning of the experiment, the robot is not in contact with the object. The manipulation trajectory optimizes for the position where the robot initiates the contact,

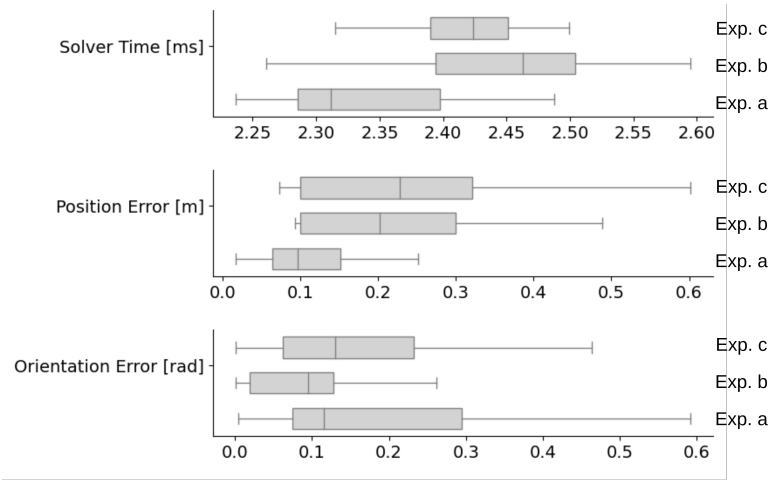


Figure 5.6: Box plots for various metrics related to the goal-reaching performance, including solver time, position error and orientation error. Each box plot shows the distribution of data points for the respective metric, with the median represented by the central line and the interquartile range represented by the box. Each metric includes data from three groups corresponding to the experimental conditions in Fig. 5a, 5b, 5c.

respectively, starting from the middle, left side, and right side of the object for Fig. 5a, 5b and 5c. After getting in contact with the object, the robot maneuvers around it to adjust its pose and push it forward to the goal position smoothly. The experiments are repeated 50 times for each condition, whose statistical results are shown in Fig. 5.6. The average solver times for the manipulation trajectory optimization are 2.34, 2.45, and 2.41 milliseconds, respectively. The average position errors are 0.16, 0.23, and 0.25 meters, while the average orientation errors are 0.23, 0.16, and 0.17 radians. This highlights the efficiency and effectiveness of the trajectory optimization process, ensuring quick computations and relatively low errors in both position and orientation.

In comparison, some experiments were conducted without the proposed contact-implicit trajectory, using only the goal-driven finite-horizon controller [91], as shown in Fig. 5d, 5e and 5f. This controller starts with contact with the object, eliminating the need for the reaching phase. Initially, the process proceeds well, but it ultimately results in local-optima configurations as shown in Fig. 5d and 5f. As an underactuated system, the robot struggles to transmit sufficient force to push the object to the goal while minimizing cost. Due to uncertainties in contact modeling, the object's movement under pushing can be unpredictable, sometimes breaking robot-object contact, as shown in Fig 5e. Without a reactive recovery method, the configuration becomes infeasible for the pushing controller, preventing the manipulation from continuing.

For the goal-reaching experiments, the proposed contact-implicit trajectory optimization effectively guides planar manipulation to reach the target while avoiding local optima traps. Due to its fast-solving capability, it can quickly generate new manipulation plans to

recover from unexpected situations, such as losing contact with the object.

The proposed approach proves especially useful in obstacle-aware environments, where obstacle avoidance complicates the manipulation planning, increasing the likelihood of encountering local optima. In two obstacle-aware environment settings, our robot successfully pushes the object along the planned trajectory and avoids collisions with obstacles, as shown in Fig. 5.7c and 5.7a. We compare our method to a highly reactive manipulation controller based on Model Predictive Path Integral (MPPI) control [27]. MPPI is considered one of the most reactive methods in contact-based manipulation due to its ability to dynamically switch contacts from all possible locations.

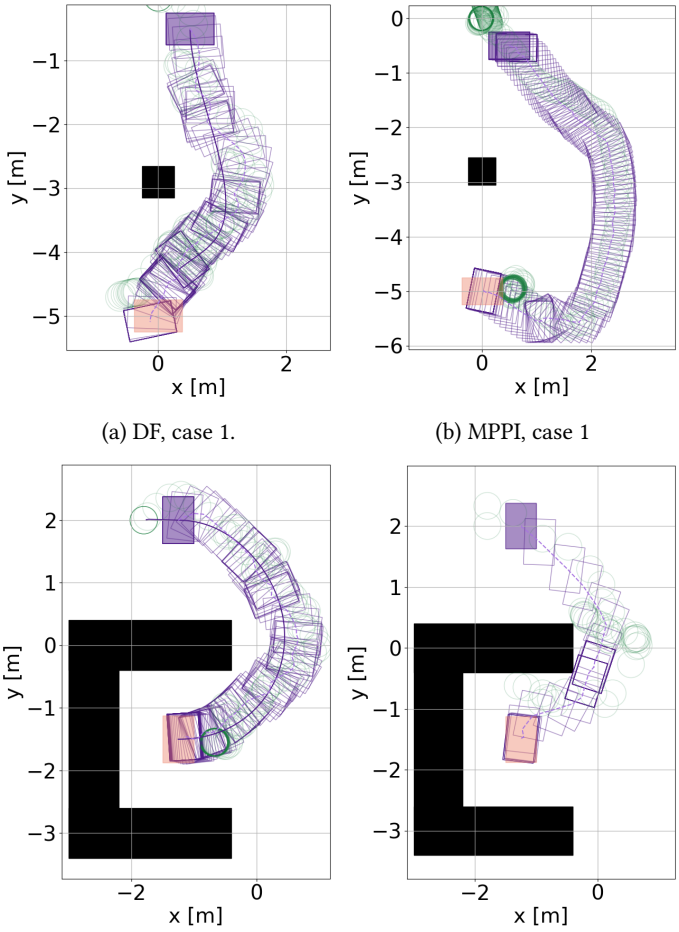
The MPPI-based controller utilizes the GPU-parallelizable IsaacGym simulator to compute forward dynamics, enabling rapid sampling and evaluating potential actions. We tuned the MPPI hyperparameters for optimal performance, using $K = 1000$ simulated environments in IsaacGym and a planning horizon of $T = 20$. The resulting trajectories from MPPI can be seen in Fig. 5.7d and 5.7b. Although the MPPI-controlled robot eventually completes the task, its trajectory is significantly longer than that of our proposed method, as seen in Fig. 5.7d. Additionally, it collides with an obstacle in Fig. 5.7b. While MPPI excels in solving problems with non-linear and discontinuous dynamics, such as contact-rich manipulations, it relies on performing hundreds of parallel rollouts, which demands substantial computational resources.

In an ablation study, 30 experiments were conducted for each scenario using both methods. The results are shown in Fig. 5.8. In scenario Fig. 5.7a, 5.7b, both the proposed and the MPPI methods achieved success rates of 86.7%. In another scenario Fig. 5.7c, 5.7d, the proposed method achieved a 100% success rate, while the MPPI method achieved 66.7%. Despite the MPPI method's slightly lower success rate, it cannot ensure collision avoidance. The success rates of cases without collision avoidance drop to 56.7% and 23.3%. The sampling-based nature of MPPI introduces an element of randomness in action selection. If the samples fail to find a path to the goal, the system can become trapped in a sample distribution that fails to explore the correct direction. In contrast, our method provides a more deterministic and computationally efficient approach while maintaining the ability to react to changes in the environment. Although the MPPI method achieves a lower average position error than the proposed approach in the success trials, its average orientation error is higher and exhibits greater variance. Furthermore, we observed that increasing the weight for orientation error in the MPPI cost function leads to an even lower success rate. Overall, the proposed method demonstrates advantages in simplicity and computational efficiency compared to MPPI.

5.6.2 Real-world results

We also tested the proposed approach in real-world experiments. The robot we used is the customized Mirte Master robot, which is a low-cost education robot. The radius of the robot is 0.22 m. The object to be transported has a size of 0.37x0.41 m. The robot and object states are provided by a motion capture system (OptiTrack) and a Kalman filter, which runs at 120Hz. A goal-reaching experiment, replicating the setup shown in Fig. 5.5, was conducted. The experiment result is presented in Fig. 5.1.

The performance of the proposed approach was also tested under disturbances, as shown in Fig. 5.9. Initially, the robot is not in contact with the object, so a contact-implicit trajectory is planned. Then, the robot moves to the first planned contact position, as shown in Fig. 5.9a. The robot started transporting the object aiming to the goal while being disturbed by a person in Fig. 5.9b-5.9e. By quickly planning for a new contact-implicit trajectory, the robot easily recovered from the disturbances. Finally, it successfully moves the object to the goal position, as shown in Fig. 5.9f. The full trajectory is depicted in Fig. 5.10.



(a) DF, case 1.

(b) MPPI, case 1

(c) DF, case 2.

(d) MPPI, case 2.

Figure 5.7: Manipulation with obstacle avoidance. DF represents the proposed reactive manipulation based on differential flatness. Using this approach, the robot successfully pushes the object along the planned trajectory while avoiding obstacles, as shown in (a) and (c). Compared to the Model Predictive Path Integral (MPPI) control, the resulting trajectory successfully avoids obstacles but is significantly longer than that of the proposed method.

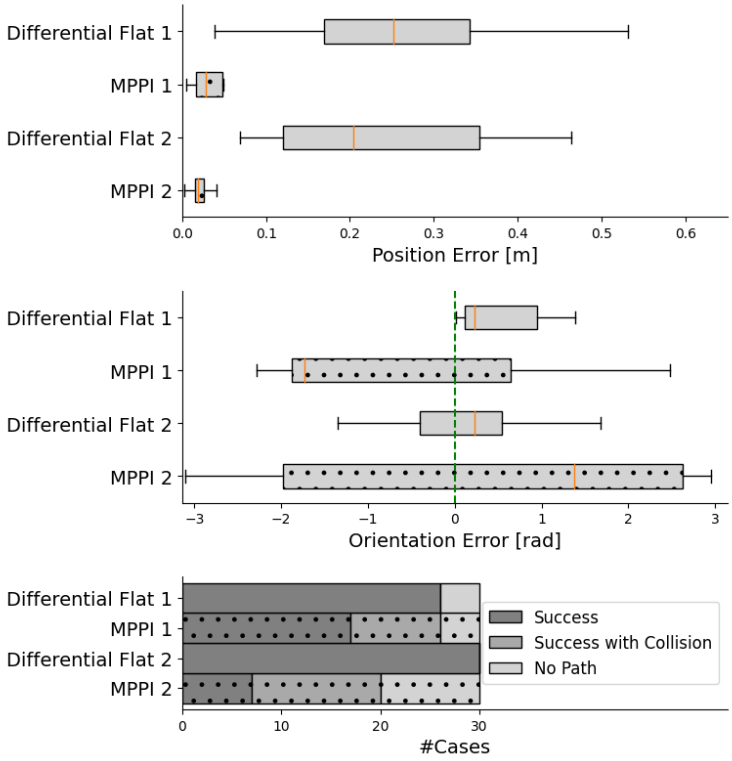


Figure 5.8: Performance comparison between the proposed method and MPPI method across three metrics: position error, orientation error, and success rates.

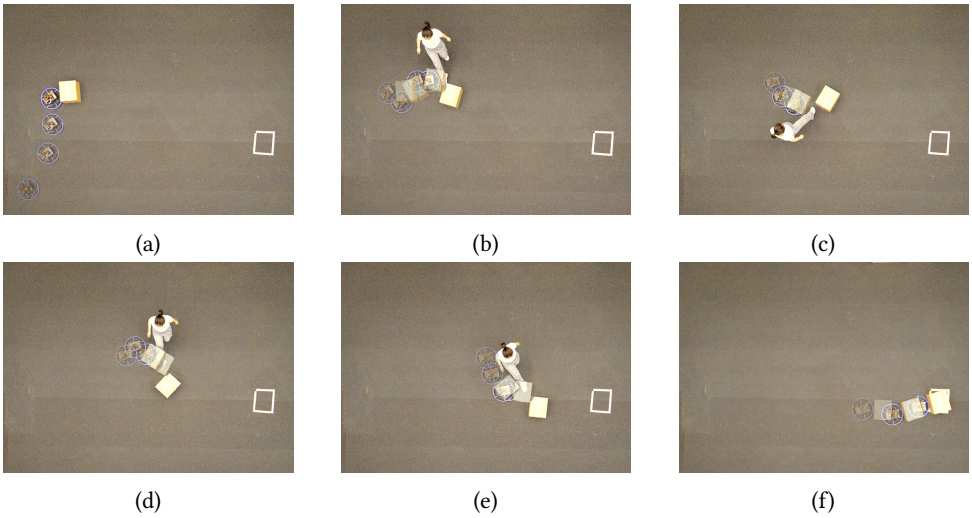


Figure 5.9: Manipulation under disturbances. The transparency of the robots and boxes represents their movement. The robot aims to push the paper box to the goal, indicated by the white rectangle. In (b-e), random disturbances are introduced, yet the robot successfully adapts the contact configuration and continues pushing smoothly. The paper box is ultimately delivered to the goal, as shown in (f).

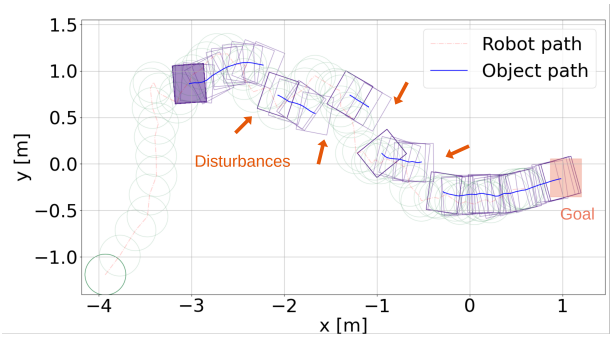


Figure 5.10: The overall trajectory from the experiment in Fig. 5.9 is shown. The robot successfully handles random disturbances and completes the pushing task.

5.7 Conclusion

This chapter introduces a reactive planning and control approach for nonprehensile planar manipulations. We first proved that the underactuated pushing system is differentially flat. By exploiting this differential flatness property, we simplify the pushing planning problem as a trajectory optimization problem for the object, thus eliminating the need for complex integration techniques. Additionally, our method formulates the initial contact point as a decision variable, enabling seamless integration of reaching and pushing tasks into a unified trajectory optimization framework. This holistic approach results in a contact-implicit planner capable of reacting to unforeseen circumstances, such as losing contact with the object or changes in goal pose. Through experimental validation, our method offers reactivity, simplicity, and robustness.

While exploiting differential flatness has significantly simplified contact-implicit trajectory planning, further improvements are possible. Future work will focus on reducing computational demands of the local MPC controller. Developing a feedback linearization controller based on differential flatness [44] could streamline the control problem, potentially reducing complexity. This avenue will be explored in our future research.

6

Reinforcement learning compensated extended Kalman filter for state estimation

Parts of this chapter appeared in:

- **Y. Tang**, L. Hu, Q. Zhang, W. Pan, "Reinforcement learning compensated extended Kalman filter for attitude estimation" *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*
- L. Hu*, **Y. Tang***, Z. Zhou, W. Pan, "Reinforcement learning for orientation estimation using inertial sensors with performance guarantee" *IEEE International Conference on Robotics and Automation (ICRA), 2021.*

Given the importance of state estimation in nonprehensile manipulation, this chapter presents our work on improving the robustness of state estimation using reinforcement learning. We focus on its implementation in a simple case: attitude estimation using sensor data from Inertial Measurement Units (IMU). However, this approach can be easily adapted to state estimation problems in higher dimensions. The proposed method leverages ideas from classic Kalman Filters and improves estimation robustness by learning the estimation gain through reinforcement learning. Our approach demonstrates superior performance compared to conventional methods in three challenging scenarios: inaccurate initial state estimates, inaccurate filter gains, and even non-Gaussian noises. Experiments with real-world data further validate the effectiveness of the proposed method.

6.1 Introduction

IMU is widely used to provide accurate attitude estimation in many fields, including: aerospace [98], robotics [99] and human motion analysis [100]. Even though the gyroscope alone can compute the sensor's orientation by integrating the angular velocity over time, it suffers from accumulated errors, especially in the drift of angular estimation [101]. A data fusion approach using additional sensors such as an accelerometer and magnetometer is often adopted to achieve higher attitude estimation accuracy. A variety of estimation methods such as the extended Kalman filter (EKF) [102], unscented Kalman filter (UKF) [103] and complementary filter (CF) [104] have been proposed, which compute a more reliable estimate using the data collected from all available sensors.

The EKF and UKF, as two variants of the Kalman filter, use linearisation and deterministic sampling methods, respectively, to obtain a more accurate estimate for nonlinear dynamic systems. The CF uses an optimized gradient descent algorithm to compute the gyroscope measurement error direction as a quaternion derivative [59]. Nonetheless, all three methods are sensitive to the initial state estimate. An accurate initial state estimate can lead to the estimation's fast convergence, but the initial estimate is typically chosen empirically, and thus, it is hard to guarantee its accuracy. Furthermore, a filter gain tuning procedure is usually required when deploying filter algorithms in real-world systems. Due to approximation or numerical optimization methods used in calculating the filter gain, the optimality of the estimation algorithms does not hold, necessitating manual tuning of the filter gains. Moreover, the motion acceleration of the sensor may align with the gravity direction sporadically. This will change the level of measurement noise at some time instants/intervals. Using the classical estimation methods without tuning the filter gain will suffer from slow convergence or even divergence in estimation performance.

Several approaches have been proposed to tune the filter gain. [105] proposed a fuzzy processing method to improve the convergence rate. [106] proposes a switching architecture to prioritize the measurement of sensors in different working conditions to yield robust performance. Learning-based methods are also used to calibrate the IMU parameters and compute gyro corrections that filter undesirable errors in the raw IMU signals [61].

This chapter tries to combine reinforcement learning (RL) with the EKF to design a

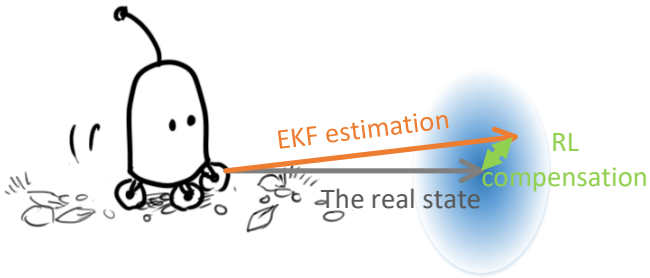


Figure 6.1: A cartoon illustration of the extended Kalman filter with reinforcement learning compensation.

tuning-free estimation algorithm insensitive to inaccurate initial state estimate and filter gain. It leverages the merits of both data-driven RL and model-based probabilistic methods instead of only using RL to train the filter gain [69]. The key idea is to add a learnable policy using RL on top of a referenced gain using EKF. Specifically, an RL policy is trained to compute the gyro correction according to the estimation residuals. Then the learned RL policy is applied and acts as a supplement correction based on the EKF. Intuitively, the learned RL policy compensates strongly when the EKF performs bad (big estimate residual) and remains "idle" if the EKF performs well. Thanks to RL, there is no need to tune the filter parameters in our proposed method manually. As an additional benefit of the added RL in the estimation, the proposed method even works well with non-Gaussian noise, as shown in the experiments. On the other hand, compared with [69, 107] where a pure RL based algorithm is trained/learned from scratch, our proposed method uses the EKF to provide a good starting point of RL training and hence is potentially more sample efficient. A similar idea has recently been explored for control applications [108, 109].

To summarise, this chapter aims to design a tuning-free attitude estimation algorithm that maintains good performance even under inaccurate initial state estimates and/or filter gains. We have proposed a double-stage fusion architecture to correct the gyro drift leveraging both EKF and RL methods (as shown in Fig. 6.1). The proposed estimation method shows superior performance compared with the usual methods in all three scenarios: inaccurate initial state estimate, inaccurate filter gain, and even non-Gaussian noises. Experiments with real data further validate the effectiveness of the proposed method. The rest of the paper is organized as follows. Section 6.2 describes the application of EKF in attitude estimation; Section 6.3 presents the implementation of reinforcement learning in sensor fusion for attitude estimation; Section 6.4 details the RL-based correction step built upon the EKF and outlines the complete pipeline of our proposed filter; finally, Section 6.5 provides experimental results and discussion.

6.2 Extended Kalman filter for sensor fusion

In this chapter, the inertial sensors (3D gyroscopes and 3D accelerometers) combined with the magnetometer are used to estimate the attitude. The gyroscopes calculate the attitude movements by integrating the measured angular velocities. The accelerometers and magnetometers observe the local magnetic and gravity direction to estimate the sensor frame's attitude relative to the earth frame. Due to the accumulated error in the integration process and the measurement noise, a Kalman filter is widely used to fuse the separate sensor data. More details can be found in [51].

6.2.1 Orientation from angular velocity

A tri-axis gyroscope measures the angular velocity along the x, y, z axes of the sensor frame, termed $y_\omega = [y_{\omega x} \;\; y_{\omega y} \;\; y_{\omega z}]$. The dynamics of the quaternion are given as

$$\dot{q}_t^{\mathrm{nb}} = \tfrac{1}{2} q_{t-1}^{\mathrm{nb}} \otimes y_{\omega,t}, \tag{6.1}$$

where n and b stand for the navigation frame and the body frame respectively, the $\otimes$ operate denotes a quaternion product, $y_{\omega,t}$ is the angular velocity measured at time t. Accumulating rotation overtime is typically done by discretisation, i.e.,

$$\begin{aligned} q_t^{\mathrm{nb}} &= q_{t-1}^{\mathrm{nb}} + T \cdot \dot{q}_t^{\mathrm{nb}} \\ &= q_{t-1}^{\mathrm{nb}} + \tfrac{T}{2} q_{t-1}^{\mathrm{nb}} \otimes y_{\omega,t} \\ &= q_{t-1}^{\mathrm{nb}} \otimes \exp\left(\tfrac{T}{2} y_{\omega,t}\right) \end{aligned} \tag{6.2}$$

where T denotes the sampling period, $\exp(\cdot)$ corresponds to the exponential function of the quaternion. This "integration" procedure is known to be very sensitive to the measurement noise of the angular velocities.

6.2.2 Orientation from vector observations

In attitude estimation, it is typically assumed that the accelerometer only measures the gravity and a magnetometer only measures the earth's magnetic field [59]. With the direction of an earth's field known in the earth frame y^{n}, a measurement of the field's direction within the sensor frame y^{b} will allow an orientation of the sensor frame relative to the earth frame q^{nb} to be calculated. A determined direction of the earth's field can be expressed in the sensor's frame:

$$y^{\mathrm{b}} = (q^{\mathrm{nb}})^{\star} \otimes y^{\mathrm{n}} \otimes q^{\mathrm{nb}} = R^{\mathrm{nb}} \cdot y^{\mathrm{n}} \tag{6.3}$$

where * denotes conjugate of the quaternion, y^{n} is a direction vector in the earth frame that represents either the direction of gravity or that of the magnetic field. The corresponding description of y^{n} in the sensor frame is denoted as y^{b}. And R^{nb} is the rotation matrix associated with the orientation q^{nb}.

For any single measurement y^{b}, there will not be a unique orientation solution to the under-determined function $h(y^{\mathrm{b}}, y^{\mathrm{n}}, q^{\mathrm{nb}}) = 0$ (according to Eq. (6.3)). Thus, the gravity and magnetic observations are used to reference the Tilt-Pitch angle and the yaw angle,

respectively. While both the accelerometer and magnetometer measurements are often contaminated by large measurement noise under some working conditions [59], they need to be fused with another stable measurement source, e.g., a gyroscope, to achieve better performance.

6.2.3 Extended Kalman filter for attitude estimation

The EKF is widely used in attitude estimation to fuse the measurements from gyroscopes, accelerometers, and magnetometers for a single, accurate estimate of the orientation. The orientation is estimated recursively by performing a *prediction update* and a *correction update*. The *prediction update* use model (6.2) to predict the state (orientation estimation in terms of quaternion) of the next time step as follows

$$\begin{aligned} \tilde{q}^{\mathrm{nb}}_{t|t-1} &= \hat{q}^{\mathrm{nb}}_{t-1|t-1} \otimes \exp\left(\tfrac{T}{2} y_{\omega,t-1}\right), \\ P_{t|t-1} &= F_{t-1} P_{t-1|t-1} F_{t-1}^{\top} + G_{t-1} Q G_{t-1}^{\top}, \end{aligned} \tag{6.4}$$

with $Q = \Sigma_\omega$, $F_{t-1} = \left(\exp(\tfrac{T}{2} y_{\omega,t-1})\right)^R$ and $G_{t-1} = -\tfrac{T}{2}\left(\hat{q}^{\mathrm{nb}}_{t-1|t-1}\right)^L \frac{\partial \exp(e_{\omega,t-1})}{\partial e_{\omega,t-1}}$. Σ_ω is the covariance matrix of the gyroscope measurement noise, $(\cdot)^L$ and $(\cdot)^R$ are the left- and right-quaternion-product matrices respectively [110].

By using (6.3), the measurements from the accelerometer and magnetometer are fused to correct the state predictions. The *correction update* equations of the EKF are as follows:

$$\begin{aligned} \tilde{q}^{\mathrm{nb}}_{t|t} &= \tilde{q}^{\mathrm{nb}}_{t|t-1} + K_t \varepsilon_t, \\ \tilde{P}_{t|t} &= P_{t|t-1} - K_t S_t K_t^{\top}, \end{aligned} \tag{6.5}$$

where

$$\begin{aligned} \varepsilon_t &\triangleq y_t - y_{t|t-1}, \\ S_t &\triangleq H_t P_{t|t-1} H_t^{\top} + R, \\ K_t &\triangleq P_{t|t-1} H_t^{\top} S_t^{-1} \end{aligned} \tag{6.6}$$

and $y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}$, $y_{t|t-1} = \begin{pmatrix} -\tilde{R}^{\mathrm{bn}}_{t|t-1} g^{\mathrm{n}} \\ \tilde{R}^{\mathrm{bn}}_{t|t-1} m^{\mathrm{n}} \end{pmatrix}$, $H_t = \begin{pmatrix} -\left.\frac{\partial \tilde{R}^{\mathrm{bn}}_{t|t-1}}{\partial \tilde{q}^{\mathrm{nb}}_{t|t-1}}\right|_{\tilde{q}^{\mathrm{nb}}_{t|t-1}=\tilde{q}^{\mathrm{nb}}_{t|t-1}} g^{\mathrm{n}} \\ \left.\frac{\partial \tilde{R}^{\mathrm{bn}}_{t|t-1}}{\partial \tilde{q}^{\mathrm{nb}}_{t|t-1}}\right|_{\tilde{q}^{\mathrm{nb}}_{t|t-1}=\tilde{q}^{\mathrm{nb}}_{t|t-1}} m^{\mathrm{n}} \end{pmatrix}$, $R = \begin{pmatrix} \Sigma_{\mathrm{a}} & 0 \\ 0 & \Sigma_{\mathrm{m}} \end{pmatrix}$. Σ_{a} and Σ_{m} denote the covariance matrix of the measurement noise of accelerometer and magnetometor respectively, g^{n} and m^{n} are the accelerometer and magnetometor measurements at time t respectively.

However, there are a few drawbacks of the EKF and its variants. First, the linearisation error is inevitable [111], since the EKF is based on the first-order approximation of the nonlinear models (Eq. (6.1)). This means an inaccurate initial estimate $q^{\mathrm{nb}}_{0|0}$, i.e., the point of the first linearisation is undesirable. Second, the filter gain computed using inaccurate noise covariance in the EKF, needs to be tuned a priori. Third, the EKF is derived based on the assumption that the measurement noise distribution is Gaussian and time-invariant,

which are often too limited in practice. For example, the covariance of measurement noise may change abruptly over time. To address these issues, we propose a reinforcement learning (RL) approach to compensate for the estimation obtained from the EKF.

6.3 Reinforcement learning for state estimation

For orientation estimation, its system dynamics is shown in (6.5). And our goal is to design the estimator gain like the classic Kalman filter. Different from other nonlinear filtering techniques based on linearisation, we will show that the computation of the estimator gain can be solved as a RL problem.

6.3.1 System dynamics and state estimator

Reformulate (6.5) to achieve the system dynamics:

$$q_{t+1}^{\mathrm{nb}} = q_t^{\mathrm{nb}} \odot \exp_q\left(\frac{T}{2}(y_{\omega,t} - e_{\omega,t})\right), \tag{6.7}$$

where $q_t^{\mathrm{nb}} \in \mathbb{R}^4$ is the unit quaternion for the orientation of the body frame with respect to the navigation frame at time instant $t \in [0, T]$, $\exp_q(\cdot)$ corresponds to the exponential function of the quaternion, and $y_{\omega,t}$ is the gyroscope measurement. The distribution of the gyroscope noise is assumed to be Gaussian, i.e., $e_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega)$ where Σ_ω is the covariance matrix.

To estimate q_{t+1}^{nb}, the following estimator in terms of the orientation deviation is often proposed [51, 112] as illustrated in Section 6.2:

$$\hat{q}_{t+1|t}^{\mathrm{nb}} = \hat{q}_{t|t}^{\mathrm{nb}} \odot \exp_q\left(\tfrac{T}{2} y_{\omega,t}\right), \tag{6.8a}$$

$$\hat{\eta}_{t+1} = K_{t+1}(y_{t+1} - \hat{y}_{t+1|t}), \tag{6.8b}$$

$$\hat{q}_{t+1|t+1}^{\mathrm{nb}} = \exp_q(\hat{\eta}_{t+1}) \odot \hat{q}_{t+1|t}^{\mathrm{nb}} \tag{6.8c}$$

with

$$y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}, \qquad \hat{y}_{t+1|t} = \begin{pmatrix} -R\left\{\hat{q}_{t|t}^{\mathrm{nb}} \odot \exp_q(\frac{T}{2} y_{\omega,t})\right\}^\top g^n \\ R\left\{\hat{q}_{t|t}^{\mathrm{nb}} \odot \exp_q(\frac{T}{2} y_{\omega,t})\right\}^\top m^n \end{pmatrix},$$

where $\hat{q}_{t+1|t}^{\mathrm{nb}}$ is the linearisation point parametrised in terms of quaternions, $\hat{\eta}_{t+1}^{\mathrm{n}}$ is the state estimate of the orientation deviation, and $R\{\cdot\}$ denotes the matrix formula of translation from quaternion to rotation. The goal is to obtain K_{t+1}, i.e., the estimator gain at time instant $t+1$, which will be explained later in Section 6.3.2.

Define the orientation error

$$\tilde{q}_t \triangleq q_t^{\mathrm{nb}} \odot \left(\hat{q}_{t|t}^{\mathrm{nb}}\right)^{\mathrm{c}}, \tag{6.9}$$

or equivalently $q_t^{\text{nb}} = \tilde{q}_t \odot \hat{q}_{t|t}^{\text{nb}}$. From (6.7) and (6.8), we have

$$\begin{aligned}
&\tilde{q}_{t+1} = q_{t+1}^{\text{nb}} \odot \left(\hat{q}_{t+1|t+1}^{\text{nb}}\right)^{\text{c}} \\
=&\left(\left(\tilde{q}_t \odot \hat{q}_{t|t}^{\text{nb}}\right) \odot \exp_q\left(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t})\right)\right) \odot \\
&\left(\left(\exp_q\left(\tfrac{1}{2}K_{t+1}\left(y_{t+1} - \hat{y}_{t+1|t}\right)\right) \odot \hat{q}_{t|t}^{\text{nb}} \odot \exp_q(\tfrac{T}{2}y_{\omega,t})\right)\right)^{\text{c}}
\end{aligned} \tag{6.10}$$

where, $(\cdot)^c$ denotes the conjugate of quaternion.

Furthermore, to escape the unit determinant condition of the quaternion representation of rotation, the logarithm map of the quaternion is used [113]:

$$[\eta_{t+1}]_\times = \log(\tilde{q}_{t+1}) \tag{6.11}$$

where η_{t+1} is the orientation deviation and the skew operator $[\cdot]_\times$ produces the cross-product matrix.

6.3.2 Estimate error dynamics as Markov decision process

We can rewrite the residual of the estimated state, i.e., $\hat{\eta}^{\text{nb}} \in \mathbb{R}^3$, on rotation group $SO(3)$ instead of quaternion to drop the unit determinant condition, where $\hat{\eta}^{\text{nb}} = \tilde{q}^{\text{nb}} \otimes (\hat{q}^{\text{nb}})^\star$. When it is changed back to quaternions, the exponential map will be used [110]:

$$exp: \ \mathbb{R}^3 \to SO(3); \ \ q_{\hat{\eta}^{\text{nb}}} = \exp\left(\frac{\hat{\eta}^{\text{nb}}}{2}\right). \tag{6.12}$$

By combining (6.10) and (6.11), the estimate error dynamics can actually be modelled as a Markov decision process (MDP) which is defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{C}, \gamma >$:

$$\tilde{q}_{t+1} \sim \mathcal{P}(\tilde{q}_{t+1}|\tilde{q}_t, K_{t+1}), \forall t \in \mathbb{Z}_+, \tag{6.13}$$

where the estimate error $\tilde{q}_t \in \mathcal{S}$ is the state, the estimator gain $K_{t+1} \in \mathcal{A}$ is the action sampled from a stochastic policy. Considering that the ground truth q_t is known during training phase, the mapping between $\tilde{q}_t$ and $\hat{\eta}_t$ is bijective to some extent according to (6.8c) and (6.9). For convenience, in our implementation of the algorithm, we treat $\pi(K_{t+1}|\hat{\eta}_t)$ and $\pi(K_{t+1}|\tilde{q}_t)$ equivalently.

The state dynamics can be characterised by the transition probability function $\mathcal{P}(\tilde{q}_{t+1}|\tilde{q}_t, K_{t+1})$. RL algorithms can be used to find the policy π, given a cost function[1] $C(\tilde{q}_t, K_{t+1}) \in \mathcal{C}$ that measures the goodness of a state-action pair. In state estimation, it is often desired that the estimate error $\tilde{q}_t$ converges exponentially to a finite bound in mean square. As such, the cost function is selected as $C(\tilde{q}_t, K_{t+1}) = \mathbb{E}_{P(\cdot|\tilde{q}_t, K_{t+1})}[\|\tilde{q}_{t+1}\|^2]$, and the return is the sum of discounted cost $\sum_{\tau=t}^{\infty} \gamma^{\tau-t} C(\tilde{q}_t, K_{t+1})$ with the discount factor $\gamma \in [0,\ 1)$, where $\mathbb{E}[\cdot]$ denotes the expected value.

[1] We will use cost, which is often used in control literature, instead of reward.

***Definition* 6.1.** *[114] The estimate error $\tilde{q}_t$ in the MDP* (6.13) *is said to be exponentially bounded in mean square if* $\exists\ \eta > 0$ *and* $0 < \varphi < 1$*, such that*

$$\mathbb{E}[\|\tilde{q}_t\|^2] \leq \eta \mathbb{E}[\|\tilde{q}_0\|^2]\varphi^t + p, \tag{6.14}$$

holds at all the time instants $t \geq 0$, where p is a positive constant number.

In this chapter, our goal is to learn the estimator gain $K_{t+1} = \pi(\hat{\eta}_t)$ in (6.8) which can be seen as a policy obtained using an RL algorithm, such that the mean square of the estimate error of $\tilde{q}_t$ in (6.13) is guaranteed to converge exponentially to a positive bound. Different from the EKF where K_{t+1} is computed using the linearisation approximation, in this chapter K_{t+1} is approximated by a DNN $\pi(\cdot)$.

6.3.3 Estimation error boundedness guarantee

In this section, we propose the main theorem to guarantee the boundedness of the estimate error. Before proceeding, some notations need be clarified. $\rho(\tilde{q}_0)$ denotes the distribution of the starting state estimate error $\tilde{q}_0$. The state distribution of state estimate error at a certain instant t as $P(\tilde{q}_t|\rho,\pi,t)$ is defined in an iterative way: $P(\tilde{q}_{t+1} = s'|\rho,\pi,t+1) = \int_S P(\tilde{q}_t = s|\rho,\pi,t)P_\pi(s'|s)ds$. The following assumption, which is often used in RL literature, is needed:

***Assumption* 6.1.** *The Markov chain in* (6.33) *induced by a policy π is ergodic with a unique distribution probability. That is, $\exists\ p_\pi(s)$, such that*

$$p_\pi(s) = \lim_{t\to\infty} P(\tilde{q}_t = s|\rho,\pi,t) \tag{6.15}$$

***Theorem* 6.1.** *The error dynamics* (6.33) *is exponentially bounded in mean square if there exists a Lyapunov function $L(\tilde{q}_t) : S \to R^+$ and positive constants α_1, α_2 and δ such that*

$$\alpha_1 \mathbb{E}_\pi[\|\tilde{q}_t\|^2] - \delta \leq L(\tilde{q}_t) \leq \alpha_1 \mathbb{E}_\pi[\|\tilde{q}_t\|^2] \tag{6.16}$$

and

$$\begin{aligned}\lim_{N\to+\infty}[\ln(\mathbb{E}_{\tilde{q}_t\sim\mu_N}(\mathbb{E}_{\tilde{q}_{t+1}\sim P_\pi}L(\tilde{q}_{t+1})))\\ -\mathbb{E}_{\tilde{q}_t\sim\mu_N}\ln(L(\tilde{q}_t))] \leq -\alpha_2\end{aligned} \tag{6.17}$$

where

$$\mu_N(s) \triangleq \frac{1}{N}\sum_{t=0}^{N-1} P(\tilde{q}_t = s|\rho,\pi,t) \tag{6.18}$$

Proof: We have

$$\begin{aligned}
&\ln(\mathbb{E}_{\tilde{q}_t\sim\mu_N}(\mathbb{E}_{\tilde{q}_{t+1}\sim P_\pi}L(\tilde{q}_{t+1}))\\
=&\ln(\int_S\frac{1}{N}\sum_{t=0}^{N-1}P(\tilde{q}_t=s|\rho,\pi,t)\int_S P_\pi(s'|s)L(s')\,ds'\,ds)\\
=&\ln(\int_S(\int_S\frac{1}{N}\sum_{t=0}^{N-1}P(\tilde{q}_t=s|\rho,\pi,t)P_\pi(s'|s)\,ds)L(s')\,ds')\\
=&\ln(\int_S(\frac{1}{N}\sum_{t=0}^{N-1}P(\tilde{q}_{t+1}=s'|\rho,\pi,t+1))L(s')\,ds')\\
=&\ln((\frac{1}{N}\sum_{t=0}^{N-1}\int_S P(\tilde{q}_{t+1}=s'|\rho,\pi,t+1))L(s')\,ds')\\
\geq&\frac{1}{N}\sum_{t=0}^{N-1}\ln((\int_S P(\tilde{q}_{t+1}=s'|\rho,\pi,t+1))L(s')\,ds')
\end{aligned}\tag{6.19}$$

where the last inequality follows from the fact that $\ln(x)$ is a concave function on R^+. Similarly, noting that $-\ln(x)$ is a convex function we have

$$\begin{aligned}
&-\mathbb{E}_{\tilde{q}_t\sim\mu_N}\ln L(\tilde{q}_t)\\
=&-\int_S\frac{1}{N}\sum_{t=0}^{N-1}P(\tilde{q}_t=s|\rho,\pi,t)\ln(L(s))\,ds\\
=&\frac{1}{N}\sum_{t=0}^{N-1}\int_S P(\tilde{q}_t=s|\rho,\pi,t)(-\ln L(s))\,ds\\
\geq&\frac{1}{N}\sum_{t=0}^{N-1}-\ln(\int_S P(\tilde{q}_t=s|\rho,\pi,t)L(s)\,ds)
\end{aligned}\tag{6.20}$$

It follows from the above two inequalities that

$$\begin{aligned}
&\ln(\mathbb{E}_{\tilde{q}_t\sim\mu_N}(\mathbb{E}_{\tilde{q}_{t+1}\sim P_\pi}L(\tilde{q}_{t+1}))-\mathbb{E}_{\tilde{q}_t\sim\mu_N}\ln L(\tilde{q}_t))\\
\geq&\frac{1}{N}\sum_{t=0}^{N-1}\ln\frac{\int_S P(\tilde{q}_{t+1}=s'|\rho,\pi,t+1)L(s')\,ds'}{\int_S P(\tilde{q}_t=s|\rho,\pi,t)L(s)\,ds}\\
\geq&\frac{1}{N}\sum_{t=0}^{N-1}\ln\frac{\mathbb{E}_{\tilde{q}_{t+1}}L(\tilde{q}_{t+1})}{\mathbb{E}_{\tilde{q}_t}L(\tilde{q}_t)}
\end{aligned}\tag{6.21}$$

Substituting the above into (6.17), we obtain

$$\lim_{N\to+\infty}\frac{1}{N}\sum_{t=0}^{N-1}\ln\frac{\mathbb{E}_{\tilde{q}_{t+1}}L(\tilde{q}_{t+1})}{\mathbb{E}_{\tilde{q}_t}L(\tilde{q}_t)}\leq-\alpha_2 \tag{6.22}$$

then

$$\lim_{N\to+\infty}\frac{1}{N}\ln\frac{\mathbb{E}_{\tilde{q}_N}L(\tilde{q}_N)}{\mathbb{E}_{\tilde{q}_0}L(\tilde{q}_0)}\leq-\alpha_2 \tag{6.23}$$

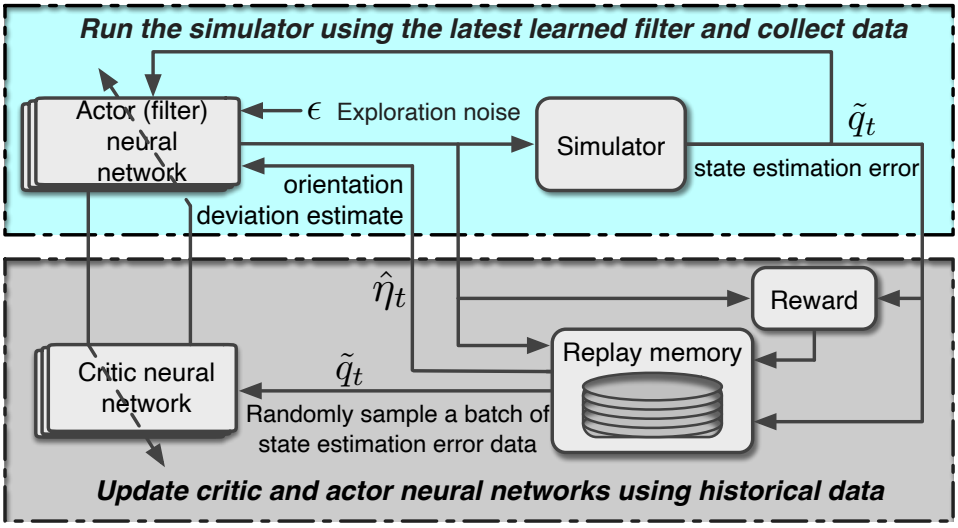


Figure 6.2: Offline RL training process of orientation state estimator.

It means that $\forall \epsilon > 0, \exists N_\epsilon$, $\frac{1}{N} \ln \frac{\mathbb{E}_{\tilde{q}_N} L(\tilde{q}_N)}{\mathbb{E}_{\tilde{q}_0} L(\tilde{q}_0)} < -\alpha_2 + \epsilon < 0$ holds when $N > N_\epsilon$, namely

$$\frac{\mathbb{E}_{\tilde{q}_N} L(\tilde{q}_N)}{\mathbb{E}_{\tilde{q}_0} L(\tilde{q}_0)} \leq \mathrm{e}^{N(-\alpha_2+\epsilon)}, \forall N > N_\epsilon \tag{6.24}$$

So we get for sufficiently large $N > N_\epsilon$,

$$\mathbb{E}_{\tilde{q}_N \sim P(\tilde{q}_N|\rho,\pi,N)} L(\tilde{q}_N) \leq \mathrm{e}^{N(-\alpha_2+\epsilon)} \mathbb{E}_{\tilde{q}_0 \sim \rho(\tilde{q}_0)} L(\tilde{q}_0) \tag{6.25}$$

By Equation (6.16) we have the following result

$$\begin{aligned} &\mathbb{E}_{\tilde{q}_N \sim P(\tilde{q}_N|\rho,\pi,N)} \mathbb{E}_\pi \|\tilde{q}_N\|^2 \\ \leq &\mathrm{e}^{N(-\alpha_2+\epsilon)} \mathbb{E}_{\tilde{q}_0 \sim \rho(\tilde{q}_0)} \mathbb{E}_\pi \|\tilde{q}_0\|^2 + \frac{\delta}{\alpha_1} \end{aligned} \tag{6.26}$$

So far, it has been proved that the estimate error $\tilde{q}_t$ in (6.33) is exponentially bounded according to Definition 1.

6.3.4 Lyapunov-based reinforcement learning orientation estimation algorithm

In this section, we will combine SAC algorithm [65], one of the state-of-the-art RL algorithms, with the theoretical result in Section 6.3.3 to learn the gain/policy K_{t+1} for the state estimator (6.8).

Considering MDP in (6.33), the orientation estimation problem can be viewed as a RL problem in which the policy is sought after by minimising the expected accumulated cost.

Here a stochastic policy is chosen as $\pi(K_{t+1} \mid \tilde{q}_t) \sim \mathcal{N}(K_{t+1}(\tilde{q}_t), \sigma)$ from which the gain K_{t+1} for a given state $\tilde{q}_t$ is sampled [115]. The corresponding Q-function (a.k.a, state-action value function) is given as:

$$Q_\pi(\tilde{q}_t, K_{t+1}) = C_t(\tilde{q}_t, K_{t+1}) + \gamma \mathbb{E}_{\tilde{q}_{t+1}}[V_\pi(\tilde{q}_{t+1})] \tag{6.27}$$

To this end, K_{t+1} can be learned by many existing RL algorithms.

Motivated by the works in [68, 116, 117], we propose to incorporate the theoretical result in Theorem 6.1 to policy optimization in SAC as a constrained optimization problem. First of all, a Lyapunov candidate needs to be selected at the first instance. In the context of RL, a Lyapunov candidate will be parametrized/selected as the Q-function [67, 118]. In this chapter, we choose $L(\tilde{q}_t)$ in (6.16) as:

$$L_(\tilde{q}_t) = \mathbb{E}_{K_{t+1}\sim\pi}[L_c(\tilde{q}_t, K_{t+1})] \tag{6.28}$$

where $L_c(\tilde{q}_t, K_{t+1}) = Q(\tilde{q}_t, K_{t+1})$. Then the constrained optimisation problem is:

$$\begin{aligned} \min_{\pi} \quad & Q_\pi(\tilde{q}_t, K_{t+1}) \\ \text{s.t.} \quad & (6.16) \text{ and } (6.17) \\ & -\ln(\pi(K_{t+1} \mid \tilde{q}_t)) \geq \mathcal{H}_t \end{aligned} \tag{6.29}$$

where $Q_\pi(\tilde{q}_t, K_{t+1})$ is defined in (6.27), the second constraint is the minimum entropy constraint used in the SAC to improve the exploration in the action space [119] where $\mathcal{H}_t$ is the desired bound.

Denote the parametrised actor and critic as $\pi_\theta(K_{t+1}|\tilde{q}_t)$ and $Q_\phi(\tilde{q}_t, K_{t+1})$ respectively, where θ and ϕ are the parameters of the DNNs. To ensure the positiveness of a Lyapunov function, $L_\phi(\tilde{q}_t, K_{t+1})$ is selected as the square of a DNN as $L_\phi(\tilde{q}_t, K_{t+1}) = f_\phi^\top(\tilde{q}_t, K_{t+1}) f_\phi(\tilde{q}_t, K_{t+1})$, where f is the vector output of a DNN parameterised by ϕ. On the other hand, the stochastic policy $\pi_\theta(K_{t+1}|\tilde{q}_t)$ is parametrised by a DNN f_θ that depends on the state $\tilde{q}_t$ and a Gaussian noise ϵ.

Solving the above constrained optimisation problem is equivalent to minimising the following objective function:

$$\begin{aligned} J(\theta) = \mathbb{E}_{\tilde{q}_t, a_t, \tilde{q}_{t+1}, c_t \sim D}[\alpha(\ln(\pi_\theta(f_\theta(\tilde{q}_t, \epsilon)|\tilde{q}_t)) + \mathcal{H}_t) \\ + \lambda\big(\ln L_\phi(\tilde{q}_{t+1}, f_\theta(\tilde{q}_{t+1}, \epsilon)) - \ln L(\tilde{q}_t, a_t) + \alpha_2\big)] \end{aligned} \tag{6.30}$$

where $\mathcal{D}$ is the replay memory of the training samples, α and λ are Lagrange multipliers which control the relative importance of constraints in (6.29).

In the actor-critic framework, the parameters of policy network are updated through stochastic gradient descent of (6.30). The training process can be seen in Fig. 6.2. It can be proved that the policy can converge to an optimal one that ensures the orientation estimate error $\mathbb{E}[\|\tilde{q}_t\|^2]$ converges to a constant as $t \to \infty, \forall \tilde{q}_t \in S$. Pseudo code of the proposed Lyapunov-based reinforcement learning orientation estimation (LRLOE) algorithm is showed in Algorithm 1.

Algorithm 1 LRLOE algorithm

1: Set the initial parameters ϕ for the Lyapunov function $\mathcal{L}_\phi$, θ for the estimator gain policy $\pi_\theta(K_1|\tilde{q}_0)$, λ for the Lagrangian multiplier, α for the temperature parameter, and the replay memory $\mathcal{D}$
2: Set the target parameter $\bar{\theta}$ as $\bar{\theta} \leftarrow \theta$
3: **while** Training **do**
4: **for** each data collection step **do**
5: Choose estimator gain K_{k+1} using $\pi_\theta(K_{k+1}|\tilde{q}_k)$
6: Simulate (6.7) and (6.3) with the orientation estimator (6.8) to collect samples $\tilde{q}_k$
7: $\mathcal{D} \leftarrow \mathcal{D} \cup \tilde{q}_k$
8: **end for**
9: update L_ϕ, π_θ, λ, α by optimising (6.30)
10: **end while**
11: Output θ^*, ϕ^*, λ^*, and α^*

6.4 RL compensated EKF algorithm

6

Different from using reinforcement learning to learn the estimator gain directly, we propose to use RL to compensate the estimate residual in the EKF. Comparing to the direct RL approach, the proposed RL compensated EKF (RLC-EKF) algorithm has the advantage of being more interpretable and easier to implement.

6.4.1 Problem formulation

We first rewrite the residual of the EKF, i.e., $\hat{\eta}^{\text{nb}} \in \mathbb{R}^3$, on rotation group $SO(3)$ instead of quaternion to drop the unit determinant condition, where $\hat{\eta}^{\text{nb}} = \tilde{q}^{\text{nb}} \otimes (\hat{q}^{\text{nb}})^\star$. When it is changed back to quaternions, the exponential map in Eq. 6.12 will be used.

The high-level plan can be illustrated in Fig. 6.4. In the RL correction module, the difference between the measurements y_t from sensors and the estimated observation $\tilde{y}_{t|t} = h(\tilde{q}^{\text{nb}}_{t|t}) = \begin{pmatrix} -\tilde{R}^{\text{bn}}_{t|t} g^{\text{n}} \\ \tilde{R}^{\text{bn}}_{t|t} m^{\text{n}} \end{pmatrix}$ from the EKF where $h(\cdot)$ is a compact form of (6.3), $\varepsilon^{\text{RL}}_t = y_t - \tilde{y}_{t|t}$ is used to obtain the estimate residual $\hat{\eta}^{\text{nb,RL}}_t$ by computing a gain U_t, in order to get the improved estimate $\hat{q}^{\text{nb}}_{t|t}$. The estimate residual $\hat{\eta}^{\text{nb,RL}}_t$ computed using RL algorithms plays a role of compensating the estimate $\tilde{q}^{\text{nb}}_{t|t}$ obtained from the EKF.

Then we have the following equations:

$$\begin{aligned} \hat{\eta}^{\text{nb,RL}}_t &= U_t \varepsilon^{\text{RL}}_t \\ \hat{q}^{\text{nb}}_{t|t} &= \exp\left(\frac{\hat{\eta}^{\text{nb,RL}}_t}{2}\right) \otimes \tilde{q}^{\text{nb}}_{t|t} \end{aligned} \tag{6.31}$$

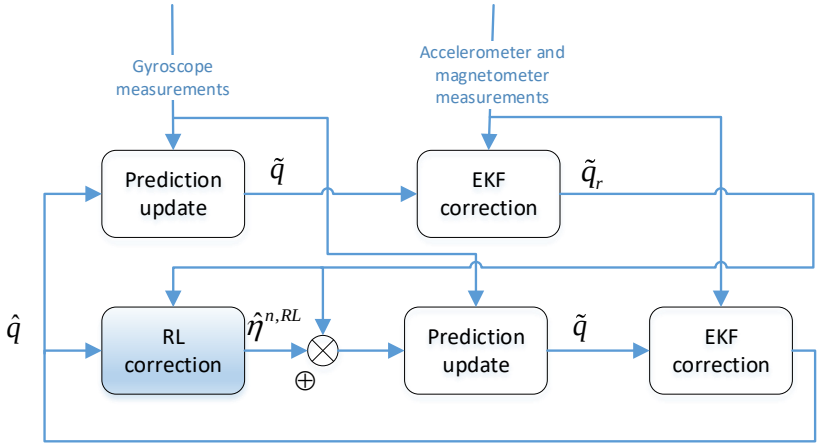


Figure 6.3: Schematic illustration of RLC-EKF algorithm.

Based on the filter process in (6.31), we have

$$\begin{aligned}\hat{\eta}_t^{\mathrm{nb,RL}} &= 2\log\left(\hat{q}_{t|t}^{\mathrm{nb}} \otimes (\tilde{q}_{t|t}^{\mathrm{nb}})^{\star}\right) \\ &= 2\log\left((\exp(\tfrac{U_t \varepsilon_t^{\mathrm{RL}}}{2}) \otimes \tilde{q}_{t|t}^{\mathrm{nb}}) \otimes (\tilde{q}_{t|t}^{\mathrm{nb}})^{\star}\right)\end{aligned} \tag{6.32}$$

where $\tilde{q}_{t|t}^{\mathrm{nb}} = f(\hat{q}_{t-1|t-1}^{\mathrm{nb}}) = \exp\left(\frac{\hat{\eta}_{t-1}^{\mathrm{nb,RL}}}{2}\right) \otimes \tilde{q}_{t-1|t-1}^{\mathrm{nb}}$, and $f(\cdot)$ denotes the estimation correction step in the EKF.

The dynamics of the residual $\hat{\eta}_t^{\mathrm{nb,RL}}$ can be characterised as an MDP, in which the stochasticity essentially comes from the sensor noise:

$$\hat{\eta}_t^{\mathrm{nb,RL}} \sim \mathcal{P}\left(\hat{\eta}_t^{\mathrm{nb,RL}} | \hat{\eta}_{t-1}^{\mathrm{nb,RL}}, U_t\right), \forall t \in \mathbb{Z}_+, \tag{6.33}$$

where the estimate residual $\hat{\eta}_t^{\mathrm{nb,RL}} \in \mathcal{S}$ is the state, the estimated gain $U_t \in \mathcal{A}$ is the action sampled from the trained policy, $\mathcal{P}(\hat{\eta}_t^{\mathrm{nb,RL}} | \hat{\eta}_{t-1}^{\mathrm{nb,RL}}, U_t)$ indicated the transition probability function of the estimation.

To this end, the attitude estimation problem can be formulated as an RL problem, i.e., learn a policy U_t to control the estimate residual $\hat{\eta}_t^{\mathrm{nb,RL}}$.

6.4.2 RL compensated EKF algorithm

In the RL problem, a cost function[2] $C(\hat{\eta}_{t-1}^{\mathrm{nb,RL}}, U_t) \in C$ will be used to measure the goodness of a state-action pair, i.e.,

$$C(\hat{\eta}_{t-1}^{\mathrm{nb,RL}}, U_t) = \mathbb{E}_{P(\cdot|\hat{\eta}_{t-1}^{\mathrm{nb,RL}}, U_t)}[\|\hat{\eta}_t^{\mathrm{nb,RL}}\|^2] \tag{6.34}$$

[2]We use cost function instead of reward function.

Algorithm 2 RL Compensated EKF (RLC-EKF) Algorithm for Orientation Estimation

INPUTS: Initial data $\{y_{w,t}, y_{a,t}\}_{t=1}^{N}$, magnetometer data $\{y_{m,t}\}$, and the initial estimation of the orientation $q_{0|0}^{\text{nb}}$,

OUTPUTS: Improved estimation of the orientation $\hat{q}_{t|t}^{\text{nb}}$.

1: **for** t=1,2,3...N **do**

Prediction Update

2: Apply (6.4) with angular velocity.

EKF Correction:

3: Correction update using $y_{a,t}$ and $y_{m,t}$ as in (6.5)

Quaternion Normalisation:

4: Normalise the quaternion and its covariance as $\tilde{q}_{t|t}^{\text{nb}} = \frac{\tilde{q}_{t|t}^{\text{nb}}}{\|\tilde{q}_{t|t}^{\text{nb}}\|_2}$, $\tilde{P}_{t|t}^{\text{EKF}} = J_t \tilde{P}_{t|t} J_t^{\top}$ with $J_t = \frac{1}{\|\tilde{q}_{t|t}^{\text{nb}}\|_2^3} \tilde{q}_{t|t}^{\text{nb}} \left(\tilde{q}_{t|t}^{\text{nb}}\right)^{\top}$.

RL Correction - estimate the residual in estimation:

5: Compute the residual $\hat{\eta}_t^{\text{nb,RL}}$ and gain U_t using Algorithm 3, where $\hat{\eta}_t^{\text{nb,RL}} = U_t \varepsilon_t^{\text{RL}},, U_t = \pi(\hat{\eta}_{t-1}^{\text{nb,RL}}), \varepsilon_t^{\text{RL}} = y_t - y_{t|t}$, $y_t = \begin{pmatrix} y_{\text{a},t} \\ y_{\text{m},t} \end{pmatrix}$, and $y_{t|t} = \begin{pmatrix} -\tilde{R}_{t|t}^{\text{bn}} g^{\text{n}} \\ \tilde{R}_{t|t}^{\text{bn}} m^{\text{n}} \end{pmatrix}$.

RL Correction - inject the estimate residual:

6: Inject the error into the quaternion state and adapt the covariance matrix $\hat{q}_{t|t}^{\text{nb}} = \exp\left(\frac{\hat{\eta}_t^{\text{nb,RL}}}{2}\right) \odot \tilde{q}_{t|t}^{\text{nb}}$ and $\hat{P}_{t|t} = M_t \tilde{P}_{t|t} M_t^{\top}$ with $M_t = \left(\exp\left(\frac{\hat{\eta}_t^{\text{nb,RL}}}{2}\right)\right)^{L}$.

7: **end for**

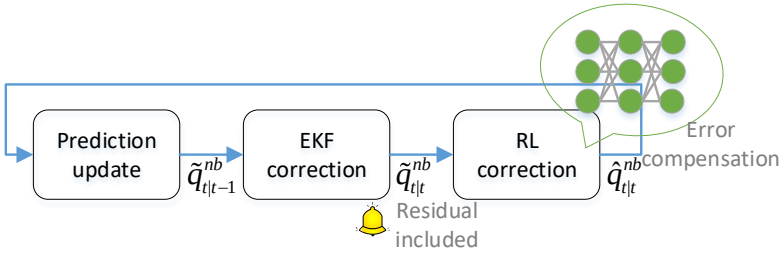


Figure 6.4: RL compensated EKF workflow. First, gyroscope measurements are used to predict the attitude using (6.4)*. Then an EKF correction is performed to update the prediction using* (6.5)*. Last, a RL policy is further used to compensate the residuals using* (6.31)*.*

and the return is the sum of discounted cost $\sum_{\tau=t}^{\infty} \gamma^{\tau-t} C(\hat{\eta}^{\text{nb,RL}}_{t-1}, U_t)$ with the discount factor $\gamma \in [0,\ 1)$. Our aim is to learn the estimator gain $U_t = \pi(\hat{\eta}^{\text{nb,RL}}_{t-1})$ in (6.31) as a policy using RL algorithm. And the policy will be learned/approximated by using a deep neural network. This means any state-of-the-art deep RL algorithms can be readily applied.

The full pipeline is illustrated in Fig. 6.3. The proposed RL compensated EKF (RLC-EKF) algorithm is summarised in Algorithm 2. It should be noted that the superscript $\tilde{\cdot}$ (not $\hat{\cdot}$) indicates the estimate is intermediate and still needs to be updated. More reliable estimation of the orientation can be obtained after the two correction steps, i.e., EKF and RL Correction in Algorithm 2, as depicted in Fig. 6.4.

6.4.3 Convergence of estimate error

In this attitude estimation task, it is desired that the estimate error $\hat{\eta}^{\text{nb,RL}}_{t-1}$ converges to a real number as small as possible. The vanilla EKF should be stable to ensure the convergence of the attitude estimation. After introducing the reinforcement learning compensation structure, the final estimate of the proposed filter is expected to be better than the vanilla EKF and should not deteriorate the convergence of EKF. Inspired by the work [108, 116], based on EKF, we will show that the mean square of the estimate residual in the RL-compensated filter is guaranteed to converge within a positive bound.

The value function in reinforcement learning can be written as

$$\mathcal{V}^i\left(\hat{\eta}^{\text{nb,RL}}_t\right) = -\sum_{\tau=t}^{\infty} \gamma^{\tau-t} C^i(\hat{\eta}^{\text{nb,RL}}_{t-1}, U^i_t).$$

In the presented design, the RL mechanism is introduced to further improve the performance of EKF. Hence, it is reasonable to assume that $\mathcal{V}^i\left(\hat{\eta}^{\text{nb,RL}}_t\right)$ is a continuously differentiable function with

$$\mathcal{V}^i\left(\hat{\eta}^{\text{nb,RL}}_{t+1}\right) - \mathcal{V}^i\left(\hat{\eta}^{\text{nb,RL}}_t\right) \leq -\mathcal{W}\left(\hat{\eta}^{\text{nb,RL}}_t\right) + \mu^i$$

where $\mathcal{W}\left(\hat{\eta}^{\text{nb,RL}}_t\right)$ is a continuous positive definite function, $\mu^i > 0$, and i denotes the i-th iteration of the RL algorithm. There exists $\mathcal{W}\left(\hat{\eta}^{\text{nb,RL}}_t\right) > \mu$. When $i = 0$, it corresponds to

Algorithm 3 RL algorithm for Step 5 in Algorithm 2

1: Initialise parameters θ for the critic $\mathcal{V}_\theta\left(\hat{\eta}_t^{\text{nb,RL}}\right)$, and ϕ for the actor network. The initial value of U_t^0 is from the vanila EKF module. Initialise the replay memory $\mathcal{D} \leftarrow \emptyset$.
2: Assign initial values to the critic parameter $\theta \leftarrow \theta^0$ and its target $\bar{\theta} \leftarrow \theta^0$
3: **for** Data collection steps **do**
4: Choose an action U_t sampled from $\pi(\hat{\eta}_t^{\text{nb,RL}})$
5: Run the simulator and EKF module & collect data
6: **end for**
7: **for** each gradient update step **do**
8: Sample a batch of data $\mathcal{B}$ from $\mathcal{D}$
9: $\theta \leftarrow \theta - l_V \nabla_\theta J_V(\theta)$
10: $\phi \leftarrow \phi - l_\pi \nabla_\phi J_{\boldsymbol{\pi}}(\phi)$
11: $\bar{\theta} \leftarrow \kappa\theta + (1-\kappa)\bar{\theta}$
12: **end for**

the vanilla EKF scenario, as the RL policy has negligible initial values around zeros. Next, let's approve it also holds in the later iterations.

Let U_t^i be the estimated gain of the i-th iteration of RL algorithm. Since $\mathcal{V}^i\left(\hat{\eta}_t^{\text{nb,RL}}\right) = -C^i(\hat{\eta}_t^{\text{nb,RL}}, U_t^i) + \gamma\mathcal{V}^i\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right)$, we have

$$(1-\gamma)\mathcal{V}^i\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right) \leq -\mathcal{W}\left(\hat{\eta}_t^{\text{nb,RL}}\right) + \mu^i - C^i(\hat{\eta}_t^{\text{nb,RL}}, U_t^i)$$

In the policy evaluation, the following Bellman backup operation is repeatedly conducted.

$$\mathcal{V}^{i+1}\left(\hat{\eta}_t^{\text{nb,RL}}\right) = -C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1}) + \gamma\mathcal{V}^i\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right)$$

In the policy improvement, U_t is updated to minimise the discounted accumulated cost, so

$$\begin{aligned}
&\mathcal{V}^{i+1}\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right) - \mathcal{V}^{i+1}\left(\hat{\eta}_t^{\text{nb,RL}}\right) \\
&= C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1}) + (1-\gamma)\mathcal{V}^{i+1}\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right) \\
&\leq C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1}) + (1-\gamma)\mathcal{V}^i\left(\hat{\eta}_{t+1}^{\text{nb,RL}}\right) \\
&\leq -\mathcal{W}\left(\hat{\eta}_t^{\text{nb,RL}}\right) + \mu^i - C^i(\hat{\eta}_t^{\text{nb,RL}}, U_t^i) + C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1})
\end{aligned}$$

Hence, the RL algorithm will ensure stable performance for all iterations. At each iteration of the policy improvement, the discounted accumulated cost will be reduced, so $C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1}) \leq C^i(\hat{\eta}_t^{\text{nb,RL}}, U_t^i)$ and $\mu^i - C^i(\hat{\eta}_t^{\text{nb,RL}}, U_t^i) + C^{i+1}(\hat{\eta}_t^{\text{nb,RL}}, U_t^{i+1}) \leq \mu^i$. It implies that each iteration will potentially reduce the ultimate bound of $\hat{\eta}_t^{\text{nb,RL}}$. In the implementation, accumulated discounted cost will be approximated by a multiple layer perceptron (MLP) with parameters denoted by θ. The actor network parameters are denoted by ϕ. The training process of the RL module in Step 5 of Algorithm 2 is summarised in Algorithm 3, in which J_V and J_π denote the optimisation objective for the critic and actor objectives [108, 116].

6.5 Experimental results

This section will evaluate the proposed algorithm (RLC-EKF) on both simulated and real data. We compare with the EKF [51], the CF [59] and the RLF [107]. We evaluate the performance in three scenarios: (1) inaccurate initial state estimate, (2) inaccurate filter gain, (3) inaccurate noise model. Before going into the details, a quick summary of the feature of the algorithms can be found in Table. 6.1

Table 6.1: Features of RLC-EKF compared with other methods.

Applicability	EKF	CF	RLF	RLC-EKF
Inaccurate initial estimation	×	√	×	√
Inaccurate filter gain	×	×	√	√
Inaccurate noise model	×	√	√	√

A relatively trivial pattern of the angular velocity (see Fig.2 in our previous work [107]) is used as the training profile. The angular velocity profile remains the same in each training episode, while the initial state q^{nb} and the initial estimation of the state $\hat{q}^{\text{nb}}_{0|0}$ are randomly sampled with a uniform distribution $\mathbb{U}([-1,-1,-1,-1],[1,1,1,1])$. The sampling rate is 100 Hz (consistent with that for real data in Section 6.5.2). During training, the sensor noise is sampled with the following distribution [51]: $e_{\omega,t} \sim \mathcal{N}(0,\Sigma_\omega), \Sigma_\omega = 0.0003 I_{3\times3}$, $e_{\text{a},t} \sim \mathcal{N}(0,\Sigma_{\text{a}}), \Sigma_{\text{a}} = 0.0005 I_{3\times3}$, $e_{\text{m},t} \sim \mathcal{N}(0,\Sigma_{\text{m}}), \Sigma_{\text{m}} = 0.0003 I_{3\times3}$. We independently train 20 policies and select the one with the lowest validate error for inference on unknown profiles. The training of the RL policy in RLC-EKF used PPO2 [120] for Algorithm 3 in Section. 6.3. The hyperparameters are the same as our previous work (see Table.1 in [107]).

6.5.1 Results for simulated data

Scenario 1: Inaccurate initial estimation

We compare our algorithm with EKF, CF and RLF when the initial estimate $\hat{q}^{\text{nb}}_{0|0}$ is inaccurate and randomly sampled with a uniform distribution $\hat{q}^{\text{nb}}_{0|0} \sim \mathbb{U}([-1,-1,-1,-1],[1,1,1,1])$. The estimation is expected to converge to the true state as quickly as possible. The hyperparameters for EKF and CF are directly adapted from [51, 59]. The CF gain β is set as 0.041. The adjustable measurement covariance in EKF is chosen as the true sensor covariance. The results of the estimation performance are shown in Fig. 6.5. The results indicate that our algorithm can quickly converge to the true state compared with EKF and CF.

Scenario 2: Inaccurate filter gain

The performance of the filters largely depends on their filter gain. In CF it is the adjustable parameter β. In the EKF, since $K_t \triangleq P_{t|t-1}H_t^\top(H_t P_{t|t-1} H_t^\top + R)^{-1}$ where the covariance of measurement noise R needs to be specified/tuned in the beginning, the gain K_t is determined by R. Experimental results of EKF and CF with different filter gains can be found in Fig. 6.6. With different filter gains, their estimation performance varies a lot. It significantly stresses

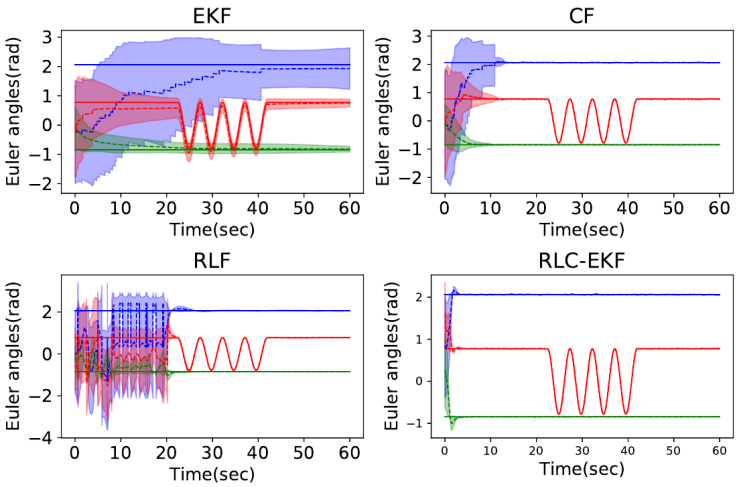


Figure 6.5: The attitude estimation performance of EKF, CF, RLF and RLC-EKF when the initial estimation is randomly selected. The solid and dashed lines respectively represent the ground truth and the average estimate. The shaded areas indicate the standard deviation over 50 independent runs.

the importance of parameter tuning.

Scenario 3: Inaccurate noise model

In this scenario, we test our algorithm on measurements with an inaccurate noise model. An additional constant bias of 0.02 rad/s is added to the simulated gyroscope measurements. The experimental result is shown in Figure. 6.7. Our proposed RLC-EKF method has shown good performance regardless of the other bias. Simultaneously, the extra unexpected noise has introduced a constant bias in the estimation of EKF. It is because such Kalman filter-based methods are derived based on the assumption that the measurement noise has a known covariance model and cannot deal with the case with an inaccurate noise model. After being compensated by an RL policy, this constraint will not exist anymore.

6.5.2 Results for real data

Finally, we test the proposed algorithm on a real-world dataset. The data is collected from the Trivisio Colibri Wireless IMU with a logging rate of 100Hz. The reference measurement of the orientation is provided as ground truth from motion capture equipment by tracking the optical markers fixed to the sensor platform. The optical and IMU data have been time-synchronised and aligned beforehand.

The dataset is 100 seconds long and split into training and inference datasets separately. The first half of the collected data is used for training and the rest for inference. We randomly selected a consecutive sequence of a length of 1000 samples as a training episode in the

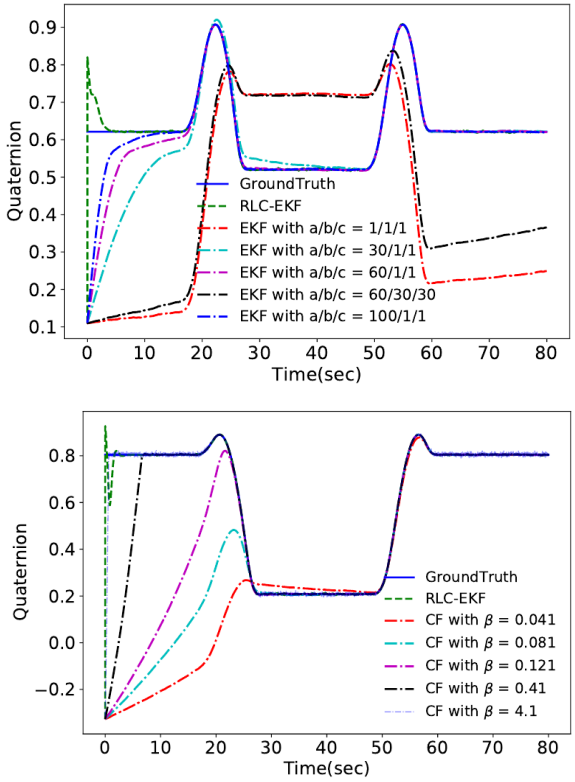


Figure 6.6: The top figure shows the experimental results of EKF with different covariance matrices. We set the estimated measurement covariance of the gyroscope, accelerometer, and magnetometer as their real covariance, respectively times a, b and c. The bottom figure shows the experimental results of the CF with different initial filter gain, β. We set β as 0.041, 0.081, 0.121, 0.41 and 4.1 respectively.

training dataset. The test results are shown in Fig. 6.9. With an uncertain starting point, all the filters successfully converged to the true state and have shown similar estimation performance after convergence. Here we compare the filters' performance on the second half of the trajectory. The results can be found in Table 6.2.

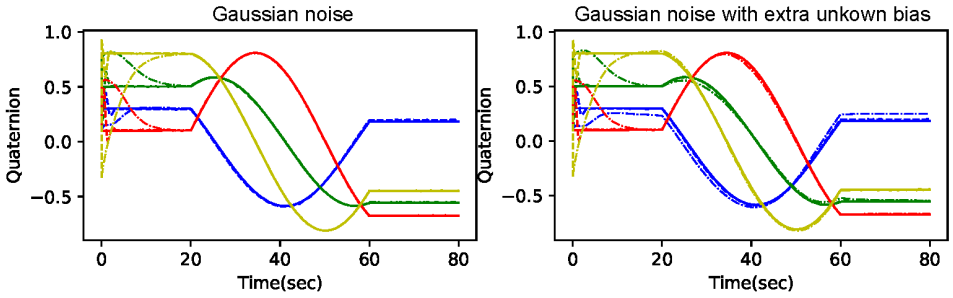


Figure 6.7: The attitude estimation performance of RLC-EKF and EKF on measurements with and without an accurate noise model. One only includes a known Gaussian noise. The other includes a known Gaussian noise with unknown extra bias. The solid, dashed, and dash-dot lines represent the ground truth, the RLC-EKF estimation, and the EKF estimation.

Table 6.2: RMSE of the orientation estimates

RMSE	Yaw$[rad]$	Pitch$[rad]$	Roll$[rad]$
RLC-EKF	0.241	0.020	0.038
EKF	0.186	0.175	0.041
CF	0.260	0.019	0.041

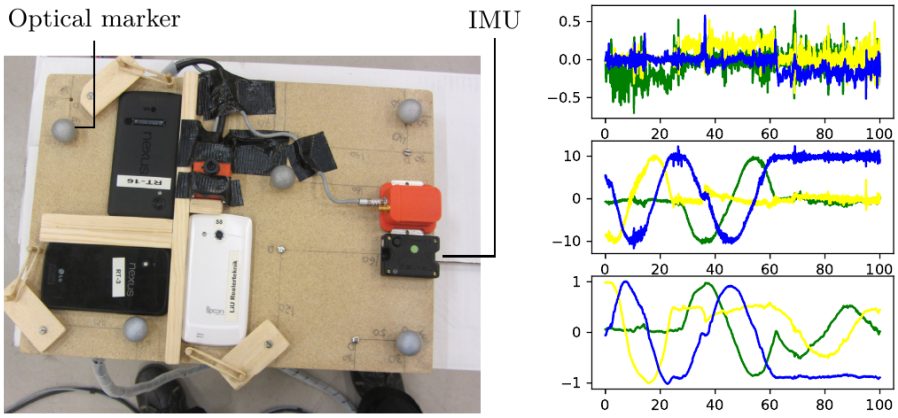


Figure 6.8: Real dataset (adapted from Fig. 4.2 and 4.3 in [51]). Left: A snapshot of the platform for collecting real dataset Right: Measurements from an accelerometer ($y_{a,t}$, top), a gyroscope ($y_{\omega,t}$, middle) and a magnetometer ($y_{m,t}$, bottom) for 100 seconds of data collected with the IMU shown in the left figure.).

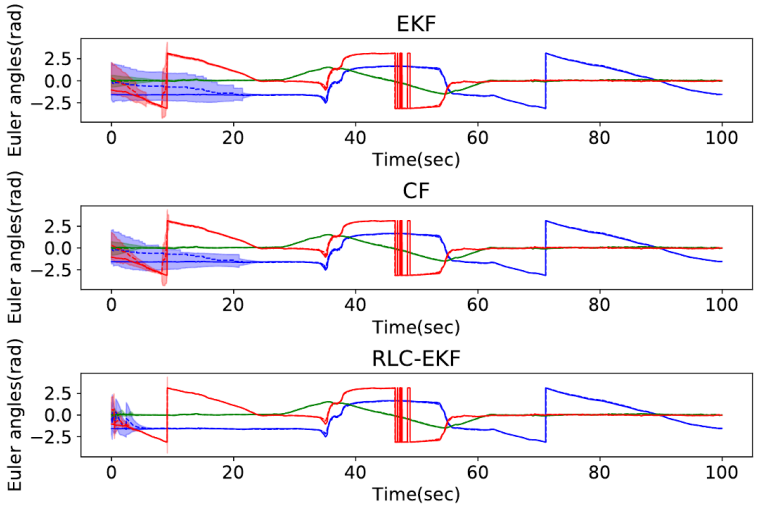


Figure 6.9: Estimated Euler angles for real data. The attitude estimation is expressed in a more intuitive manner with Euler angles instead of quaternions. The solid and dashed lines correspond to the ground truth and the mean estimation respectively. The shaded areas indicate the standard deviation over 50 independent runs.

6.6 Conclusions

Orientation estimation using inertial sensors combined with a magnetometer is well-studied, and many algorithms have been proposed. However, there hardly exist any algorithms with theoretical guarantees of estimation convergence. In this chapter, we propose a reinforcement learning based orientation estimation method and prove that its estimate error converges to a positive scalar in mean square with a guarantee. The proposed method shows superior estimation performance compared with some well-known ones in terms of arbitrary estimation initialization and adaptation to very large angular velocities.

What's more, we introduced a novel attitude estimation method by combining the classic EKF with a deep reinforcement learning algorithm. The proposed algorithm has a faster convergence speed than the pure RL approach, and it shows the advantage that it is insensitive to (1) inaccurate initial estimate, (2) inaccurate initial gain, and (3) inaccurate noise model. The effectiveness of the proposed approach is demonstrated on both simulated and real datasets.

7

Conclusions and future work

7.1 Conclusions

This thesis explores the challenge of delivering unwieldy objects using mobile robots, addressing the problem from two main perspectives.

In Chapter 3, we propose a stable manipulation approach that emphasizes maintaining a stiff contact between the robot and the object during manipulation. This fixed contact reduces uncertainties during manipulation and simplifies planning. We derive a concise linear motion constraint for stable pushing with differential-drive mobile robots, which allows us to formulate the pushing planning problem as a constrained optimization problem using nonlinear model predictive control (NMPC). This approach is easily implementable within existing robot navigation frameworks and achieves a high success rate in real-world applications. Our experiments demonstrate that the NMPC-based planner outperforms a reactive pushing strategy in terms of efficiency, reducing the robot's traveled distance by 23.8% and time by 77.4%. Furthermore, our method requires four fewer hyperparameters and decision variables than the Linear Time-Varying (LTV) MPC approach, making it easier to implement. We validate the proposed method through real-world experiments with two differential-drive robots, Husky and Boxer, under various friction conditions.

However, the stable pushing approach also limits the range of forces that can be transmitted, thereby restricting the system's maneuverability. To overcome this limitation, Chapter 4 introduces a free pushing method. This approach allows the robot to maneuver freely around the object, enabling a wider range of pushing forces. We model the robot-object contact as a sliding joint, which smooths transitions across different contact points. Additionally, to ensure the feasibility of the planned pushes, we develop a robot-object contact model that accounts for the shape and kinematics of the robot in pushing modeling and planning. Finally, a Model Predictive Controller solves the pushing planning problem in real time. Experimental results show that the proposed method achieves an average success rate of 83% with an accuracy of 0.085m when pushing to the selected goals. Compared to the stable pushing method, this approach improves the agility and efficiency of mobile pushers and demonstrates robustness in achieving the task while tolerating modeling errors.

Despite these advancements, challenges persist. The underactuated nature of non-prehensile manipulation systems can lead to the controller becoming trapped in local optima, and the high nonlinearity complicates planning. Moreover, a significant challenge in planning and controlling for nonprehensile manipulation lies in the discontinuity of system dynamics. The first two chapters focus on modeling the manipulation system continuously. However, uncertainties during contact sometimes lead to unexpected contact loss, requiring a contact-implicit planner capable of reacting to these unforeseen circumstances. To address this, we propose a holistic approach in Chapter 5 by formulating the initial contact point as a decision variable, enabling seamless integration of reaching and pushing tasks within a unified trajectory optimization framework. Additionally, we prove that the underactuated pushing system is differentially flat. By exploiting this property, we simplify the pushing planning problem into a trajectory optimization problem for the object, addressing the challenges posed by the system's high nonlinearity. The entire manipulation

plan, including reaching and manipulating, only takes an average of 2 milliseconds to solve. With the benefit of fast calculations, our method offers robustness, ensuring quick recovery from disturbances or uncertainties during manipulation.

Readers may wonder why we shifted focus from nonprehensile manipulation planning and control to state estimation research in Chapter 6. In fact, the work presented in Chapter 6 was conducted during the first year of my PhD, when I was still exploring various research interests. State estimation is a fundamental topic in robotics, underpinning both control and planning processes.

Leveraging advancements in deep learning technologies, we demonstrated that state estimators can be trained in an unsupervised manner, significantly reducing the need for laborious human intervention. Our approach began with the introduction of a deep reinforcement learning algorithm for orientation estimation, where we proved the convergence of the estimate error for the learned policy. Rather than training the policy from scratch, we then employed reinforcement learning to enhance the classical extended Kalman filter estimation. The proposed approach is able to learn the filter gain directly from sensor measurements, demonstrating superior performance compared to conventional methods, particularly in challenging scenarios such as inaccurate initial state estimates, imprecise filter gains, and even non-Gaussian noise environments.

This work, while distinct from our later focus on nonprehensile manipulation, contributes valuable insights on the potential of machine learning techniques in improving fundamental estimation processes. There are also many planning and control approaches based on reinforcement learning. However, when I surveyed the literature, I found that purely learning-based approaches for planning and control often require excessive data and are less efficient than classic model-based methods. Inspired by the pure Reinforcement Learning state estimator and the RL-compensated EKF in Chapter 6, I believe that a hybrid approach combining model-based and learning-based methods could lead to more efficient manipulation policy learning, balancing the data efficiency of model-based methods with the adaptability of learning-based approaches.

Recognizing the wealth of existing scientific knowledge about physical models, I chose to focus on model-based approaches for nonprehensile manipulation in my PhD. My goal is to first have a better understanding of physics while addressing planning and control for manipulation. After that, I aim to combine model-based and learning-based methods in manipulation planning and control. This is left as my future work.

7.2 Future work

Two research topics are explored in this thesis: planning and control for unwieldy object delivery, with a focus on the usage of nonholonomic mobile robots, and state estimation with a focus on the usage of deep reinforcement learning. Based on the experience gained from these two topics, I propose the following future research directions:

Physics accelerated policy learning for robot manipulation

As discussed in Chapter 7.1, deep learning-based approaches such as imitation learning and reinforcement learning often require vast amounts of data, which significantly limits their efficiency. Rather than learning from scratch, a more promising approach is to develop algorithms that combine the data efficiency of model-based methods with the adaptability of model-free reinforcement learning.

For instance, recent works like [121] and [122] leverage differentiable physics models to accelerate the learning process. Differentiable simulation offers the promise of faster convergence and more stable training by computing low-variance first-order gradients using the robot model. Furthermore, developing meta-learning algorithms that can quickly adapt to new manipulation tasks by leveraging learned physical principles presents an exciting avenue for research. This approach could significantly enhance the generalizability and efficiency of learning-based manipulation systems.

By integrating physics-based models with learning algorithms, we can potentially overcome the data efficiency limitations of pure learning-based approaches while maintaining their flexibility and adaptability to complex, real-world scenarios.

System identification for contact-based manipulation

While significant progress has been made in model-based control for robotics (as shown in Chapters 3, 4, and 5), accurately modeling contact dynamics remains a formidable challenge, particularly in scenarios involving rich, complex interactions. No model can perfectly capture the intricacies of real-world physics, especially at the interface of contact.

A promising direction for future research lies in developing hybrid approaches that combine simplified models with adaptive learning techniques. For example, [123] model complicated contact with a simplified model, then quickly identify its parameters and use these to train a policy parameterized by the learned model. Building on this approach, future work could:

1. Investigate how solutions from simplified physical models can serve as initial policies, which can then be fine-tuned with reinforcement learning for more complex, realistic scenarios.

2. Develop methods for continuously updating contact model parameters during task execution, allowing for real-time adaptation to changing conditions.

These strategies could lead to more robust and adaptive control in complex manipulation tasks.

Whole-body contact-rich manipulation

Contact detection and prediction remain challenging topics in robotics. In this thesis, we simplified the contact problem by using rectangle and cylinder-shaped mobile robots for

delivery tasks, resulting in either point contacts around the cylinder or line contacts along the rectangle's edges. However, to achieve truly dexterous manipulation, we must move beyond these simplified contact models and leverage the full body of the robot for more robust and versatile interactions.

Future research in whole-body contact manipulation could focus on:

1. Continuous Contact Surface Modeling. Extend contact models beyond discrete point and line contacts to continuous surface contacts, but it requires a better model of the contact surface. This could enable more natural and stable interactions, especially for tasks involving complex object geometries.

2. Tactile-Visual Fusion for Better Contact State Estimation. Combine high-resolution tactile sensing with visual information to accurately estimate and predict complex contact states during whole-body manipulation. This multi-modal approach could provide a more comprehensive understanding of the robot's interaction with its environment.

Task and motion planning with LLM

Reinforcement learning, imitation learning, and model-based optimization approaches typically target short-horizon manipulation tasks. However, for long-horizon tasks, integrated task and motion planning is essential. To simplify this planning process, we can leverage task planning and decomposition techniques.

Large Language Models (LLMs) offer promising opportunities to enhance these planning processes in robotics. They can help break down complex manipulation tasks into simpler subtasks and effectively guide lower-level control algorithms. Moreover, LLMs show promise in synthesizing vast amounts of information about object properties, manipulation strategies, and physical principles to inform planning and control algorithms. We can utilize LLM-derived knowledge about object affordances and typical manipulation strategies to guide sampling-based motion planners.

Finally, LLMs have the potential to simplify manipulation programming by creating interfaces that allow non-expert users to specify complex manipulation tasks using natural language. These LLMs can then translate these instructions into executable robot plans, making advanced robotics more accessible to a broader user base.

Bibliography

[1] Shokraneh K Moghaddam and Ellips Masehian. Planning robot navigation among movable obstacles (namo) through a recursive approach. *Journal of Intelligent & Robotic Systems*, 83:603–634, 2016.

[2] Kai Zhang, Eric Lucet, Julien Alexandre dit Sandretto, and David Filliat. Navigation among movable obstacles using machine learning based total time cost optimization. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11321–11327. IEEE, 2023.

[3] Benoit Renault, Jacques Saraydaryan, and Olivier Simonin. Modeling a social placement cost to extend navigation among movable obstacles (namo) algorithms. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11345–11351. IEEE, 2020.

[4] Maozhen Wang, Rui Luo, Aykut Özgün Önol, and Taşkin Padir. Affordance-based mobile robot navigation among movable obstacles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2734–2740. IEEE, 2020.

[5] Zehui Meng, Hao Sun, Ken BH Teo, and Marcelo H Ang. Active path clearing navigation through environment reconfiguration in presence of movable obstacles. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 156–163. IEEE, 2018.

[6] Hsueh-Cheng Wang, Siao-Cing Huang, Po-Jui Huang, Kuo-Lun Wang, Yi-Chen Teng, Yu-Ting Ko, Dongsuk Jeon, and I-Chen Wu. Curriculum reinforcement learning from avoiding collisions to navigating among movable obstacles in diverse environments. *IEEE Robotics and Automation Letters*, 8(5):2740–2747, 2023.

[7] Jonathan Scholz, Nehchal Jindal, Martin Levihn, Charles L Isbell, and Henrik I Christensen. Navigation among movable obstacles with learned dynamic constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3706–3713. IEEE, 2016.

[8] Kirsty Ellis, Henry Zhang, Danail Stoyanov, and Dimitrios Kanoulas. Navigation among movable obstacles with object localization using photorealistic simulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1711–1716. IEEE, 2022.

[9] Nikolay Zherdev, Mikhail Kurenkov, Kristina Belikova, and Dzmitry Tsetserukou. Swipebot: Dnn-based autonomous robot navigation among movable obstacles in

cluttered environments. In *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pages 1–5. IEEE, 2023.

[10] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[11] Ermano Arruda, Michael J Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L Wyatt. Uncertainty averse pushing with model predictive path integral control. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 497–502. IEEE, 2017.

[12] Francois Robert Hogan, Eudald Romo Grau, and Alberto Rodriguez. Reactive planar manipulation with convex hybrid mpc. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 247–253. IEEE, 2018.

[13] Francois R Hogan and Alberto Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *The International Journal of Robotics Research*, 39(7):755–773, 2020.

[14] João Moura, Theodoros Stouraitis, and Sethu Vijayakumar. Non-prehensile planar manipulation via trajectory optimization with complementarity constraints. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 970–976. IEEE, 2022.

[15] Neel Doshi, Francois R Hogan, and Alberto Rodriguez. Hybrid differential dynamic programming for planar manipulation primitives. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6759–6765. IEEE, 2020.

[16] Victor Emeli and Akansel Cosgun. Joint path and push planning among movable obstacles. *arXiv preprint arXiv:2010.14733*, 2020.

[17] B.P. Gerkey and M.J. Mataric. Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 464–469, 2002.

[18] D. Nieuwenhuisen, A.F. van der Stappen, and M.H. Overmars. Path planning for pushing a disk using compliance. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 714–720, 2005.

[19] Nikhil Chavan-Dafle and Alberto Rodriguez. Stable prehensile pushing: In-hand manipulation with alternating sticking contacts. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 254–261. IEEE, 2018.

[20] Marek Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Moerwald, and Jeremy L Wyatt. Learning modular and transferable forward models of the motions of push manipulated objects. *Autonomous Robots*, 41(5):1061–1082, 2017.

[21] Adam Heins, Michael Jakob, and Angela P Schoellig. Mobile manipulation in unknown environments with differential inverse kinematics control. In *2021 18th Conference on Robots and Vision (CRV)*, pages 64–71. IEEE, 2021.

[22] Dennis Nieuwenhuisen, A Frank van der Stappen, and Mark H Overmars. Pushing using compliance. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2010–2016. IEEE, 2006.

[23] Senka Krivic and Justus Piater. Online adaptation of robot pushing control to object properties. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4614–4621, 2018.

[24] Senka Krivic, Emre Ugur, and Justus Piater. A robust pushing skill for object delivery between obstacles. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1184–1189, 2016.

[25] Senka Krivic and Justus Piater. Pushing corridors for delivering unknown objects with a mobile robot. *Autonomous Robots*, 43(6):1435–1452, 2019.

[26] Tekin Meriçli, Manuela Veloso, and H Levent Akın. Push-manipulation of complex passive mobile objects using experimentally acquired motion models. *Autonomous Robots*, 38(3):317–329, 2015.

[27] Corrado Pezzato, Chadi Salmi, Max Spahn, Elia Trevisan, Javier Alonso-Mora, and Carlos Hernandez Corbato. Sampling-based model predictive control leveraging parallelizable physics simulations. *arXiv preprint arXiv:2307.09105*, 2023.

[28] Manfred Lau, Jun Mitani, and Takeo Igarashi. Automatic learning of pushing strategy for delivery of irregular-shaped objects. In *2011 IEEE international conference on robotics and automation*, pages 3733–3738. IEEE, 2011.

[29] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, pages 1930–1936, 2015.

[30] Anuj Pasricha, Yi-Shiuan Tung, Bradley Hayes, and Alessandro Roncone. Pokerrt: Poking as a skill and failure recovery tactic for planar non-prehensile manipulation. *IEEE Robotics and Automation Letters*, 7(2):4480–4487, 2022.

[31] Claudio Zito, Rustam Stolkin, Marek Kopicki, and Jeremy L Wyatt. Two-level rrt planning for robotic push manipulation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 678–685. IEEE, 2012.

[32] Bernardo Aceituno, Alberto Rodriguez, Shubham Tulsiani, Abhinav Gupta, and Mustafa Mukadam. A differentiable recipe for learning visual non-prehensile planar manipulation. In *2022, Conference on Robot Learning*, pages 137–147. PMLR, 2022.

[33] Nils Dengler, David Großklaus, and Maren Bennewitz. Learning goal-oriented non-prehensile pushing in cluttered scenes. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1116–1122. IEEE, 2022.

[34] Nils Dengler, Juan Del Aguila Ferrandis, João Moura, Sethu Vijayakumar, and Maren Bennewitz. Learning goal-directed object pushing in cluttered scenes with location-based attention. *arXiv preprint arXiv:2403.17667*, 2024.

[35] Wisdom C Agboh and Mehmet R Dogar. Pushing fast and slow: task-adaptive planning for non-prehensile manipulation under uncertainty. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pages 160–176. Springer, 2020.

[36] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[37] Matthew T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[38] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 800–815. Springer, 2020.

[39] Filippo Bertoncelli, Fabio Ruggiero, and Lorenzo Sabattini. Linear time-varying mpc for nonprehensile object manipulation with a nonholonomic mobile robot. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11032–11038. IEEE, 2020.

[40] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. Acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, Oct 2021.

[41] Jean-Pierre Sleiman, Jan Carius, Ruben Grandia, Martin Wermelinger, and Marco Hutter. Contact-implicit trajectory optimization for dynamic object manipulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6814–6821. IEEE, 2019.

[42] Lin Cong, Michael Grner, Philipp Ruppel, Hongzhuo Liang, Norman Hendrich, and Jianwei Zhang. Self-adapting recurrent models for object pushing from learning in simulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5310. IEEE, 2020.

[43] Amir Zarei Khabjani, Hossein Karimpour, and Mehdi Keshmiri. Robotic box pushing under indeterminate anisotropic friction properties. *International Journal of Dynamics and Control*, 9(3):872–884, 2021.

[44] Jiaji Zhou, Yifan Hou, and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 38(12-13):1477–1489, 2019.

[45] Amin Fakhari, Mehdi Keshmiri, and Mohammad Keshmiri. Dynamic modeling and slippage analysis in object manipulation by soft fingers. In *ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2014.

[46] Alberto Rigo, Yiyu Chen, Satyandra K Gupta, and Quan Nguyen. Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control. *arXiv preprint arXiv:2210.03442*, 2022.

[47] Mark De Berg and Dirk HP Gerrits. Computing push plans for disk-shaped robots. *The International Journal of Computational Geometry & Applications (IJCGA)*, 23(01):29–48, 2013.

[48] Dennis Nieuwenhuisen, A Frank van der Stappen, and Mark H Overmars. Path planning for pushing a disk using compliance. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 714–720. IEEE, 2005.

[49] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[50] Huiyu Zhou and Huosheng Hu. Human motion tracking for rehabilitation—a survey. *Biomedical signal processing and control*, 3(1):1–18, 2008.

[51] Manon Kok, Jeroen D Hol, and Thomas B Schön. Using inertial sensors for position and orientation estimation. *Foundations and Trends in Signal Processing*, 11(1-2):1–153, 2017.

[52] Angelo M Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE Transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.

[53] João Luís Marins, Xiaoping Yun, Eric R Bachmann, Robert B McGhee, and Michael J Zyda. An extended kalman filter for quaternion-based orientation estimation using marg sensors. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS).*, volume 4, pages 2003–2011. IEEE, 2001.

[54] Roberto G Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors*, 15(8):19302–19330, 2015.

[55] Sen Wang, Hongkai Wen, Ronald Clark, and Niki Trigoni. Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1910–1917. IEEE, 2016.

[56] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3995–4001, 2017.

[57] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[58] Edgar Kraft. A quaternion-based unscented kalman filter for orientation tracking. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 1, pages 47–54. IEEE Cairns, Queensland, Australia, 2003.

[59] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011.

[60] Manon Kok, Jeroen Hol, and Thomas Schön. An optimization-based approach to human body motion capture using inertial sensors. In *19th World Congress of the International Federation of Automatic Control (IFAC), Cape Town, South Africa, August 24-29, 2014*, pages 79–85. International Federation of Automatic Control, 2014.

[61] Martin Brossard, Silvere Bonnabel, and Axel Barrau. Denoising imu gyroscopes with deep learning for open-loop attitude estimation. *arXiv preprint arXiv:2002.10718*, 2020.

[62] Jun Morimoto and Kenji Doya. Reinforcement learning state estimator. *Neural Computation*, 19(3):730–756, 2007.

[63] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[64] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[65] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[66] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[67] Vassilios Petridis and Stavros Petridis. Construction of neural network based lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065. IEEE, 2006.

[68] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters (RA-L & IROS), accepted and in press*, 2020.

[69] Liang Hu, Chengwei Wu, and Wei Pan. Lyapunov-based reinforcement learning state estimator. *arXiv preprint arXiv:2010.13529*, 2020.

[70] Jochen Stüber, Claudio Zito, and Rustam Stolkin. Let's push things forward: A survey on robot pushing. *Frontiers in Robotics and AI*, 7:8, 2020.

[71] Kevin M Lynch. The mechanics of fine manipulation by pushing. In *ICRA*, pages 2269–2276. Citeseer, 1992.

[72] Spyros G Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[73] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[74] Soo Hong Lee and MR Cutkosky. Fixture planning with friction. *Journal of Manufacturing Science and Engineering*, 113(3):320–327, 1991.

[75] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[76] Mohd Saifizi Saidonr, Hazry Desa, and Md Noor Rudzuan. A differential steering control with proportional controller for an autonomous mobile robot. In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, pages 90–94. IEEE, 2011.

[77] Kevin M Lynch. Locally controllable manipulation by stable pushing. *IEEE Transactions on Robotics and Automation*, 15(2):318–327, 1999.

[78] Stephen Boyd. Convex optimization–boyd and vandenberghe, 2004.

[79] Filippo Bertoncelli, Fabio Ruggiero, and Lorenzo Sabattini. Characterization of grasp configurations for multi-robot object pushing. In *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 38–46. IEEE, 2021.

[80] Jean Chagas Vaz and Paul Oh. Material handling by humanoid robot while pushing carts using a walking pattern based on capture point. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9796–9801. IEEE, 2020.

[81] Pankaj K Agarwal, J-C Latombe, Rajeev Motwani, and Prabhakar Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3124–3129. IEEE, 1997.

[82] Yujie Tang, Hai Zhu, Susan Potters, Martijn Wisse, and Wei Pan. Unwieldy object delivery with nonholonomic mobile base: A stable pushing approach. *IEEE Robotics and Automation Letters*, 2023.

[83] Jordi Pages, Luca Marchionni, and Francesco Ferro. Tiago: the modular robot that adapts to different research needs. In *International Workshop on Robot Modularity, IROS*, volume 290, 2016.

[84] Jian Shi, J Zachary Woodruff, Paul B Umbanhowar, and Kevin M Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017.

[85] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T. Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526, 2021.

[86] Akira Nakashima, Yoshiyuki Ooka, and Yoshikazu Hayakawa. Contact transition modelling on planar manipulation system with lugre friction model. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 300–307. IEEE, 2015.

[87] T Piatkowski. Gms friction model approximation. *Mechanism and Machine Theory*, 75:1–11, 2014.

[88] Neil Biehn, Stephen L Campbell, Laurent Jay, and Tracey Westbrook. Some comments on dae theory for irk methods and trajectory optimization. *Journal of Computational and Applied Mathematics*, 120(1-2):109–131, 2000.

[89] Heike Vallery and Arend L. Schwab. *Advanced Dynamics*. Delft University of Technology, 2020.

[90] SMH Sadati, SE Naghibi, A Shiva, S Zschaler, H Hauser, I Walker, K Althoefer, and T Nanayakkara. Autotmtdyn: A matlab software package to drive tmt lagrange dynamics of series rigid-and continuum-link mechanisms. In *IROS 2018 Workshop on Soft Robotic Modeling and Control: Bringing Together Articulated Soft Robots and Soft-Bodied Robots*, 2018.

[91] Yujie Tang and Martijn Wisse. Pushing manipulation with a nonholonomic mobile base. 2023.

[92] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016.

[93] M Fliess, J Lévine, P Martin, and P Rouchon. Flatness and motion planning: the car with n trailers. In *Proc. 2nd ECC, Groningen, The Netherlands*, pages 1518–1522, 1993.

[94] Tao Pang, HJ Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on robotics*, 2023.

[95] Jake Welde, James Paulos, and Vijay Kumar. Dynamically feasible task space planning for underactuated aerial manipulators. *IEEE Robotics and Automation Letters*, 6(2):3232–3239, 2021.

[96] Hale ERİŞKİN and Ahmet YÜCESAN. Bézier curve with a minimal jerk energy. *Math. Sci. Appl. E-Notes*, 4(2):139–148, 2016.

[97] H Meier and Horst Nowacki. Interpolating curves with gradual changes in curvature. *Computer Aided Geometric Design*, 4(4):297–305, 1987.

[98] J. Svacha, K. Mohta, M. Watterson, G. Loianno, and V. Kumar. Inertial velocity and attitude estimation for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.

[99] A. Widy and K. T. Woo. Robust attitude estimation method for underwater vehicles with external and internal magnetic noise rejection using adaptive indirect kalman filter. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2595–2600, 2017.

[100] V. Joukov, J. Ćesić, K. Westermann, I. Marković, D. Kulić, and I. Petrović. Human motion estimation on lie groups using imu measurements. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1965–1972, 2017.

[101] Simone Sabatelli, Francesco Sechi, Luca Fanucci, and Alessandro Rocchi. A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units. In *2011 Design, Automation & Test in Europe*, pages 1–4. IEEE, 2011.

[102] Rachel V Vitali, Ryan S McGinnis, and Noel C Perkins. Robust error-state kalman filter for estimating imu orientation. *IEEE Sensors Journal*, 21(3):3561–3569, 2020.

[103] B. Allotta, L. Chisci, R. Costanzi, F. Fanelli, C. Fantacci, E. Meli, A. Ridolfi, A. Caiti, F. Di Corato, and D. Fenucci. A comparison between ekf-based and ukf-based navigation algorithms for auvs localization. In *OCEANS 2015 - Genova*, pages 1–5, 2015.

[104] S. P. Tseng, W. Li, C. Sheng, J. Hsu, and C. Chen. Motion and attitude estimation using inertial measurements with complementary filter. In *2011 8th Asian Control Conference (ASCC)*, pages 863–868, 2011.

[105] Mei Jin, Jinge Zhao, Ju Jin, Guohui Yu, and Wenchao Li. The adaptive kalman filter based on fuzzy logic for inertial motion capture system. *Measurement*, 49:196–204, 2014.

[106] Rick A Hyde, Laurence P Ketteringham, Simon A Neild, and Rosie JS Jones. Estimation of upper-limb orientation based on accelerometer and gyroscope measurements. *IEEE Transactions on Biomedical Engineering*, 55(2):746–754, 2008.

[107] Liang Hu, Yujie Tang, Zhipeng Zhou, and Wei Pan. Reinforcement learning for orientation estimation using inertial sensors with performance guarantee. In *IEEE International Conference on Robotics and Automation. arXiv preprint arXiv:2103.02357*, 2021.

[108] Qingrui Zhang, Wei Pan, and Vasso Reppa. Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[109] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

[110] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[111] M. A. Skoglund, Z. Sjanic, and M. Kok. On orientation estimation using iterative methods in euclidean space. In *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8, 2017.

[112] John L Crassidis, F Landis Markley, and Yang Cheng. Survey of nonlinear attitude estimation methods. *Journal of guidance, control, and dynamics*, 30(1):12–28, 2007.

[113] Joan Sola. Quaternion kinematics for the error-state kf. *Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep*, 2012.

[114] Konrad Reif, Stefan Gunther, Engin Yaz, and Rolf Unbehauen. Stochastic stability of the discrete-time extended kalman filter. *IEEE Transactions on Automatic Control*, 44(4):714–728, 1999.

[115] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introductions*. The MIT Press, 2nd edition, 2018.

[116] Qingrui Zhang, Wei Pan, and Vasso Reppa. Model-reference reinforcement learning control of autonomous surface vehicles. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5291–5296. IEEE, 2020.

[117] Qingrui Zhang, Wei Pan, and Vasso Reppa. Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles. *arXiv preprint arXiv:2008.07240*, 2020.

[118] Theodore J Perkins and Andrew G Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3(Dec):803–832, 2002.

[119] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, Jul. 2018.

[120] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[121] Yunlong Song, Sangbae Kim, and Davide Scaramuzza. Learning quadruped locomotion using differentiable simulation. *arXiv preprint arXiv:2403.14864*, 2024.

[122] Clemens Schwarke, Victor Klemm, Jesus Tordesillas, Jean-Pierre Sleiman, and Marco Hutter. Learning quadrupedal locomotion via differentiable simulation. *arXiv preprint arXiv:2404.02887*, 2024.

[123] Hien Bui and Michael Posa. Enhancing task performance of learned simplified models via reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9212–9219. IEEE, 2024.

Glossary

Notation

Throughout this thesis, scalers are denoted by plain lowercase letters, e.g. x, vectors by bold lowercase, e.g. $\mathbf{x}$, matrices by plain uppercase, e.g. M, and sets by calligraphic uppercase, e.g. $\mathcal{X}$. The superscript $\mathbf{x}^\top$ or $A^\top$ denotes the transpose of a vector $\mathbf{x}$ or a matrix A.

Denote by $\{\mathcal{W}\}$, $\{\mathcal{R}\}$, and $\{\mathcal{O}\}$, the global world frame, the robot body frame, and the object body frame, respectively. For example, a vector expressed in the robot frame is denoted by a superscript, such as ${}^{\mathcal{R}}\mathbf{x}$. ${}^{\mathcal{W}}\mathbf{x}$ is always omitted. A rotation matrix that transforms from the robot frame to the world frame is denoted as ${}^{\mathcal{W}}R_{\mathcal{R}}$.

$\mathbf{x}_{\text{r\_e}}$	the extended robot state.
$\mathbf{x}_{\text{r}}$	the robot state.
$\mathbf{x}_{\text{o}}$	the object state.
$\mathbf{p}_{\text{r}}$	robot position (the geometric center of its four wheels).
$\mathbf{u}_{\text{r}}$	the control input of the robot.
v_{r}	the linear velocity of the robot.
ω_{r}	the angular velocity of the robot.
a_{r}	the linear accelerations of the robot.
ξ_{r}	the angular accelerations of the robot.
${}^{\mathcal{O}}\mathbf{v}_{\text{o}}$	the twist of the object.
${}^{\mathcal{O}}\mathbf{w}_{\text{p}}$	wrench applied by the pusher.
${}^{\mathcal{O}}\mathbf{f}_{\text{p}}$	the pushing force.
IRC	instantaneous center of rotation.
${}^{\mathcal{O}}\mathcal{W}_{\text{p}}$	a convex hull of all possible wrenches.
${}^{\mathcal{O}}\mathcal{V}_{\text{o}}$	all possible twists of the object.
ϕ	the angle between the object and the robot frame.
d	the y coordinate of robot center in the object frame.
${}^{\mathcal{O}}f_{\text{x,p}}$	the x-component of the push force in the object frame.
$\mathbf{w}_{\text{g}}$	the friction wrench exerted by the ground on the object.
$\tilde{\mathbf{w}}_{\text{g}}$	the simplified friction wrench exerted by the ground to the object.
M_{o}	the inertia matrix of the object.
$W_{\text{o}}, L_{\text{o}}$	the width and length of the object.
r_{r}	radius of the robot.
$\tau(t)$	a geometric path on x-y plane.
$T(t)$	the path's tangent direction.
$N(t)$	the unit normal vector of the path.
$\kappa(t)$	the curvature of the path.

Specific sets

$\mathbb{R}$	real numbers.
$\mathbb{R}^n$	real n-vectors.
$\mathbb{R}^{m \times n}$	real $m \times n$ matrices.

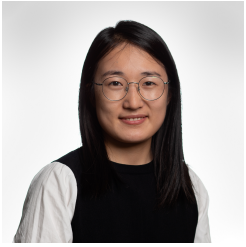
Model predictive control

N	number of time steps in receding horizon planning.
Δt	time step.
$.^t$	the super index t indicates value of the variable at stage t.
$J^t(\cdot)$	the t-th stage cost.
$J^N(\cdot)$	the terminal stage cost.
$\mathbf{h}(\cdot)$	the path constraint

List of abbreviations

NAMO	navigation among movable obstacles
NMPC	nonlinear model predictive control
MPC	model predictive control
OCP	optimal control problem
MPPI	model predictive path integral
LTV	linear time-varying
DRL	deep reinforcement learning
IMU	inertial measurement units
EKF	extended Kalman filter
UKF	unscented Kalman filter
CF	complementary filter
RRT	rapidly-exploring random tree
RL	reinforcement learning
LRLOE	Lyapunov-based reinforcement learning orientation estimation
RLC-EKF	reinforcement learning compensated EKF
MDP	Markov decision process
DNN	deep neural network
SAC	soft actor-critic
PPO	proximal policy optimization
IRC	instantaneous center of rotation
PID	proportional-integral-derivative
DAE	differential-algebraic equation
IRK	implicit Runge-Kutta
GCM	generalized coordinate model
LLM	large language model
ROS	robot operating system

Curriculum vitæ



Yujie Tang was born in May 1995 in Shanghai, China. She obtained her B.Sc. and M.Sc. degrees in Mechanical Engineering from Shanghai University, China in 2017 and 2020, respectively. From January to March in 2017, she was an intern in Fiat Chrysler Automobiles (FCA) in Shanghai, China. In her masters, she was supervised by Prof. Yangmin Xie, and worked on autonomous navigation and mapping in unstructured environments.

In January 2021, she was sponsored by the Chinese Scholarship Council (CSC) to become a PhD candidate at the Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands. From 2019 to 2020, she worked with Dr. Wei Pan on Reinforcement learning-based state estimation. Then she was supervised by Prof. Martijn Wisse, worked on the topic of nonprehensile manipulation with Mobile Robots. Her research interest expands from robot navigation to manipulation. With a solid background in mechanical engineering, she focuses on the integration of mechanics, control, and machine learning in robotics. From October 2024 to March 2025, she interned at Xiaomi Corporation's Self-Driving and Robotics Department, working on reinforcement learning for robot manipulation and its sim-to-real transfer.

List of publications

Publications related to the PhD project:

- **Y. Tang**, M. Wisse, W. Pan, “Reactive Nonprehensile Planar Manipulation via Differential Flatness,” submitted to *IEEE Robotics and Automation Letters*, under review.
- **Y. Tang**, M. Wisse, W. Pan, “Unwieldy Object Delivery With Nonholonomic Mobile Base: A free Pushing Approach,” *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8991-8998, Sep. 2024.
- **Y. Tang**, H. Zhu, S. Potters, M. Wisse, W. Pan, “Unwieldy Object Delivery With Nonholonomic Mobile Base: A Stable Pushing Approach,” *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7727-7734, Nov. 2023.
- **Y. Tang**, M. Wisse, “Pushing manipulation with a Nonholonomic Mobile Base” *ECCOMAS Thematic Conference on Multibody Dynamics*, July 24 - 28, 2023, Lisbon, Portugal.
- **Y. Tang**, L. Hu, Q. Zhang, W. Pan, “Reinforcement learning compensated extended Kalman filter for attitude estimation” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*
- L. Hu*, **Y. Tang***, Z. Zhou, W. Pan, “Reinforcement learning for orientation estimation using inertial sensors with performance guarantee” *IEEE International Conference on Robotics and Automation (ICRA), 2021.*

Other publications:

- J. Zhu, **Y. Tang**, X. Shao and Y. Xie, "Multisensor Fusion Using Fuzzy Inference System for a Visual-IMU-Wheel Odometry," in *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-16, 2021.
- Y. Xie, **Y. Tang**, R. Zhou, Y. Guo and H. Shi, "Map merging with terrain-adaptive density using mobile 3D laser scanner," in *Robotics and Autonomous Systems*, vol. 134, 103649, 2021.
- **Y. Tang**, J. Cai, M. Chen, X. Yan and Y. Xie, "An autonomous exploration algorithm using environment-robot interacted traversability analysis," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.*

* indicates equal contributions.