Trajectory Generation for Mobile Manipulators with Differential Geometry Behavior Encoding beyond Model Predictive Control

Spahn, M.

DOI
10.4233/uuid:e3ed26d7-7ea6-4b8f-9197-2c52a26e322b

Publication date
2024

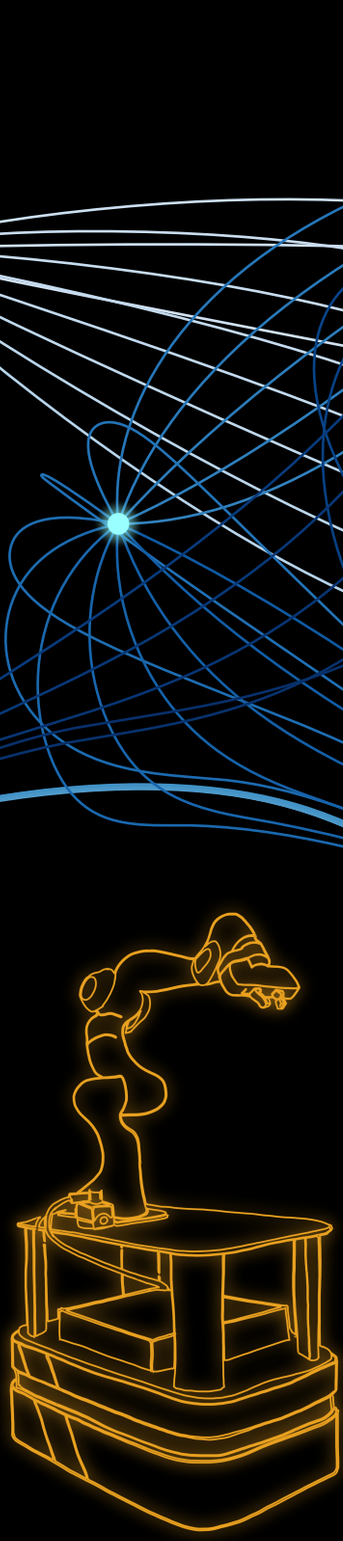
Document Version
Final published version

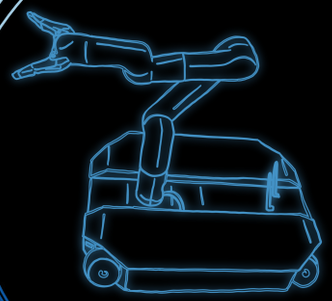
Citation (APA)
Spahn, M. (2024). *Trajectory Generation for Mobile Manipulators with Differential Geometry: Behavior Encoding beyond Model Predictive Control*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:e3ed26d7-7ea6-4b8f-9197-2c52a26e322b

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Trajectory Generation for Mobile Manipulators with Differential Geometry

Behavior Encoding beyond Model Predictive Control





Max Spahn

Trajectory Generation for Mobile Manipulators with Differential Geometry

Max Spahn

Propositions

accompanying the dissertation

Trajectory Generation for Mobile Manipulators with Differential Geometry

Behavior Encoding beyond
Model Predictive Control

by

Max Spahn

1. Safety in the sense of collision avoidance is inherently in conflict with fast robotic manipulation. (This thesis)
2. In robotics, robustness is more valuable than optimality and formal guarantees. (This thesis)
3. Optimization fabrics create human-like motions. (This thesis)
4. Warranties on collision avoidance in dynamic environments are not realistic, striving for it is thus pointless (This thesis).
5. Academic publications in robotics are of little use for advancing the field without proper software releases.
6. Major engineering advancements are unlikely to come from academic institutions, but rather from companies.
7. Effective peer-reviewing requires a measurable reward system.
8. Individual career goals harm collaborative research efforts.
9. Effective action to combat climate change requires major natural disaster for society to understand the urgency.
10. The older a democracy is, the more socially segmented its society becomes.

These propositions are regarded as opposable and defendable, and have been approved as such by the promoters Prof. dr. M. Wisse and Dr. J. Alonso-Mora.

Trajectory Generation for Mobile Manipulators with Differential Geometry

Behavior Encoding beyond Model Predictive Control

TRAJECTORY GENERATION FOR MOBILE MANIPULATORS WITH DIFFERENTIAL GEOMETRY

BEHAVIOR ENCODING BEYOND MODEL PREDICTIVE CONTROL

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 of Doctorates

to be defended publicly on

Wednesday the *11th* of *December 2024*, at *17:30*

by

Max SPAHN

Master of Science in Mechanical Engineering,
Rheinisch-Westfälische Technische Hochschule Aachen, Germany
born in Bergisch Gladbach, Germany.

This dissertation has been approved by the promotors.

promotor: Prof. dr. M. Wisse
copromotor: Dr. J. Alonso-Mora

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Prof. dr. M. Wisse,	Delft University of Technology, promotor
Dr. J. Alonso-Mora,	Delft University of Technology, promotor
Indpendent members:	
Prof. dr. ir. D. A. Abbink,	Delft University of Technology
Dr. S. Calinon,	Idiap Research Institute, Switzerland
Dr. A. Saccon,	Eindhoven University of Technology
Dr. N. Ratliff,	NVIDIA Research, USA
Prof. dr. ir. J.C.F de Winter,	Delft University of Technology, *reserve member*

This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Keywords:	Trajectory generation, motion planning, mobile manipulation, geometric control, model predictive control
Printed by:	Gildeprint, The Netherlands
Front & Back:	Max Spahn using Geometric Fabrics and Matplotlib Code for visuals available at https://github.com/maxspahn/visuals_fabrics

ISBN 978-94-6366-971-9

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

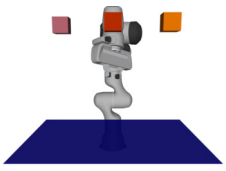
CONTENTS

SUMMARY

As robotics will play a crucial role in the future of our modern societies, the need for advancements in the field is more pronounced than ever. While robots are already present in industrial settings, they are noticeably absent from dynamic environments. Dynamic environments are characterized by other moving agents, such as humans, varying tasks and high safety requirements. With the aim to deploy robots in such environments, Trajectory Generation (TG) becomes crucial. TG approaches aim to compute sequences of control commands that bring the robot from its current configuration to a desired goal state while avoiding collisions with obstacles and itself. Thus, it is directly placed between task planning, the problem of defining what high level tasks should be executed in which order, and control, the problem of executing motor commands. A good solution to TG must be fast, to cope with changes in the environment, flexible to different goal definitions, and it should promote safety. Most advancements in the field of robotics in TG focus either on manipulators or on mobile robots. However, the combination of both systems seems inevitable for deployment in human-shared environments.

TG for mobile manipulation is usually formulated as an optimization problem over a finite time horizon. This approach is known as Model Predictive Control (MPC) and is widely used in the field of autonomous driving thanks to its feasibility and stability guarantees. In Chapter 4, we present a method to bring MPC to mobile manipulation. The method formulates the TG problem for the entire kinematic chain and relies on Free Space Decomposition (FSD) for collision avoidance. This leads to reasonable control frequencies of 10Hz independent on the amount of collision obstacles in the environment. Importantly, this approach allows for coupled motion of the mobile base and the manipulator. This is beneficial in situations where synchronization of the two subsystems is crucial, such as opening doors or moving obstacles around. Despite simplifications on the environment representations, computational costs limit the applicability of MPC to mobile manipulation as motion is not considered truly reactive and different components, such as goal attraction and collision avoidance, are not easily separable.

A recent novel approach to receding horizon control is the formulation as a purely geometric problem. Early successes in this direction, including Cartesian Impedance Control (CIC) and Artificial Potential Fields (APF), led to the formulation as sets of dynamical systems on smooth manifolds in the configuration space. The framework of Optimization Fabrics (fabrics) unifies such ideas, offers stability guarantees in static environments, and results in highly reactive behavior, similar to simple low-level controllers. This framework relies on non-Riemannian geometry to shape a smooth manifold of the configuration space with individual behaviors, such as collision avoidance, joint-limit avoidance, and goal attraction. In Chapter 5, we present a generalization of fabrics to dynamic environments. We refer to the resulting framework as Dynamic Fabrics (DF). The generalization uses time-



parameterized manifolds to integrate moving obstacles and time-parameterized reference trajectories. The latter is especially important for long-horizon TG that may exhibit local minima. Importantly, Chapter 5 shows that the dynamic component of DF is required when coping with moving obstacles. As repulsive forces in fabrics are proportional to the approaching speed of obstacle and robot, collision avoidance in a pseudo-static fashion is not sufficient when the robot is moving slowly. Finally, we deploy the general framework of DF to several real-world settings showing the applicability of the framework to mobile manipulation. First, we present a way to integrate non-holonomic constraints into the framework. Despite loosing formal guarantees on convergence, the method is shown to be the natural extension to wheeled mobile robots characterized by non-holonomic constraints. Second, Chapter 6 presents a symbolic implementation of fabrics to achieve higher control frequencies. Symbolic implementations are possible because the framework of fabrics is based on differential equations of second order, for which a closed-form solution exists. For changing environments, obstacles states are then only concretized at runtime. Additionally, we show symbolic hyperparameters can be tuned automatically to achieve expert-level tuning performance. Third, to overcome high requirements on the perception pipeline, Chapter 7 integrates different implicit environment representations into the framework. Using Signed Distance Fields (SDF) and FSD for example is widely used in mobile robotics when formulating TG as MPC. We show that the same representations can be used in fabrics while achieving faster solver times. Finally, Chapter 8 deploys a mobile manipulator controlled by fabrics in a supermarket. Dexterous manipulation is programmed using learning-from-demonstration, with fabrics as the underlying encoding. That allows to teach rather than program complicated behaviors while maintaining properties on collision avoidance.

This thesis presents insights into aspects of motion planning, advances the framework of fabrics for TG, and compares it extensively to the more commonly used method of MPC. Through the ideas presented in this thesis, we hope to encourage the usage of geometric properties of robotic systems deployed to human-shared environments. This approach does not only provide reactive TG but also may act as a compact encoding of trajectories for learning-based methods in the future.

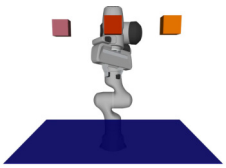
SAMENVATTING

Omdat robotica een cruciale rol zal spelen in de toekomst van onze moderne samenlevingen, is de behoefte aan vooruitgang op dit gebied groter dan ooit. Hoewel robots al aanwezig zijn in industriële omgevingen, zijn ze opvallend afwezig in dynamische omgevingen. Dynamische omgevingen worden gekenmerkt door andere bewegende agenten, zoals mensen, variërende taken en hoge veiligheidseisen. Met als doel robots in dergelijke omgevingen in te zetten, wordt Trajectory Generation (TG) cruciaal. TG-benaderingen zijn gericht op het berekenen van reeksen acties die de robot van zijn huidige configuratie naar een gewenste eindtoestand brengen, terwijl botsingen met obstakels en met zichzelf worden vermeden. Hierdoor bevindt TG zich direct tussen taakplanning – het probleem van welke taken in welke volgorde moeten worden uitgevoerd – en controle, het probleem van het uitvoeren van motorcommando's. Een goede oplossing voor TG moet snel zijn om veranderingen in de omgeving het hoofd te bieden, flexibel voor verschillende doelstellingen, en de veiligheid bevorderen. De meeste vooruitgangen op het gebied van robotica in TG richten zich ofwel op manipulatoren of op mobiele robots. Echter, de combinatie van beide systemen lijkt onvermijdelijk voor inzet in door mensen gedeelde omgevingen.

TG voor mobiele manipulatie wordt meestal geformuleerd als een optimalisatieprobleem met een eindige tijdshorizon. Deze benadering staat bekend als Model Predictive Control (MPC) en wordt veel gebruikt op het gebied van autonoom rijden vanwege de haalbaarheids- en stabiliteitsgaranties. In Chapter 4 presenteren we een methode om MPC toe te passen op mobiele manipulatie. De methode formuleert het TG-probleem voor de gehele kinematische keten en maakt gebruik van Free Space Decomposition (FSD) voor het vermijden van botsingen. Dit leidt tot controlefrequenties van 10 Hz, onafhankelijk van het aantal obstakels in de omgeving. Belangrijk is dat deze aanpak een gekoppelde beweging van de mobiele basis en de manipulator mogelijk maakt. Dit is voordelig in situaties waarin synchronisatie van de twee subsystemen cruciaal is, zoals bij het openen van deuren of het verplaatsen van obstakels.

Ondanks deze vereenvoudigingen in de representatie van de omgeving, beperken de rekentijden de toepasbaarheid van MPC op mobiele manipulatie, omdat de beweging niet echt reactief wordt beschouwd en verschillende componenten zoals aantrekking tot het doel en het vermijden van botsingen niet gemakkelijk te scheiden zijn. Een recente benadering voor receding horizon control is de formulering als een puur geometrisch probleem. Vroege successen in deze richting, waaronder Cartesian Impedance Control (CIC) en Artificial Potential Fields (APF), leidden tot de formulering als verzamelingen dynamische systemen in de configuratieruimte. Het kader van Optimization Fabrics (fabrics) verenigt dergelijke ideeën, biedt stabiliteitsgaranties in statische omgevingen en resulteert in zeer reactief gedrag vergelijkbaar met eenvoudige laag-niveau controllers.

Dit kader is gebaseerd op niet-Riemanniaanse geometrie om een manifold van de confi-



guratieruimte te vormen met individuele gedragingen zoals het vermijden van botsingen, motorlimieten en het bereiken van het doel. In Chapter 5 presenteren we een generalisatie van fabrics naar dynamische omgevingen. We verwijzen naar het resulterende kader als Dynamic Fabrics (DF). De generalisatie maakt gebruik van tijdgeparameteriseerde manifolds om bewegende obstakels en tijdgeparameteriseerde referentietrajecten te integreren. Dit laatste is vooral belangrijk voor TG met een lange horizon die lokale minima kan vertonen. Belangrijk is dat in Chapter 5 wordt aangetoond dat de dynamische component van DF noodzakelijk is bij het omgaan met bewegende obstakels. Omdat afstotende krachten in fabrics evenredig zijn aan de naderingssnelheid van obstakel en robot, is een pseudo-statische botsingsvermijding niet voldoende wanneer de robot langzaam beweegt.

Tot slot implementeren we het algemene kader van DF in verschillende real-world omgevingen om de toepasbaarheid van het kader op mobiele manipulatie te demonstreren. Ten eerste presenteren we een manier om niet-holonome beperkingen in het kader te integreren. Ondanks het verlies van formele garanties op convergentie, blijkt de methode de natuurlijke uitbreiding voor mobiele robots met niet-holonome beperkingen. Ten tweede wordt in Chapter 6 een symbolische implementatie van fabrics gepresenteerd om hogere controlefrequenties te bereiken. Symbolische implementaties zijn mogelijk omdat het kader van fabrics is gebaseerd op differentiaalvergelijkingen van de tweede orde, waarvoor een gesloten oplossing bestaat. Voor veranderende omgevingen worden obstakelstatussen alleen in runtime aangeroepen. Daarnaast tonen we aan dat symbolische hyperparameters automatisch kunnen worden afgesteld om een prestatie op expertniveau te bereiken. Ten derde integreert Chapter 7 verschillende impliciete representaties van de omgeving, om aan de hoge eisen van de perceptiepijplijn te voldoen, Het gebruik van Signed Distance Fields (SDF) en FSD is bijvoorbeeld veelgebruikt in de mobiele robotica bij het formuleren van TG als MPC. We laten zien dat dezelfde voorstellingen in fabrics kunnen worden gebruikt, met kortere berekeningstijden als resultaat. Tot slot implementeert Chapter 8 een mobiele manipulator die wordt bestuurd door fabrics in een supermarkt. Behendige manipulaties worden geprogrammeerd met behulp van leren-uit-demonstratie, met fabrics als de onderliggende codering. Dit maakt het mogelijk om ingewikkelde gedragingen aan te leren in plaats van ze te programmeren, het vermijden van botsingen als eigenschap behouden blijft.

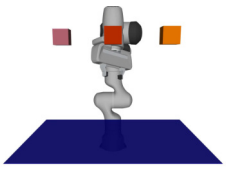
Deze scriptie presenteert inzichten in aspecten van bewegingsplanning, breidt het kader van fabrics voor TG uit en vergelijkt het uitgebreid met de meer gebruikte methode van MPC. Met de ideeën die in deze scriptie worden gepresenteerd, hopen we het gebruik van geometrische eigenschappen van robotsystemen in door mensen gedeelde omgevingen te stimuleren. Deze benadering biedt niet alleen reactieve TG, maar kan ook dienen als een compacte codering van trajecten voor toekomstig gebruik in op leren gebaseerde methoden.

ZUSAMMENFASSUNG

Da Robotik eine zentrale Rolle in der Zukunft unserer modernen Gesellschaften spielen wird, ist der Bedarf an Fortschritten in diesem Bereich ausgeprägter denn je. Während Roboter bereits in industriellen Umgebungen präsent sind, findet man sie selten in dynamischen Umgebungen. Dynamische Umgebungen sind durch andere sich-bewegende Akteure wie Menschen, wechselnde Aufgaben und hohe Sicherheitsanforderungen gekennzeichnet. Um Roboter in solchen Umgebungen einzusetzen, wird Trajectory Generation (TG) entscheidend. TG-Ansätze zielen darauf ab, Sequenzen von Steuerbefehlen zu berechnen, die den Roboter von seiner aktuellen Konfiguration zu einem gewünschten Zielzustand bringen und dabei Kollisionen mit Hindernissen und sich selbst vermeiden. Somit ist TG direkt zwischen der Aufgabenplanung – dem Problem, welche Aufgaben in welcher Reihenfolge ausgeführt werden sollen – und der Steuerung – dem Problem der Ausführung von Motorbefehlen – angesiedelt. Eine gute Lösung für TG muss schnell sein, um auf Änderungen in der Umgebung reagieren zu können, flexibel für verschiedene Zieldefinitionen und sicher. Die meisten Fortschritte im Bereich der Robotik in Bezug auf TG konzentrieren sich entweder auf Manipulatoren oder auf mobile Roboter. Die Kombination beider Systeme scheint jedoch für den Einsatz in von Menschen geteilten Umgebungen unumgänglich.

TG für mobile Manipulation wird häufig als Optimierungsproblem über einen endlichen Zeithorizont formuliert. Dieser Ansatz ist als Model Predictive Control (MPC) bekannt und wird im Rahmen von autonomen Fahren aufgrund seiner Machbarkeits- und Stabilitätsgarantien weit verbreitet eingesetzt. In Chapter 4 präsentieren wir eine Methode, um MPC auf mobile Manipulation anzuwenden. Die Methode formuliert das TG-Problem für die gesamte kinematische Kette und nutzt Free Space Decomposition (FSD) zur Kollisionsvermeidung. Dies führt zu ausreichend schnellen Steuerfrequenzen von 10 Hz, unabhängig von der Anzahl der Kollisionshindernisse in der Umgebung. Wichtig ist, dass dieser Ansatz eine gekoppelte Bewegung der mobilen Basis und des Manipulators ermöglicht. Dies ist vorteilhaft in Situationen, in denen die Synchronisation der beiden Teilsysteme entscheidend ist, wie beim Öffnen von Türen oder beim Ausweichen komplexer Hindernisse.

Trotz dieser Vereinfachungen in der Umgebungsdarstellung schränken die Rechenkosten die Anwendbarkeit von MPC auf die mobile Manipulation ein, da die Bewegung nicht wirklich als reaktiv betrachtet wird und verschiedene Komponenten wie Zielanziehung und Kollisionsvermeidung nicht leicht trennbar sind. Ein neuer Ansatz zu Receding-Horizon-Control ist die Formulierung als rein geometrisches Problem. Erste Erfolge in diesem Gebiet, einschließlich Cartesian Impedance Control (CIC) und Artificial Potential Fields (APF), führten zur Formulierung als Menge von dynamischen Systemen auf glatten Mannigfaltigkeiten im Konfigurationsraum. Optimization Fabrics (fabrics) vereinheitlichen solche Ideen, bietet Stabilitätsgarantien in statischen Umgebungen und ermöglicht ein hochreaktives Verhalten ähnlich wie bei einfachen Steuerungsalgorithmen.



Das Framework basiert auf nicht-riemannscher Geometrie, um eine glatte Mannigfaltigkeit des Konfigurationsraums mit individuellen Verhaltensweisen wie Kollisionsvermeidung, Gelenkgrenzenvermeidung und Zielanziehung zu gestalten. In Chapter 5 präsentieren wir eine Verallgemeinerung von fabrics auf dynamische Umgebungen. Wir bezeichnen den resultierenden Ansatz als Dynamic Fabrics (DF). Die Verallgemeinerung nutzt zeitparameterisierte Mannigfaltigkeiten zur Integration von sich bewegenden Hindernissen und zeitparameterisierten Referenztrajektorien. Letzteres ist besonders wichtig für TG über lange Zeiträume, die lokale Minima aufweisen können. Wichtig ist, dass in Chapter 5 gezeigt wird, dass die dynamische Komponente von DF notwendig ist, um mit beweglichen Hindernissen umzugehen. Da abstoßende Kräfte in fabrics proportional zur Annäherungsgeschwindigkeit von Hindernis und Roboter sind, ist eine pseudo-statische Kollisionsvermeidung nicht ausreichend, wenn sich der Roboter langsam bewegt.

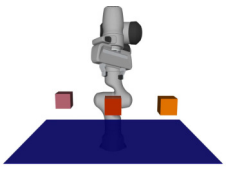
Schließlich setzen wir das allgemeine Framework von DF in mehreren realen Umgebungen ein, um die Anwendbarkeit auf die mobile Manipulation zu demonstrieren. Zuerst präsentieren wir eine Methode, um nicht-holonome Beschränkungen zu integrieren. Trotz des Verlusts formaler Konvergenzgarantien erweist sich die Methode als natürliche Erweiterung auf mobile Roboter mit nicht-holonomischen Einschränkungen. Zweitens wird in Chapter 6 eine symbolische Implementierung von fabrics vorgestellt, um höhere Steuerfrequenzen zu erreichen. Symbolische Implementierungen sind möglich, da fabrics auf Differenzialgleichungen zweiter Ordnung basieren, für die eine geschlossene Lösung existiert. Für sich ändernde Umgebungen werden die Hinderniszustände dann nur zur Laufzeit konkretisiert. Außerdem zeigen wir, dass symbolische Hyperparameter automatisch abgestimmt werden können, um eine Experten-Tuning-Niveau zu erreichen. Drittens integriert Chapter 7, um hohe Anforderungen an die Wahrnehmungspipeline zu überwinden, verschiedene implizite Umgebungsdarstellungen. Die Verwendung von Signed Distance Fields (SDF) und FSD wird beispielsweise häufig in der mobilen Robotik verwendet, wenn TG als MPC formuliert wird. Wir zeigen, dass dieselben Darstellungen in fabrics verwendet werden können und dabei schnellere Lösungszeiten erreicht werden. Schließlich wird in Chapter 8 ein mobiler Manipulator, der von fabrics gesteuert wird, in einem Supermarkt eingesetzt. Geschickte Manipulationen werden mittels kinestetischem Lernen programmiert, wobei fabrics als zugrunde liegende Kodierung verwendet wird. Dadurch wird es möglich, komplizierte Verhaltensweisen zu lehren, anstatt sie zu programmieren, während die Eigenschaften der Kollisionsvermeidung erhalten bleiben.

Diese Dissertation präsentiert Einblicke in verschiedene Aspekte der Bewegungsplanung, erweitert den Rahmen von fabrics für die TG und vergleicht ihn umfassend mit der häufiger verwendeten Methode der MPC. Mit den in dieser Dissertation vorgestellten Ideen hoffen wir, die Nutzung geometrischer Eigenschaften von Robotersystemen in von Menschen geteilten Umgebungen zu fördern. Dieser Ansatz bietet nicht nur eine reaktive TG, sondern kann auch als kompakte Kodierung von Trajektorien für zukünftige lernbasierte Methoden dienen.

1

INTRODUCTION

This chapter places this dissertation in the context of modern societies and their challenges. Specifically, we motivate this thesis by the demographic changes in the global North. In the process, we identify Trajectory Generation as a central problem in robotics, recall existing approaches and discuss their limitations. Finally, we present the contributions and the outline of this dissertation.



1

As the countries of the global North are struggling with the challenges of shifting demographics (Fig. 1.1) and a predominantly educated workforce resulting in a scarcity of affordable labor for physically demanding tasks, the need for technological solutions is more pronounced than ever [1]–[3]. Technological progress in the corresponding field of robotics today is disappointing, especially in an era where the non-embodied counterpart, i.e. speech recognition [4], text generation [5], and computer vision [6], is touching all of our lives. Specifically, strides in natural language processing and computer vision have significantly improved in the recent past and may rather sooner than later increase our productivity substantially. In contrast, embodied systems still lag behind in sophistication and are still mainly bound to industrial settings.

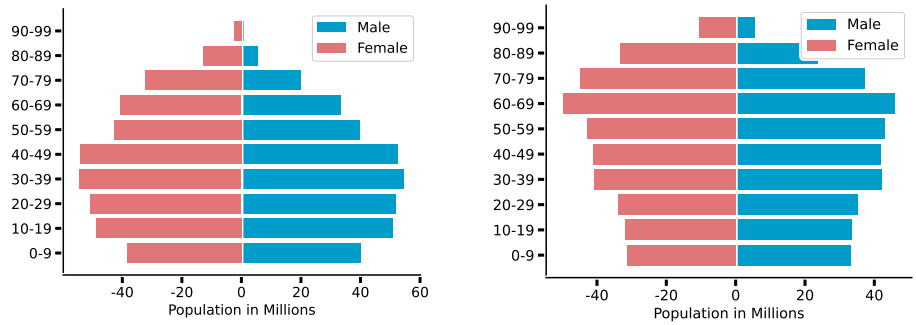


Figure 1.1: Population portions by age in 2000 (left) and predictions for 2050 (right) in Europe [1]. The aging society leads to labor shortage which may be counterbalanced with an increase in automation.

Industrial robots, see Fig. 1.2a are highly capable in certain tasks, but these systems are often unable to adapt to changes in real-time and lack the nuanced understanding and safety required for dynamic, human-shared environments. Collaborative robots promote safety through a lightweight hardware design and a broader set of sensors, see Fig. 1.2b. These robots are designed to exist safely alongside their human counterparts using different low-level control approaches.

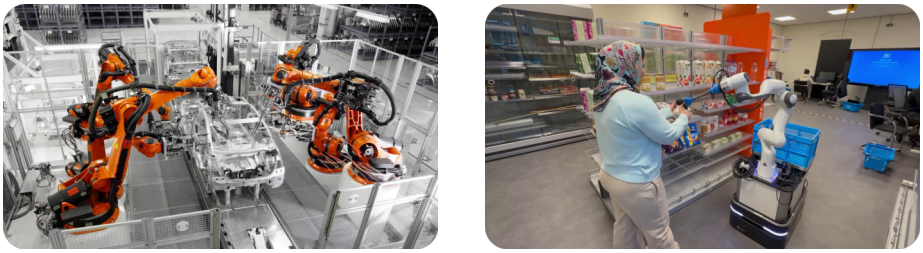
Next to safety, robots must be equipped with a similar level of mobility as humans to perform meaningful tasks. Mobility of industrial robots is limited as they are statically attached to structural elements. Placing robots in environments, that are built with the mobility of humans in mind, requires unlocking robots from their static sockets.

In summary, to help aging societies in dealing with labor shortage, the next generation of robots should be safe for humans while equipped with the same level of mobility.

MOBILE MANIPULATION

One key feature of modern robots, for human-shared environments, is mobility. The combination of highly mobile ground vehicles and manipulators is referred to as mobile manipulation. This concept enables robots to navigate *and* interact with their surroundings in ways previously deemed challenging. By endowing robots with the ability to move and manipu-

[1] United Nations, Department of Economic and Social Affairs, Population Division (2022)



(a) Industrial robot

(b) Collaborative robot

Figure 1.2: The difference between environments where robots used to live in (left) [2]and where we expect them to operate in the future (right, robot used in this thesis).

late objects in diverse environments, we further open avenues for addressing the mentioned societal challenges. Specifically, mobile manipulators could be used for tasks from warehouse operations, restocking shelves, and package delivery to intricate processes like food harvesting and service tasks at home. In short, mobile manipulation is a key concept if robots should perform similar tasks as humans in the same environments.

CHALLENGES IN HUMAN-SHARED ENVIRONMENTS

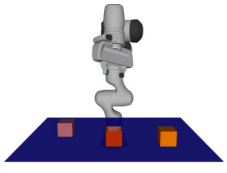
However, despite their potential, robots are noticeably absent from human-shared environments. The complexity of such spaces, coupled with safety constraints, presents formidable challenges. This is where Trajectory Generation (TG) becomes pivotal as one of the basic building blocks for robotics software stacks. It determines the commands sent to the motors in real-time, orchestrating a harmonious movement that brings the robot closer to its goal while ensuring the safety of itself and its environment.

While the hardware is ready for deployment, the missing link lies in methods for generating trajectories that prioritize safety while providing high success rates and short cycle times. This dissertation embarks on a journey to unravel this critical aspect of robotics, exploring innovative TG methods that not only unlock the potential of robotic hardware but also lead to a world where robots seamlessly coexist with humans in shared spaces. Importantly, the presented methods are all designed with mobile manipulation in mind, as the increased level of complexity of such systems may rule out some methods that are superior for either the mobile base or the manipulator.

1.1. LIMITATIONS OF CURRENT APPROACHES

In the quest for effective TG, various methods have been proposed, each with its set of advantages and drawbacks. The classical setup for robots – where they have proven to be quite effective – is behind fences. This setup simplifies TG dramatically as collision avoidance can be done during the installation process and trajectories can remain unchanged for years. Therefore, high computational costs for TG are negligible as it is only performed once during installation. However, their Achilles' heel lies in their incapacity to adapt to dynamic, changing environments.

[2]Image from https://www.vanch.net.



Recognizing this limitation has led to the rise of reactive TG methods, where *reactive* refers to the fact that TG is performed in real-time only considering local changes to the environment. We divide these methods into two categories: optimization-based approaches and control-based approaches. Both categories are heavily investigated in the literature.

OPTIMIZATION-BASED APPROACHES

With the aim of safe robots, many approaches are rooted in optimization problems, centered around an objective function and constraints that favor safety guarantees and optimality. Such approaches are often referred to as Model Predictive Control (MPC) schemes or receding horizon optimization [7]. While optimization-based methods have demonstrated success in autonomous driving applications, they stumble when confronted with real-time constraints in systems with a large number of degrees of freedom due to high computational costs [8]. This is also investigated in this thesis in Chapter 4.

In the age of deep neural networks, learning-based approaches have gained popularity as they promise similar performance to optimization-based techniques. After all, they solve the same problem, while changing the naming of the objective function from cost function to loss (or reward) function. Computational costs are often mentioned when criticizing such approaches, but as computational capacity increases rapidly and important advancements in natural language processing resulted from more capable hardware and larger datasets [4], it seems little justified. Their major drawback lies in their inability to integrate safety constraints in a principled manner, and they are often prone to overfitting specific use cases during training and thus lacking generalizability when confronted with new scenarios [9].

CONTROL-BASED APPROACHES

In the early days of robotics, potential field methods were popular in the context of TG. They are based on the idea of modeling the robot as a point in a potential field built in the robot's configuration space, where the goal is a minimum and obstacles are maxima [10], [11]. This approach is known to be computationally affordable but only covers the static components of the problem, and thus does not naturally integrate the dynamics of the robot.

Another notable contender from the same era is Cartesian Impedance Control (CIC), praised for its rapid responsiveness and perceived safety [12]. CIC models the point of interest on the kinematic chain, usually the end-effector, as a spring-damper-system attracted to the goal pose. In contrast to potential field theory, the motion of the robot is thus taken intro consideration. However, it falls short by not encompassing crucial components for collision-free TG, such as self-collision avoidance or joint-limit avoidance. Modeling the point of interest of the robot as a second-order differential equation raises the question whether all components of the TG problem can be handled in this way.

We can observe that optimization-based approaches focus on the practical side of TG and often ignore the underlying geometric structure [13]. For example, formulating the TG problem in the Euclidean work space of the end effector, as it is done in many optimization-based methods and learning approaches, is a simplification of the problem that ignores the configuration space of the robot. Effectively, this approach uses a Euclidean geometry. In contrast, the work space can also be modeled as a Riemannian (or even non-Riemannian)

manifold of the configuration space [14]. Then, other components, that live in different manifolds of the configuration space, can be seamlessly integrated into the TG problem [13]. This insight has led to a series of works where TG has been addressed purely geometrically [13], [15]–[19].

1.2. GEOMETRIC APPROACHES

Although methods that consider the geometry of the robot's configuration space are closely related to control-based approaches, we distinguish them because this thesis focuses mainly on non-Euclidean approaches. The most visual difference between Euclidean and non-Euclidean geometries is the concept of distance. Our understanding of distance is intuitively based on the Euclidean geometry, where the distance between two points is the length of the straight line connecting them. Formally, the metric tensor defining distance is independent of the position in the space. This is not the case in non-Euclidean geometries, where the metric tensor can depend on the position in space. As a consequence, the distance between two points is not necessarily the length of the straight line connecting them. This fundamental difference between Euclidean and non-Euclidean geometries can be utilized for TG. Specifically, we can *shape* the geometry in such a way that distances between points in the configuration space that are *desirable* are shorter than distances between points that are *undesirable*. Given the resulting geometry, a desired trajectory becomes equivalent to the shortest path in this geometry. Computationally, finding a shortest path in a geometry is often cheaper than solving an optimization problem, for geometries used in this thesis, the costs are dominated by one matrix inversion.

In this dissertation, we recall and extend a framework that allows to design TG by iteratively shaping a geometry in the configuration space. Specifically, individual desirerable behaviors are combined using summation. These behaviors are designed as special geometries for which stability properties are conserved during summation. This concept, elegantly formalized as Optimization Fabrics (fabrics), offers a nuanced understanding of TG. Informally speaking, the composition shapes the landscape on which the trajectory is then generated.

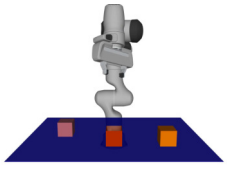
1.3. CONTRIBUTIONS

There are two lines of research in this thesis. Given the success of MPC in the field of autonomous driving and mobile robotics, we explore how this method can be adapted to mobile manipulation.

Part 1: Model Predictive Control

As manipulators and even more so mobile manipulators are characterized by a high number of degrees of freedom, this thesis investigates how MPC formulations known from autonomous driving must be adapted and how it performs in comparison to geometric methods.

As the results from the first part reveal that MPC is not competitive in terms of reactivity and computational costs, the second, and main part, of this thesis focuses on geometric methods. Specifically, we study the framework of Optimization Fabrics and its suitability



for mobile manipulation in dynamic environments.

Part 2: Optimization Fabrics

The main part of this thesis is dedicated to the exploration of the geometric framework of fabrics for mobile manipulation in dynamic environments. In that context, we investigate how the framework generalizes to dynamic environments, how global path planning can be integrated and how implicit environment representations, such as point clouds can be used for collision avoidance.

To address the challenges formulated above, several contributions are made in this thesis.

Mobile manipulation through MPC Chapter 4 presents an MPC formulation for whole body mobile manipulation. The presented method proposes to cope with poor scaling in the number of collision constraints, by using the concept of Free Space Decomposition (FSD). This method achieves a control frequency of 10Hz *independent* on the amount of collision obstacles in the environment.

Generalization of fabrics to Dynamic Fabrics (DF) Chapter 5 generalizes the framework of fabrics for dynamic environments. We refer to the resulting framework as DF. Specifically, we introduce time-parameterized manifolds to integrate moving obstacles and time-parameterized goal definitions. Technically, we prove that energy conservation is preserved when *pulling* time-parameterized components into the configuration manifold. Similarly, we prove convergence to time-parameterized goals under a simple construction condition. Quantitative comparisons between our method and the original framework of fabrics reveal that failures due to collision is reduced from 9/20 to 0/20 for a 7-Degree of Freedom (DoF) robot in a real-world experiment. In terms of path following, trajectories generated with DF achieve a by a factor of 2 reduced summed error when compared with the original framework of fabrics for a 7-DoF robot in simulation.

Symbolic implementation of fabrics As fabrics offer a closed-form solution to TG, we present a symbolic implementation of fabrics in Chapter 6. This reduces solver times to approximately 1ms for a 7-DoF robots, because the fabric composition, combining the individual behaviors, is pre-computed and not performed at runtime. The TG policy can be arbitrarily parameterized, and is thus suitable for parameter refinement at runtime. Chapter 6 proposes the use of Bayesian optimization for automated parameter tuning of fabrics, reducing the need for expert knowledge. We show that expert level tuning can be achieved without prior knowledge of the framework. Additionally, tuning can be transferred to different robots without substantial loss of performance.

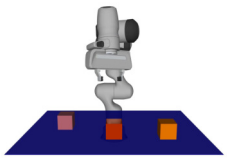
Implicit environment representations Chapter 7 integrates implicit environment representations into the framework of fabrics. This allows for the relaxation of the requirement for the perception pipeline, as the environment is represented implicitly in the policy. Specifically, we compare FSD, Signed Distance Fields (SDF) and raw point clouds as the input for collision avoidance.

Demonstration in a Real-world setting Chapter 8 demonstrates the usability of fabrics in a prototype application, specifically in the context of order-picking in supermarkets, showcasing real-world applicability and effectiveness. In this real-world scenario, we make use of the ability to arbitrarily formulate goals in different task spaces. The goal definition of pointing towards the shelf can then be very different to the goal definition of placing an item in the shopping basket.

These contributions collectively advance the field of TG for mobile manipulation, offering novel solutions to address key challenges and paving the way for future research and development efforts.

1.4. OUTLINE

The outline of this thesis is depicted in Fig. 1.3. After this introduction, Chapter 3 first introduces relevant literature in the field of TG and motion planning. Then, Chapter 2 summarize the required tools from optimal control and from differential geometry used in this thesis. Chapter 4 explores an MPC formulation for whole body control with a mobile manipulator. Specifically, we integrate FSD to cope with poor scaling in the number of obstacles. In Chapter 5, we derive a generalization of optimization fabrics to dynamic environments, including path following and avoidance with moving obstacles. Chapter 6 highlights the benefits of formulating TG in a symbolic way to improve real-time capabilities and allow for online parameter tuning. Chapter 7 showcases the integration of implicit environment representations into the framework of fabrics to relax the requirements for the perception pipeline. Chapter 8 showcases the usage of fabrics in a prototype application in the context of order-picking in supermarkets. Finally, Chapter 9 summarizes the findings, their potential impact on deploying robots in human-shared environments, discusses the main differences between the presented methods and gives some recommendations for future direction of research.



1

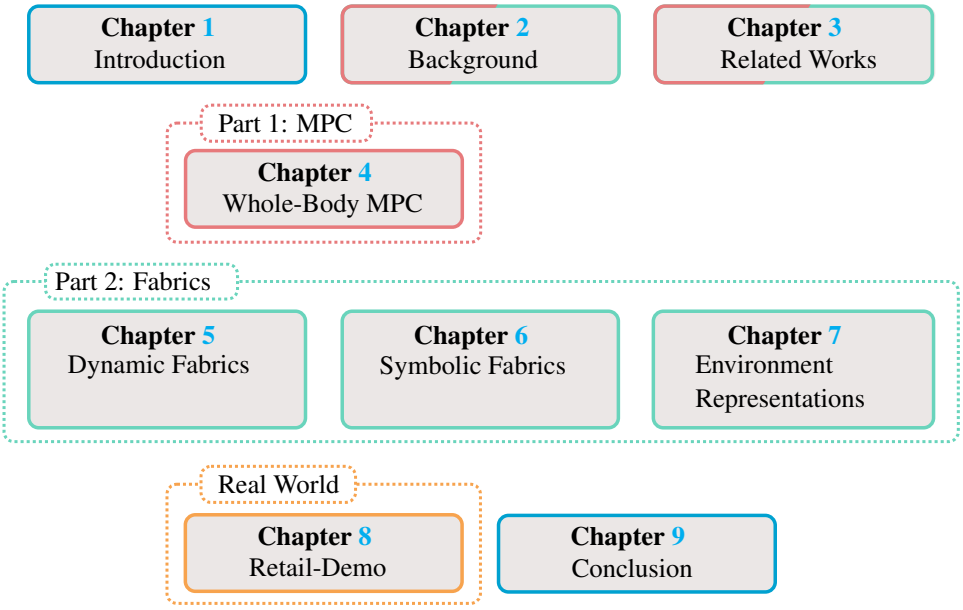


Figure 1.3: Outline of this thesis. General chapters are colored in blue, chapters related to MPC are colored in red, chapters related to fabrics are colored in green, and demonstrations are colored in yellow.

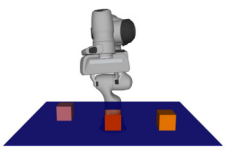
2

BACKGROUND

Having laid out the motivation for this thesis, we now proceed to present the background knowledge required to understand the contributions of this thesis. After the introduction of some general notations, we recall the Model Predictive Control (MPC) framework in a general time-discrete setting. Then, we state relevant concepts of differential geometry that are required in the aim of understanding Optimization Fabrics (fabrics). This will lead to the formal introduction of fabrics which is the main topic of this thesis. For an experienced reader, this chapter, or parts of it, may be skipped.

Parts of this chapter appeared in the following publications:

- ☞ **M. Spahn**, M. Wisse and J. Alonso-Mora. *"Dynamic Optimization Fabrics for Motion Generation"*. IEEE Transactions on Robotics, 2023.
- ☞ **M. Spahn** and J. Alonso-Mora. *"Autotuning Symbolic Optimization Fabrics for Trajectory Generation"*. IEEE International Conference on Robotics and Automation, 2023.
- ☞ **M. Spahn**, B. Brito and J. Alonso-Mora. *"Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition"*. IEEE International Conference on Robotics and Automation, 2021.



2

This thesis is concerned with motion planning for robots with a special focus on mobile manipulators. Motion planning is a fundamental problem that aims at finding feasible trajectories between an initial state and a goal state amidst changing environments. Importantly, the goal state might be defined in a task specific way. For example, the goal state might be defined in the task space (e.g., the end-effector position) or in the configuration space (e.g., the joint position). That places motion planning between task planning and control.

To relate this definition of motion planning to the literature, it can be further split into global path planning and Trajectory Generation (TG). The former is the process of computing a collision-free path from the initial to the goal state, while the latter is the process of computing a smooth trajectory that follows the path as closely as possible while satisfying the robot's kinematic and dynamic constraints [20]. In this thesis, we often evaluate methods for TG without the need for a global path planner. Instead of following a global path, the TG is then defined by the initial and goal state. We use *policy* as a general term to refer to a method that generates actions at each time step and thus solves the TG problem.

Modern research in robotics is increasingly focusing on robots that operate in dynamic environments. However, *dynamic environments* is a broad term that can refer to environments with a variety of different characteristics. In this thesis, we define a dynamic environment as an environment in which the robot's task is, generally, not known prior to deployment. In contrast to structured environments, such as production lines, those dynamic environments are semi-structured or even completely unstructured. These may contain moving obstacles, such as other robots, humans, or other unpredictable objects. Besides, goal configurations, such as goal locations, the size and shape of manipulation objects, may vary during deployment. Examples of such environments are supermarkets, hospitals, or homes, see Fig. 2.1.

Figure 2.1: Examples of dynamic environments: robots may be deployed in the hospital to assist the personnel [21] or in supermarkets, see Chapter 8.

2.1. NOTATIONS

Throughout this thesis, vectors are denoted in bold lowercase letters, $\boldsymbol{x}$, matrices in capital, $\boldsymbol{M}$, and sets in calligraphic uppercase, $\mathcal{M}$. In the context of discrete-time systems, we denote the time step using a subscript, e.g., $\mathbf{x}_k$ is the state at time k. The transpose of a matrix is denoted by a superscript ^{T}, the inverse by a superscript $^{-1}$, and the pseudo-inverse by a superscript †. Partial derivatives are denoted by $\partial_{\mathbf{x}}\mathbf{y}$ in text block, or more explicitly

as

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}.$$

$\|\mathbf{x}\|_M = \mathbf{x}^T M \mathbf{x}$ denotes the weighted squared norm.

We denote $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^n$ a configuration of the robot with n its degrees of freedom; $\mathcal{Q}$ is the configuration space of the generalized coordinates of the system. Generally, $\boldsymbol{q}(t)$ defines the robot's configuration at time t, so that $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ define the instantaneous time derivatives of the robot's configuration. Similarly, we assume that there is a set of task variables $\boldsymbol{x}_j \in \mathcal{X}_j \subset \mathbb{R}^{m_j}$ with variable dimension $m_j \leq n$. The task manifold $\mathcal{X}_j$ defines an arbitrary manifold of the configuration space $\mathcal{Q}$ in which a robotic task can be represented. Further, we assume that there is a smooth differential map $\phi_j : \mathbb{R}^n \to \mathbb{R}^{m_j}$ that relates the configuration space to the j^{th} task space. For example, when a task variable is defined as the end-effector position, then ϕ_j is the positional part of the forward kinematics. Conversely, if a task variable is defined to be the joint position, then ϕ_j is the identity function. In the following, we drop the subscript j in most cases for readability when the context is clear.

In this work, we assume that ϕ is twice differentiable so that the Jacobian is defined as

$$\boldsymbol{J}_\phi = \frac{\partial \phi}{\partial \boldsymbol{q}} \in \mathcal{R}^{m \times n}, \tag{2.1}$$

or $\boldsymbol{J}_\phi = \partial_{\boldsymbol{q}} \phi$ for short. Thus, we can write the total time derivatives of $\boldsymbol{x}$ as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_\phi \dot{\boldsymbol{q}} \tag{2.2}$$

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}_\phi \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{q}}. \tag{2.3}$$

2.2. MODEL PREDICTIVE CONTROL

Optimization methods, such as dynamic programming are popular when addressing TG problems that are characterized by multiple equality and inequality constraints. The optimization problem is composed of a cost function, a dynamic model of the system, and a set of constraints. This thesis uses exclusively discrete time dynamics, such that the dynamics of the system are given by

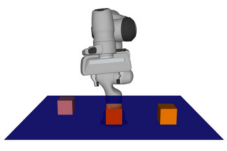
$$\boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k, \boldsymbol{u}_k), \tag{2.4}$$

where $\boldsymbol{z}_k \in \mathbb{R}^n$, $\boldsymbol{u}_k \in \mathbb{R}^m$ are the state and control input vectors, respectively.

Multiple objectives of the TG problem are combined in the cost function:

$$J(\boldsymbol{z}, \boldsymbol{u}) = \sum_{k=0}^{N-1} J_k(\boldsymbol{z}_k, \boldsymbol{u}_k) + J_f(\boldsymbol{z}_N, \boldsymbol{u}_N), \tag{2.5}$$

where N is the prediction horizon, and J_k and J_f are the stage and terminal cost, respectively.



The general MPC problem is then formulated as

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & J(\boldsymbol{z}, \boldsymbol{u}) \\ \text{subject to} \quad & \boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k, \boldsymbol{u}_k), \quad k = 0, \dots, N-1 \\ & g(\boldsymbol{z}_k, \boldsymbol{u}_k) \leq 0, \quad k = 0, \dots, N-1 \\ & \boldsymbol{z}_k \in \mathcal{Z}, \quad \boldsymbol{u}_k \in \mathcal{U}, \quad k = 0, \dots, N \\ & \boldsymbol{z}_0 = \boldsymbol{z}_{\text{init}}, \end{aligned} \tag{2.6}$$

where $\boldsymbol{z}_0$ is the initial state, $\mathcal{Z}$ and $\mathcal{U}$ are the admissible sets of states and control inputs, respectively. Inequality constraints are denoted by $g(\boldsymbol{z}_k, \boldsymbol{u}_k)$.

At each time step, the MPC problem is solved to find the sequence of control inputs $\boldsymbol{u}_k^*$ that minimize the cost function. The first control input of the optimal sequence is then applied to the system. The subsequent actions in the sequence are used as the initial guess for the next timestep. This process is usually referred to as receding horizon control [22].

2.3. DIFFERENTIAL GEOMETRY

In this section, we consciously use a simplified formalism for differential geometry to facilitate its understanding For a more thorough introduction to differential geometry, we refer to [23]–[26].

Differential geometry is the study of geometry using calculus. It is a mathematical discipline that uses the techniques of differential and integral calculus, as well as linear algebra, to study problems in geometry. In the context of this thesis, we recall some of the basic concepts of differential geometry that are required to understand the formalism of Optimization Fabrics.

2.3.1. MANIFOLDS

A manifold is a topological space that locally resembles Euclidean space near each point. More precisely, a manifold is a topological space M such that for each point $p \in M$, there exists a neighborhood U of p that is homeomorphic to an open subset of $\mathbb{R}^n$. The dimension of the manifold is the dimension of the Euclidean space to which it is locally homeomorphic.

A smooth manifold is a manifold that is equipped with a smooth structure. A smooth structure is a collection of charts that cover the manifold and are compatible with each other. A chart is a homeomorphism from an open subset of the manifold to an open subset of $\mathbb{R}^n$. The compatibility condition is that the transition maps between overlapping charts are smooth. A smooth map between two smooth manifolds is a map that is smooth in the sense that it is smooth in each chart.

2.4. OPTIMIZATION FABRICS

In this section, we introduce the concept of Optimization Fabrics (fabrics) which are a framework for TG that is based on the theory of differential geometry. The main idea is to

design TG as a second-order dynamical system. The trajectory generator is defined by the differential equations, as we will lay out in the following.

2.4.1. SPECTRAL SEMI-SPRAYS

Let us first define the base class for all that follows. Let $\mathcal{X}$ be a smooth manifold for which we denote vectors as $\boldsymbol{x} \in \mathcal{X}$ and $\dot{\boldsymbol{x}} \in T\mathcal{X}$ the tangent space of $\mathcal{X}$.

Then, a *spectral semi-spray* [18] is a second-order dynamical of the form $\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{f} = 0$, where $\boldsymbol{M}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ and $\boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ are functions of position and velocity. Besides, $\boldsymbol{M}$ is symmetric and invertible. We often denote *specs* using the shorter notation $\mathcal{S} = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$, where we drop the subscript when the space of the task variable is clear from the context. Then the trajectory is defined as the solution to the system $\ddot{\boldsymbol{x}} = -\boldsymbol{M}^{-1}\boldsymbol{f}$.

2.4.2. OPERATIONS ON SPECS

On *specs*, we can define two fundamental operations: the *summation* and the *pullback*.

Given two specs $\mathcal{S}_1 = (\boldsymbol{M}_1, \boldsymbol{f}_1)_{\mathcal{X}}$ and $\mathcal{S}_2 = (\boldsymbol{M}_2, \boldsymbol{f}_2)_{\mathcal{X}}$, their sum is defined as

$$\mathcal{S}_1 + \mathcal{S}_2 = (\boldsymbol{M}_1 + \boldsymbol{M}_2, \boldsymbol{f}_1 + \boldsymbol{f}_2)_{\mathcal{X}}. \tag{2.7}$$

Given a differential map $\phi : \mathcal{Q} \to \mathcal{X}$ and a spec $(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$, the *pullback* is defined as

$$\text{pull}_{\phi}(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}} = \left(\boldsymbol{J}_{\phi}^{T}\boldsymbol{M}\boldsymbol{J}_{\phi}, \boldsymbol{J}_{\phi}^{T}(\boldsymbol{f} + \boldsymbol{M}\dot{\boldsymbol{J}}_{\phi}\dot{\boldsymbol{q}})\right)_{Q}. \tag{2.8}$$

The pullback allows converting between two distinct manifolds (e.g. a spec could be defined in the robot's workspace and being pulled into the robot's configuration space using the pullback with ϕ being the forward kinematics).

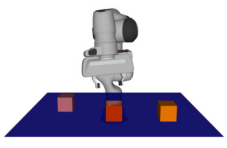
2.4.3. OPTIMIZATION FABRICS

As fabrics form a special class of specs, they inherit their properties, specifically the previously defined operations of *summation* and *pullback*. First, let us introduce a finite and differentiable potential function $\psi(\boldsymbol{x})$ defined in a task manifold $\mathcal{X}$. Then, the modified spec $\mathcal{S}_{\psi} = (\boldsymbol{M}, \boldsymbol{f} + \partial_{\boldsymbol{x}}\psi)$ is called the *forced variant* of $\mathcal{S} = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$. Only if the trajectory $\boldsymbol{x}(t)$ generated by the forced spec converges to the minimum of ψ, the spec is said to form an *optimization fabric*. When the spec only converges to the minimum when equipped with a damping term, $(\boldsymbol{M}, \boldsymbol{f} + \partial_{\boldsymbol{x}}\psi + \boldsymbol{B}\dot{\boldsymbol{x}})$, it forms a *frictionless fabric* [18, Definition 4.4]. Note that the mechanical system of a pendulum forms a frictionless fabric, as it optimizes the potential function defined by gravity when being damped (i.e., it eventually comes to rest at the configuration with minimal potential energy).

In the following, methods to construct fabrics are summarized.

2.4.4. CONSERVATIVE FABRICS AND ENERGIZATION

While the previous subsection defined what criteria are required for a spec to form an Optimization Fabric (fabric), the theory on conservative fabrics and energization offers a simple way of generating such special specs. As a full summary of the theory on fabrics



and their construction is out of scope here, this subsection only provides an outline of the theory and the reader is referred to [27], [28] for detailed derivations.

In the context of fabrics, the term *energy* describes a scalar quantity that changes as the system evolves over time. Although this quantity has a physical meaning in natural systems (e.g., kinetic energy), it can be arbitrarily defined for TG. Besides, this quantity can be used as a Lyapunov function to prove the convergence of the trajectory to the minimum of the potential function. Generally, specs and fabrics do not conserve an energy, but when they do, we call them *conservative specs*. A stationary Lagrangian [18, Definition 4.11] is one definition for an energy for which the corresponding spec, known as the Lagrangian spec $\mathcal{S}_{\mathcal{L}_e} = (\boldsymbol{M}_{\mathcal{L}_e}, \boldsymbol{f}_{\mathcal{L}_e})$, is obtained by applying the Euler-Lagrange equations. Importantly, Lagrangian specs conserve energy and do thus belong to the class of *conservative specs*. It was proven that an unbiased ([18, Definition 4.11]) Lagrangian spec forms a frictionless fabric [18, Proposition 4.18]. Such fabrics are analogously called *conservative fabrics*. There are two classes of conservative fabrics: Lagrangian fabrics (i.e., the defining energy is a Lagrangian) and the more specific subclass of Finsler fabrics (i.e., the defining energy is a Finsler structure [18, Definition 5.4]).

Definition 2.4.1. *A Finsler structure is a stationary Lagrangian $\mathcal{L}_g$ with the following properties:*

1. *Positivity:* $\mathcal{L}_g > 0 \quad \forall \dot{\boldsymbol{x}} \neq 0$,
2. *Homogeneity of degree* 1 *in velocity:* $\mathcal{L}_g(\boldsymbol{x}, \alpha\dot{\boldsymbol{x}}) = \alpha\mathcal{L}_g(\boldsymbol{x}, \dot{\boldsymbol{x}})$,
3. *For the energy* $\mathcal{L}_e = \frac{1}{2}\mathcal{L}_g^2$, *the tensor* $\partial^2_{\dot{\boldsymbol{x}}\dot{\boldsymbol{x}}}\mathcal{L}_e$ *is everywhere invertible.*

The operation of *energization* transforms a given differential equation into a conservative spec. Specifically, given an unbiased energy Lagrangian $\mathcal{L}_e$ with boundary conforming $\boldsymbol{M}_{\mathcal{L}_e}$ [18, Definition 4.6] and lower bounded energy $\mathcal{H}_e = \partial_{\dot{\boldsymbol{x}}}\mathcal{L}_e^T\dot{\boldsymbol{x}} - \mathcal{L}_e$, an unbiased spec in normalized, i.e. $\boldsymbol{h} = \boldsymbol{M}^{-1}\boldsymbol{f}$, form $\mathcal{S}_{\boldsymbol{h}} = (\boldsymbol{I}, \boldsymbol{h})$ is transformed into a frictionless fabric using energization as

$$\begin{aligned} S_{\boldsymbol{h}}^{\mathcal{L}_e} &= \text{energize}_{\mathcal{L}_e}\{S_{\boldsymbol{h}}\} \\ &= (\boldsymbol{M}_{\mathcal{L}_e}, \boldsymbol{f}_{\mathcal{L}_e} + \boldsymbol{P}_{\mathcal{L}_e}[\boldsymbol{M}_{\mathcal{L}_e}\boldsymbol{h} - \boldsymbol{f}_{\mathcal{L}_e}]), \end{aligned} \tag{2.9}$$

where $\boldsymbol{P}_{\mathcal{L}_e} = \boldsymbol{M}_{\mathcal{L}_e}\left(\boldsymbol{M}_{\mathcal{L}_e}^{-1} - \frac{\dot{\boldsymbol{x}}\dot{\boldsymbol{x}}^T}{\dot{\boldsymbol{x}}^T\boldsymbol{M}_{\mathcal{L}_e}\dot{\boldsymbol{x}}}\right)$ is an orthogonal projector. Energized specs maintain the energy of the Lagrangian and generally change the trajectory of the underlying spec $\mathcal{S}_{\boldsymbol{h}}$. However, if

1. $\mathcal{S}_{\boldsymbol{h}} = (\boldsymbol{I}, \boldsymbol{h})$ is homogeneous of degree 2,

$$\boldsymbol{h}(\boldsymbol{x}, \alpha\dot{\boldsymbol{x}}) = \alpha^2\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) \tag{2.10}$$

   and

2. the energizing Lagrangian is a Finsler structure,

the resulting energized spec forms a frictionless fabric for which the trajectory matches the original trajectory of $\mathcal{S}_{\boldsymbol{h}}$. We refer to energized fabrics with that property as *geometric*

fabrics. Geometric fabrics form the building blocks for TG with fabrics. Practically, energization equips the individual components of the planning problem with a metric when being combined with other components.

2.4.5. TRAJECTORY GENERATION USING OPTIMIZATION FABRICS

After this brief introduction to the theory of fabrics, we can lay out the general procedure for TG with the framework. When using fabrics for TG, all components including constraints and goal attraction are designed as second order differential equations. If specific design rules for these equations are respected, all components can be combined to form a converging trajectory generator. Specifically, the following steps are performed:

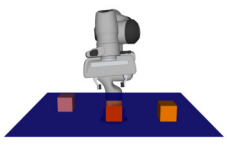
1. Design path-consistent geometries in suited manifolds of the configuration space (Eq. (2.10)).
2. Design corresponding Finsler energies defining the importance metric in this manifold (Section 2.4.4).
3. Energize all geometries with the associated Finsler energies (Section 2.4.4).
4. Pull back the energized systems into the configuration space and sum them (Section 2.4.2).
5. Force the combined system with a differentiable potential. As a composition of optimization fabrics, the resulting trajectory converges towards the potential's minimum (Section 2.4.3).

This procedure has been initial presented in [18] and is used throughout this thesis to generate trajectories with fabrics. In Chapter 5, we adopt the procedure to our generalization to dynamic settings.

Although important concepts and findings on fabrics were summarized in this section, we refer to [18] for a more in-depth presentation of fabrics. In the following, we generalize the framework of **fabricss!** (**fabricss!**) to dynamic settings.

2.5. CONCLUSION

In this chapter, we formalized the environments in which we want to deploy the presented methods. Then, the problem of TG was placed between task planning and control. Afterward, we formalized a vanilla MPC formulation for time-discrete systems. Finally, we recalled the required concepts from differential geometry and introduced the framework of fabrics for TG.



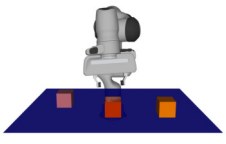
3

Related Works

In the previous chapter, we have defined Trajectory Generation (TG) as the problem of finding a sequence of actions that will move the robot from its current state to a desired state. As this thesis focuses on TG for mobile manipulation, relevant literature must therefore include works from different fields, ranging from autonomous driving and mobile robotics to manipulation. Besides, TG in dynamic environments generally includes path planning and reactive TG. Therefore, this chapter summarizes approaches ordered on a scale of reactivity, that is the frequency at which trajectories are computed. Starting with controller-like TG methods, we then move in order of increasing time-horizon up until global path planning.

Parts of this chapter appeared in the following publications:

- ☞ **M. Spahn**, M. Wisse and J. Alonso-Mora. *"Dynamic Optimization Fabrics for Motion Generation"*. IEEE Transactions on Robotics, 2023.
- ☞ **M. Spahn** and J. Alonso-Mora. *"Autotuning Symbolic Optimization Fabrics for Trajectory Generation"*. IEEE International Conference on Robotics and Automation, 2023.
- ☞ **M. Spahn**, B. Brito and J. Alonso-Mora. *"Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition"*. IEEE International Conference on Robotics and Automation, 2021.



Motion planning methods aim at finding connections between two robot states that are collision-free at the moment of planning and potentially optimal. In the context of changing environments, it is useful to place different methods on a scale of reactivity, usually measured by the achieved compute frequency, see Fig. 3.1. At the most reactive end, we find low level controllers and on the least reactive end, we find global path planning methods. In most robotics planning pipelines, a global path planner is guiding a more reactive TG method. This thesis focuses on the reactive end of the scale, and therefore we will only give a brief overview of global path planning methods.

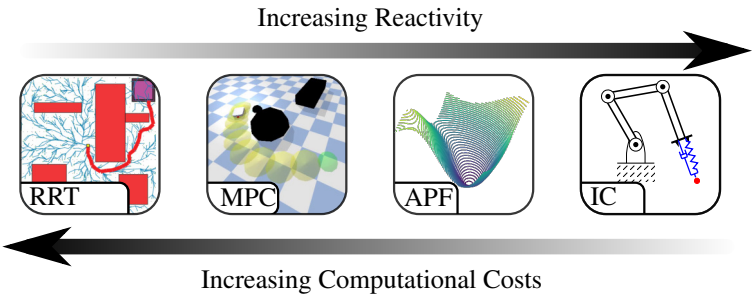


Figure 3.1: Reactivity scale of different motion planning methods revised in this thesis. Global methods, such as Rapidly-exploring Random Tree (RRT) [29] are at the least reactive end. More reactive methods, such as Model Predictive Control (MPC) [7], Artificial Potential Fields (APF) [30] and low-level control methods such as Impedance Control (IC) [12] are towards the more reactive side.

3.1. Control-based approaches

Control-based approaches to TG are methods that control the robot using the robot's current state *without* taking into account the future development. The simplest form of a control-based approach is a proportional controller whereas more involved methods solve an optimization problme taking into account the robot's kinematics and dynamics.

3.1.1. Impedance control

IC is a widely used control method in robotics [12], [31]. In IC, the controlled system is modeled as a mass-spring-damper system. The specific case of Cartesian Impedance Control (CIC) was proposed in [32] and could be characterized as a TG method, because of its ability to cope with various goal specifications and its safety properties in the proximity of humans [33], [34]. Additionally, IC can be used in collaborative settings, e.g., when lifting large objects together with humans [31]. More recent works on IC use variable impedance parameters to adapt the robot's behavior to the task at hand [31]. Often, the parameters are modified by human feedback to improve perceived safety and increase reliability of the task execution with human feedback [35], [36]. IC is a method for TG, that does not take into account the environment, such as obstacles or other agents.

3.1.2. Reactive control

In this dissertation, we refer to *reactive TG* as methods that rely on local information about the environment, including obstacles and other agents, and the robot that compute actions

without a time horizon into the future. The most prominent example of reactive TG is the Artificial Potential Fields (APF) method as we will see by visiting the different embodiments of robotic systems.

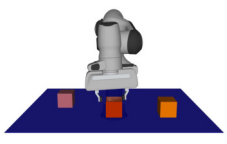
Robotic arms Pioneer works on TG for robotic arms employed APF for collision avoidance [30], [37], [38]. Building on the previous, [39] introduced the Circular Field method to address dynamic collision avoidance. To ensure collision avoidance for the end-effector when grasping a moving obstacle, [40] employed a repulsive vector. Velocity scaling of trajectories in the presence of contact forces was addressed in [41]. Reactive TG can also be formulated as an optimization problem. In this case, the optimization problem is solved at each time step. While statements about global optimality are not possible, different objectives can be encoded in the cost function and avoidance can be integrated using constraints. By maximizing manipulability, the robot remains flexible to changes in the environment while reaching a goal pose [42]. The work was later extended to integrate moving and static obstacles [43].

Mobile robots The method of APF was also used for TG of mobile robots after its introduction for robotic arms. For non-holonomic mobile robots, it was shown that APF generally exhibits multiple equilibrium points which are not necessarily stable [44]. Different to APF, the dynamic window approach [45] and its new variant proposed in [46] have proven to be efficient in generating smooth trajectories for mobile robots in static and dynamic environments. To navigate among pedestrians, [47] introduced the Social Forces model imitating the human navigation behavior and using it as navigation policy for the robot. Yet, Social Forces and its variants rely on handcrafted functions limiting their ability to handle more complex navigation scenarios. To deal with many agents, *ORCA* was proposed in [48] and later extended for non-holonomic bases in [49].

While these reactive TG methods capture more information than pure controllers, they often rely on scalar objectives functions to weigh different components.

3.1.3. GEOMETRIC APPROACHES

Reactive TG approaches, such as APF theory, may lead to contradicting behaviors, as individual policies are *fighting* against each other [15]. To overcome this shortcoming, some geometric approaches make an explicit distinction between the importance metric and the policy of individual behaviors. The metric can often be interpreted as a measure of curvature on the optimization manifold. The tuple of metric and policy defines for each *behavior* a dynamical system. Formulating TG as a dynamical system is a widely used approach [50], [51]. The concept of *modulation* of dynamic systems multiplies the system with matrix that captures directional preferences when avoiding obstacles or imitating human demonstrations [52], [53]. Similarly, Dynamic Movement Primitives (DMPs) formulate TG as a known dynamical systems with convenient stability properties for which parameters can be learned or tuned [54], [55]. The chosen dynamical systems are usally damped spring models (similar to IC) and focus on attraction to the goal, [56]. When limited to damped spring models in one task space, DMPs ared reduced to Operational Space Control (OSCs) [57].



A more general framework was introduced as Riemannian Motion Policies (RMPs) in [15], [16], [58]. RMPs represent a natural way of combining multiple policies into one joined policy. RMPs define individual sub-tasks of the motion planning as differential equations (*spectral semi sprays* or *specs* for short) of second order. Importantly, RMPs make an explicit separation between metric and forcing vector, thus being closely related to the concept of modulation of dynamical systems [53]. After defining all desired behaviors as tuples of metric and forcing vector, they are combined using *pullback* and *summation* operations in the configuration space, see Fig. 3.2. As subtasks can be defined in arbitrary manifolds of the configuration space, RMPs generalize operational space control [59]. The resulting behavior of RMPs was reported to be intuitive while keeping computational costs low [15]. The concept of RMPs was used in [16], [17] to form RMP-Flow, a motion planning algorithm that is shown to be conditionally stable and invariant across robots. An RMPs adaptation was proposed for non-holonomic robots in [60]. By incorporating the kinematic constraint into the root equation of the RMPs, the computed policy is applicable to non-holonomic robots. Besides, that work proposed a neural net to learn the collision avoidance task components.

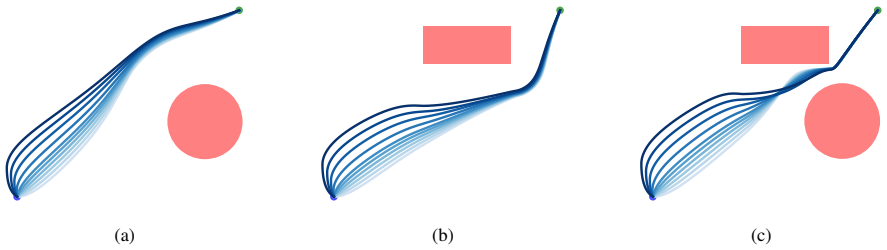


Figure 3.2: Combining different avoidance behaviors using optimization fabrics. The components defining collision avoidance with single obstacles (a,b) are combined in (c). Obstacles are shown in red. Trajectories of the point robot with various initial velocity vectors are shown in shades of blue.

Although RMPs have proven to be a powerful tool for TG, it was reported to require intuition and experience during tuning [18]. Optimization Fabrics (fabrics) with Finsler structures as metric generators simplify the motion design as the conditions for stability and convergence are inherent to the definition of Finsler structures [18], [27], [61], [62]. Opposed to RMPs, where the metric is typically user-defined, fabrics derive Finsler metrics from artificial energies, similar to approaches from control design, [63], [64], using the Euler-Lagrange-Equation from geometric mechanics.

The theory of fabrics was tested on several simple kinematic chains in [18], [27]. As fabrics design motion as a summation of several differential equations, each representing a specific constraint to the motion, it is possible to sequentially design motion [18]. This procedure allows to carefully tune individual components without harming the others. The application to a planar arm in a goal-reaching setup was successfully tested in [18]. There, the authors illustrated how the resulting motion can be modified arbitrarily by the user by adding additional constraints or preferences. Although fabrics generalize the concept of RMPs and make it accessible to a broader audience by decreasing the intuition and expertise required,

they have not yet been applied to a wide range of robots.

The reason for this lack of application of fabrics is twofold. First, all the above-mentioned methods are reactive and highly local methods, thus making them prone to local minima [65]. As RMPs and fabrics do not incorporate path following, integration of global path planning to overcome local minima is not possible to this date. Second, fabrics and RMPs do not make use of velocity estimates of obstacles but rely purely on their high reactivity in dynamic environments. As for other TG approaches, motion estimates could benefit fabrics (and RMPs) to result in even smoother motion and allow applications in such environments.

3.2. RECEDING-HORIZON TRAJECTORY OPTIMIZATION

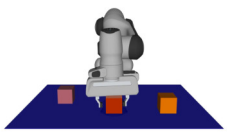
Acknowledging the limitations of reactive methods, TG is formulated as an optimization problem over a finite discrete time horizon. Such methods are known under the name of receding-horizon trajectory optimization. In line with most literature in robotics, we will refer to such methods as Model Predictive Control (MPC). Generally, several objectives are encoded in the scalar objective function, dynamics are formulated as equality constraints and inequality constraints ensure collision avoidance and joint limit avoidance. The dynamics for this problem can include the full dynamics model or simple integration schemes [7]. By explicitly solving the constrained optimization problem, this approach yields formal guarantees on stability. Stability for MPC is proven by formulating an appropriate Lyapunov function and showing that the finite time-horizon formulation with an appropriate terminal cost results in the same stability as the corresponding infinite time-horizon formulation [66]–[68].

Despite these results, formal stability guarantees in such environments are challenging, as appropriate terminal cost functions are often not commutable or too conservative. Besides, the computational costs scale with the degrees of freedom, limiting update frequencies on real systems and consequently achieve lower reactivity than purely geometric methods [69].

Some MPC formulations are non-linear and can be analyzed using methods from non-linear control. When analyzing non-linear control system, Riemannian energies lead to more detailed stability results than Lyapunov functions. By investigating the variation around the generated trajectory and its contracting towards the desired trajectory, some control designs show exponential stabilizing properties [63]. These findings have been applied to tracking control problems [64].

Mobile robots Early works on MPC for mobile robots focus on goal reaching in static environments [22]. Acknowledging the importance of global path following, most MPC formulations rely on contour errors in the objective function to ensure minimal tracking errors [70], [71].

MPC schemes were proposed for mobile robots and autonomous vehicles in [72] and [73] allowing to optimize over a prediction horizon and avoid, in advance, dynamic obstacles. Several 3D MPC formulation were proposed for drones to enable safe motion through cluttered environments [74], [75].



Robotic arms Early implementations of MPC for industrial robotics arms were presented in [76]. Especially in the context of robotic arms, uncertainty in the robot model might harm the performance. To overcome this limitation, Gaussian Processes were proposed in [77], [78] to offline learn the mismatch between model and real system. The learned model was then used to adapt the MPC scheme during runtime [78].

Mobile Manipulators In the context of mobile manipulation, less research focused on collision avoidance in dynamic environments, including changing scenes and moving obstacles. A real-time controller using MPC was presented in [79], in which either a holonomic or a non-holonomic base was combined with a two-degree-of-freedom robotic arm mounted onto the base. Although hardware constraints were respected, no collision avoidance was considered. An MPC formulation for mobile manipulators with holonomic bases that allows collision avoidance was presented in [80]. The perceived obstacles were translated into a set of spheres to be respected by the MPC scheme. The proposed approach used dynamically changing weights to change between arm motion and locomotion, resulting in a locked arm during navigation. A different weight setting was used to perform motion underneath a horizontal bar with an a priori position. The work is extended to non-holonomic bases and includes object detection in moving underneath the horizontal bar [81].

Conclusion Receding-horizon trajectory optimization methods are a powerful tool for TG in dynamic environments. Some realizations provide formal stability guarantees and can be analyzed using methods from non-linear control. Therefore, MPC approaches are widely used in the context of autonomous driving. For manipulators, model inaccuracies degrade performance and learning methods are often used to compensate for this. However, the computational costs scale with the degrees of freedom and often limit the update frequency of the control loop, effectively reducing the reactivity of the system.

3.3. Path Planning

Past works devoted to path planning can be divided into two main categories: optimization-based and sampling-based methods [82], [83]. Sampling-based path planners generate random configurations until a valid path between an initial configuration and a set of goal configurations is found [29]. Sampling-based methods, such as Rapidly-exploring Random Tree (RRT) [84]–[86] and Probabilistic Roadmap (PRM) [87], [88] are highly efficient at generating paths for systems with high-dimensional configuration spaces.

Robotic arms Motion planning problems are usually defined by goals in arbitrary task spaces, such as the 3D Euclidean space or end-effector poses. For mobile robots, where task space and configuration space are often identical [82], the mapping from task space to configuration space is straightforward. However, in the context of manipulation, tasks can be regarded as constraints to the motion planning problem. Conventional approaches to motion planning rely on inverse kinematics to transform task constraints into sets of configurations. There is abundant research on solving the resulting path planning problem in the configuration space [20], [82]. This thesis focuses on methods to solve the TG in arbitrary task spaces. Therefore, this section is limited to sampling-based methods that directly

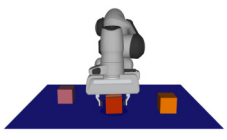
integrate tasks into the sampling process, rather than relying on the inverse kinematics.

Several methods have been proposed to directly integrate task constraints into the sampling phase. [89] proposed a method to iteratively push a random sample to the manifold adhering to the task constraint. The notion of task constraints was later extended to task space regions to define soft constraints for individual task components [90]. [91] proposed scalar-valued functions to represent task constraints for sampling-based planning. As all the above-mentioned methods rely on implicitly constrained sampling in the joint space, they exhibit high computational time, which is especially harmful to real-world applications [92].

Mobile Manipulators Despite abundant research in trajectory planning for mobile robots and robotic arms, few works focused on coupling both systems' control. It was shown that coupling the base and the robotic arm motion leads to a considerable reduction of total operational time and smoother motions [93], [94]. Nevertheless, these methods were designed for static environments and did not allow real-time collision avoidance. Furthermore, trajectory planning for the coupled system is a precondition for effective interactive navigation, including opening doors [95], [96] or moving obstacles out of the way [97].

3.4. CONCLUSION

To summarize, motion planning in robotics is a well-studied problem. In the early days of robotics, where robots were mostly deployed in static environments, motion planning was limited to global path planning. However, when exposed to dynamic environments, global path planning becomes insufficient due to its inability to quickly react to changes in the environment. Therefore, methods for solving the TG problem in dynamic environments are required. Such methods compute sequences of actions based on changes in the environment to ensure collision avoidances and adapation to changing goal definitions. Geometric methods, such as fabrics can achieve high reactivity and smooth motions whereas optimization based methods, such as MPC, excel when formal guarantees are required.



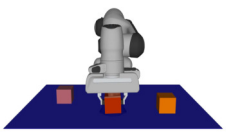
4

Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition

Now that we have delved into the crucial literature and established foundational notations and concepts for trajectory generation, this chapter involves the application of Model Predictive Control (MPC) to a mobile manipulator. This chapter introduces an implicit environment representation for MPC, demonstrating robustness and scalability with an increasing number of obstacles. Unlike the conventional approach of encoding each obstacle as a constraint in the optimization problem, we opt for a polygonal representation delineating the free space around the robot, thereby confining constraints to a fixed number. Additionally, a method for circumventing moving obstacles with mobile manipulators is devised through whole-body control, also referred to as coupled motion. The efficacy of the proposed approach is validated through simulation and real-world experiments.

This chapter is a verbatim copy of the peer-reviewed publication:

- ☞ **M. Spahn**, B. Brito and J. Alonso-Mora. *"Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition"*. IEEE International Conference on Robotics and Automation, 2021.



22.32994pt

4.1. INTRODUCTION

Mobile manipulation is the field of robotics in which a mobile robot's locomotion ability is combined with the manipulation ability of a robotic arm. However, conventional trajectory optimization methods dealing with such systems often dramatically reduce flexibility by decoupling planning for the base and the arm. Motion for both parts then need to be synchronized or are executed sequentially. Both synchronization and sequencing limit the ability of the robot to perform complex tasks, and it was shown that decoupled and sequenced approaches show significantly higher operational times [93], [94]. Yet, solving the whole-body trajectory optimization problem is challenging due to a large number of degrees of freedom (e.g., 10 for the robot used in our experiments and shown in Fig. 4.1). Moreover, dynamic and unstructured environments have not been addressed in a coupled approach yet. Such environments have been extensively investigated for autonomous vehicles (e.g., mobile robots navigating through human crowds [98], [99], drones in cluttered environments [74], [75]). In the context of autonomous vehicles, model predictive control (MPC) can effectively incorporate future evolutions of the environment [72]. A coupled MPC scheme for mobile manipulators was introduced in [80], where locomotion and manipulation were softly decoupled through dynamic weight-setting. Dynamic obstacles were not considered, and the method suffered from increasing computational costs as the environment becomes more densely populated [81]. More specifically, collision avoidance in an unstructured environment typically results in many inequality constraints that scale with the number of obstacles.

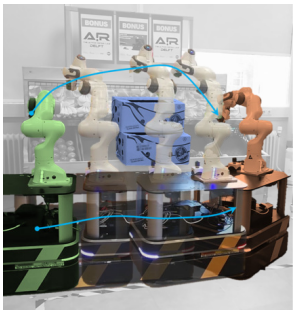


Figure 4.1: Mobile manipulator performing a pick & place task using whole-body trajectory optimization with our coupled MPC formulation.

In this work, we propose a whole-body trajectory optimization, sometimes referred to as MPC, using convex region decomposition of the free space on multiple kinematic chain links for collision avoidance with static obstacles. As a result, the number of inequality constraints remains constant regardless of the number of obstacles, allowing continuous control of the arm and the base to navigate through unstructured environments. Dynamic sphere-shaped obstacle avoidance is included using stage-dependent sphere-to-sphere inequality constraints.

Experimental results demonstrated that operational time is reduced considerably when using whole-body trajectory optimization. Computational costs for the optimization solver were independent of the number of obstacles present in the environment. Single dynamic obstacles moving at a constant velocity could be avoided successfully.

4.2. METHODS

4.2.1. MOBILE MANIPULATOR'S REPRESENTATION

Let us consider a velocity controlled mobile manipulator consisting of a mobile base with a mounted robotic arm. The coupled system's dynamics are described by the discrete-time non-linear system

$$\boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k, \boldsymbol{u}_k), \tag{4.1}$$

where $\boldsymbol{z}_k$ and $\boldsymbol{u}_k$ describe the state and the control inputs at the time-step k, respectively. The robot's state is the base position and orientation, and the manipulator's joint positions, $\boldsymbol{z} = [x, y, \theta, \mathbf{q}^{\text{arm}}]$. The robot's control inputs are the left u^l and right u^r wheel velocity, and joint velocities $\dot{\mathbf{q}}^{\text{arm}}$, hence $\mathbf{u} = [u^l, u^r, \dot{\mathbf{q}}^{\text{arm}}]$. We denote $\mathcal{Z}$ and $\mathcal{U}$ as the corresponding state and control commands admissible sets, respectively. The space occupied by the robot is denoted as $\mathcal{B}(\boldsymbol{z}) = \bigcup_{i \in \{1,\dots,n_{\text{links}}\}} \mathcal{B}_i(\boldsymbol{z})$ and $\mathcal{B}_i(\boldsymbol{z})$ denotes the space occupied by the i-th robot's link with $i \in [0, N_{\text{links}}]$. We approximate the state occupied by each link by spheres with radius r_i. The space occupied by the static obstacles and dynamic obstacles is represented as $\mathcal{O}^{\text{static}}$ and $\mathcal{O}^{\text{dynamic}}$, respectively. To limit the complexity of the problem we only consider a limited number $n_{\text{links}}\mathcal{L}_e N_{\text{links}}$ of the robot's links and the n_{dyn} closest dynamic obstacles.

4.2.2. MOBILE MANIPULATOR'S MODEL

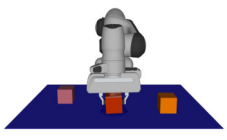
Here, we assume a differential drive model for the base and first order dynamics for the robotic manipulator:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \frac{1}{2} cos(\theta)(u^l + u^r) r_{\text{wheel}} \\ \frac{1}{2} sin(\theta)(u^l + u^r) r_{\text{wheel}} \\ \frac{(u^r - u^l) r_{\text{wheel}}}{L_{\text{wheel}}} \\ \dot{\boldsymbol{q}}_{\text{arm}} \end{bmatrix}, \tag{4.2}$$

where r_{wheel} and L_{wheel} are the wheel radius and distance between the two controllable wheels, respectively. The discrete transition function $f(\boldsymbol{z}_k, \boldsymbol{u}_k)$ can be found using a discretization scheme (e.g. Backward-Euler or Runge-Kutta). The set of admissible states ($\mathcal{Z}$) and control inputs ($\mathcal{U}$) is defined by the joint position and velocity limits

$$\begin{aligned} \boldsymbol{q}_{\text{min}} \leq \boldsymbol{q}_{\text{arm}} \leq \boldsymbol{q}_{\text{max}} \\ \boldsymbol{u}_{\text{min}} \leq \boldsymbol{u}_{\text{wheels}} \leq \boldsymbol{u}_{\text{max}} \\ \dot{\boldsymbol{q}}_{\text{min}} \leq \dot{\boldsymbol{q}}_{\text{arm}} \leq \dot{\boldsymbol{q}}_{\text{max}}. \end{aligned} \tag{4.3}$$

where $\mathbf{q}_{\text{min}}$ and $\mathbf{q}_{\text{max}}$, $\mathbf{u}_{\text{min}}$ and $\mathbf{u}_{\text{max}}$ and, $\dot{\mathbf{q}}_{\text{min}}$ and $\dot{\mathbf{q}}_{\text{max}}$ are the minimum and maximum joint position position, wheel velocity limits and joint velocity limits, respectively.



4.2.3. OPTIMIZATION PROBLEM

Consider that a reference path in $\mathbb{R}^2$ is provided for the base, denoted as a sequence of M waypoints, $([x, y]_m)_{m=0}^{M}$. The goal is to generate feasible control commands for the whole mobile manipulator enabling it to track the provided path while avoiding collisions in 3D space with dynamic and static obstacles. Hence, we formulate the trajectory planning problem for the unified system, base plus arm, as an optimization problem. As a result, we can explicitly formulate collision avoidance and kinodynamic constraints and compute control commands to generate feasible and collision-free motions. The optimization problem is formulated as

$$J^\star = \min_{\boldsymbol{z}_{0:N}, \boldsymbol{u}_{0:N}} \sum_{k=0}^{N} J(\boldsymbol{z}_k, \boldsymbol{u}_k), \tag{4.4a}$$

$$s.t. \quad \boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k, \boldsymbol{u}_k) \quad \forall k < N, \tag{4.4b}$$

$$\mathcal{B}(\boldsymbol{z}_k) \cap \left(\mathcal{O}^{\text{static}} \cup \mathcal{O}^{\text{dynamic}}\right) = \emptyset, \tag{4.4c}$$

$$\boldsymbol{u}_k \in \mathcal{U}, \boldsymbol{z}_k \in \mathcal{Z}, \tag{4.4d}$$

$$\boldsymbol{z}_0 = \boldsymbol{z}(0). \tag{4.4e}$$

In this formulation, Eq. 4.4a is the cost function (Section 4.2.4), Eq. 4.4b represents the kinodynamic constraints of the system (Section 4.2.2), Eq. 4.4c formalizes collision avoidance with static (Section 4.2.5) and dynamic obstacles (Section 4.2.6), and Eq. 4.4d defines the set of admissible states and control inputs (Section 4.2.2). Finally, Eq. 4.4e defines the initial state conditions. Note that we optimize over a prediction horizon N allowing to avoid dynamic obstacles in advance.

4.2.4. COST FUNCTION

To track the reference path, we first create a continuous path representation by approximating the provided reference path using a cubic *Bézier Curve* and using a normalized time parametrization, $\phi_k \in [0, 1]$. We denote $\boldsymbol{p}_k$ and θ_k as the predicted base position and orientation at time step k and $\boldsymbol{p}_k^r(\phi_k) = [x, y]_k^r$ and $\theta_k^r(\phi_k)$ as the reference base position and orientation at future time-step k, respectively. Then, we define a tracking error vector $\boldsymbol{e}_k := [e^c(\boldsymbol{z}_k, \phi_k), e^l(\boldsymbol{z}_k, \phi_k)]^{\mathrm{T}}$ composed by the contour $e^c(\boldsymbol{z}_k, \phi_k)$ and a lag-error $e^l(\boldsymbol{z}_k, \phi_k)$, and computed as it follows

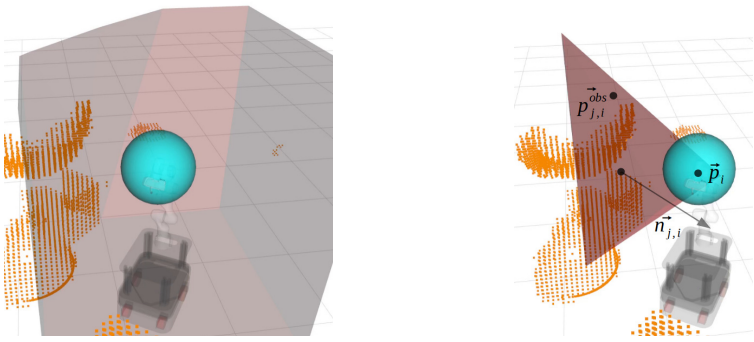
$$\boldsymbol{e}_k = \begin{bmatrix} cos(\theta_k^r) & sin(\theta_k^r) \\ -sin(\theta_k^r) & cos(\theta_k^r) \end{bmatrix} (\boldsymbol{p}_k^{\mathrm{r}} - \boldsymbol{p}_k)\,. \tag{4.5}$$

The cost function $J(\boldsymbol{z}_k, \boldsymbol{u}_k)$ is composed of the weighted ($W_{\mathbf{e}}$) quadratic tracking error, the weighted ($W_{\mathbf{q}}$) arm configuration distance-to-goal and the weighted ($W_{\mathbf{u}}$) quadratic inputs to penalize high control commands . The difference between the current and desired orientation is quadratically weighted to ensure that the robot is moving forward (w_θ). In addition, to relax the problem, we introduce the slack variable s and penalize its weighted norm

$$\begin{aligned} J(\boldsymbol{z}_k, \boldsymbol{u}_k) = \|\mathbf{e}_k\|_{W_{\mathbf{e}}} + \|\boldsymbol{u}_k\|_{W_{\mathbf{u}}} + \|(\theta_k^r - \theta_k)\|_{w_\theta} \\ + \|\boldsymbol{q}_k - \boldsymbol{q}_{\text{des,k}}\|_{W_q} + \|s_k\|_{w_{\text{slack}}}\,. \end{aligned} \tag{4.6}$$

4.2.5. COLLISION AVOIDANCE FOR STATIC OBSTACLES

In this paper, we tackle the problem of avoiding static obstacles in 3D space. Hence, we employ an octree representation of the static obstacles fed directly from 3D sensor data (e.g., depth camera). Given this information, we propose to model the free space as a set of convex polyhedrons around the robot's links instead of explicitly describing individual obstacles. This representation allows us to limit the number of collision constraints regardless of the number of obstacles, and depending only on the number of robot's links and the number of planes used for the convex regions. To compute the convex regions, we employ the method proposed in [75] using an ellipsoid based regional inflation. Fig. 4.2 depicts an ex-



(a) Polyhedron (b) Single Inequality

Figure 4.2: Generated polyhedron representing the free space in the presence of sensed pointcloud (orange) for last link and the corresponding sphere on the robot (blue) and visualization sphere-plane inequality constraint (right).

ample of one of these convex regions computed for one robot's link. For each $i \in [1, n_{\text{links}}]$, we compute a polyhedron with n_{planes} planes representing the free-space around the i-th link.

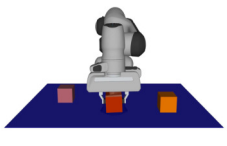
Then, we impose a linear inequality constraint between each j-th polyhedron plane and i-th link to ensure $\mathcal{B}(\boldsymbol{z}_k) \cap \mathcal{O}^{\text{static}} = \emptyset$ as

$$\begin{aligned} &\mathbf{a}^T \mathbf{p}_i \mathcal{L}_e b - (r_i + d_{\text{safety}}) \\ &\forall i \in \{1, \dots, n_{\text{links}}\}, \quad \forall j \in \{1, \dots, n_{\text{planes}}\}, \end{aligned} \tag{4.7}$$

with $\mathbf{a} = \mathbf{n}_{j,i}$ and $b = -\mathbf{n}_{j,i}^T \mathbf{p}_{j,i}^{\text{obs}}$, where $\mathbf{n}_{j,i}$ is the normal vector and $\mathbf{p}_{j,i}^{\text{obs}}$ a point on the j-th polyhedron's plane enclosing the i-th robot link, $\mathbf{p}_i$ is the i-th link position, and d_{safety} an hyper-parameter that acts as a safety margin. The proposed collision constraint ensures that each link's space is inside the convex region and thus free of static obstacles, as depicted in Fig.4.2.

4.2.6. COLLISION AVOIDANCE FOR MOVING OBSTACLES

To avoid moving obstacles, it is necessary to propagate the states of the dynamic obstacles over the planning horizon N. Using the previous constraint for dynamic collision avoidance requires the propagation of the 3D octree and the computation of the convex polyhedra for every stage, which is highly computationally expensive.



Hence, we propose to model dynamic obstacles as spheres. For each dynamic obstacle d we assume to know the position $\boldsymbol{p}_d^{\text{dyn}}$, velocity $\mathbf{v}_d^{\text{dyn}}$, and radius r_d^{dyn}, with $d = \{1, \dots, n_{\text{dyn}}\}$. Then, we employ a constant velocity model to obtain predictions on the dynamic obstacle's future positions, $\bar{\boldsymbol{p}}_{d,k}^{\text{dyn}} = \boldsymbol{p}_d^{\text{dyn}} + k\Delta t \boldsymbol{v}_d^{\text{dyn}}$. Finally, we define a non-linear collision avoidance constraint ensuring that the obstacle's space does not intersect with any link's space, $\mathcal{B}_i(\boldsymbol{z}_k) \cap \mathcal{O}^{\text{dynamic}} = \emptyset \; \forall i \in [1, n_{\text{links}}]$, imposing that the distance between both bounding spaces is larger than the sum of their radius and the previously introduced safety margin

$$\begin{aligned} \left\| \bar{\boldsymbol{p}}_{d,k}^{\text{dyn}} - \mathbf{p}_{i,k} \right\| &\geq r_d^{\text{dyn}} + r_i + d_{\text{safety}} \\ \forall i \in \{1, \dots, n_{\text{links}}\},\ d \in \{1, ..., n_{dyn}\},\ k \in 0, ..., N, \end{aligned} \tag{4.8}$$

where $\mathbf{p}_i$ is the position of the i-th robot link and r_d^{dyn} the d-th dynamic obstacle radius.

4

4.3. EXPERIMENTAL RESULTS

The presented method is evaluated in simulation and with the real hardware. After a short introduction to the experimental setup, three simulation scenarios and one real-world scenario are introduced. We compare the presented method with a sequenced MPC formulation in which arm motion and locomotion are sequenced and a conventional MPC formulation in which inequalities are formulated for individual obstacles. Scenarios are considered infeasible when following the trajectory without global replanning is not possible without violating the collision constraints.

4.3.1. EXPERIMENTAL SETUP

Hardware Setup The mobile manipulator used to validate this approach consists of the mobile base *Clearpath™ Boxer* and the robotic manipulator *Franka Emika Panda*, see Fig. 4.1. The resulting system has 10 Degrees of Freedom (DoF's) for which the dynamics are approximated using a Runge-Kutta scheme to obtain the discrete transition function $f(\boldsymbol{z}_t, \boldsymbol{u}_t)$. The presented robot is equipped with low level controllers that accept commanded velocities for all joints, one LiDAR sensor and one depth camera pointing forward. Laser data and camera depth images are fused into one pointcloud using the octomap framework [100]. Polyhedrons to describe the convex space around the links are computed using the *DecompUtil* presented in [75], into which the pointcloud generated from the vertex centers of the occupied cells of the octomap are fed. Joint positions for the arm are known at every time step using the encoders and the pose of the base is estimated using SLAM. The implementation is realized in the *Robotics Operating System* (ROS) framework, as it allows simple integration of the different components. The underlying MPC problem is solved using *FORCES-Pro* solver [101] and employing an interior-point method[102]. We used a laptop with an Intel Core i7 and 32GB of RAM to run the simulations and an Intel NUC with an Intel Core i7 and 8GB to run the real-world experiments.

Parameter Details The presented robot has nine links which are represented by four spheres for collision avoidance. The positions of the spheres on the kinematic chain are explicitly given in Table 4.1. Note that the robot is not fully contained in the union of spheres. For $d_{\text{safety}} = 0$, this could potentially result in collision. However, much larger spheres

Table 4.1: Positions of spheres for the volumetric representation.

Parent Link	Offset	Radius
base link	[0, 0, 0.25]	0.25
base link	[0.3, 0, 0.25]	0.25
link 2	[0, -0.1896, 0]	0.2275
link 7	[0, 0, 0]	0.3

would result in the inability to move close to obstacles when manipulation is requested. The parametrization with d_{safety} allows to flexibly change between different motion types, e.g. manipulation (low safety margin) and navigation (large safety margin). The centers of the given spheres are also used as seed points for the convex region generation. Two different step size were used over the time horizon, $\Delta t_1, \Delta t_2$. All remaining parameter settings are summarized in Table 4.2.

Table 4.2: Parameter Settings

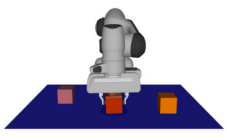
Parameter	Static Scenarios	Dynamic Scenario
Prediction Horizon	11 sec	11 sec
$\Delta t_1/\Delta t_2$	0.2/1.0 sec	0.2/1.0 sec
#Planes/Link	15	15
d_{safety}	0.15 m	0.25 m
$W_{\mathbf{e}}$	$5.0\boldsymbol{I}_2$	$5.0\boldsymbol{I}_2$
w_θ	2.0	2.0
$W_{\mathbf{q}}$	$0.7\boldsymbol{I}_7$	$0.7\boldsymbol{I}_7$
$W_{\mathbf{u}}$	$\begin{bmatrix} 0.05\boldsymbol{I}_2 & 0 \\ 0 & 5\boldsymbol{I}_7 \end{bmatrix}$	$\begin{bmatrix} 0.05\boldsymbol{I}_2 & 0 \\ 0 & 5\boldsymbol{I}_7 \end{bmatrix}$
w_{slack}	10^5	10^5

4.3.2. MANIPULATION SCENARIOS

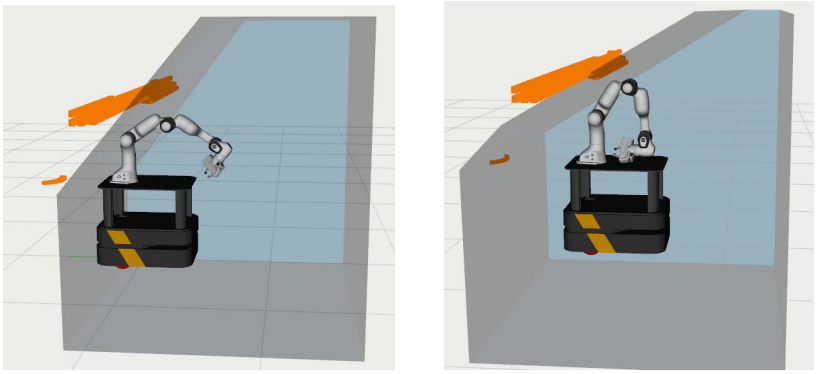
We compare our proposed method against two baseline methods: a decoupled MPC baseline (i.e., locomotion and arm motion are performed sequentially) and a coupled MPC formulation based on sphere-sphere inequality constraints, as proposed in [81]. Moreover, as this work presents a method for local trajectory optimization, the effect of global replanning during the execution is not considered. To access our method's performance, we present simulation results for:

- Static collision avoidance with a horizontal bar [81] (Sub-section 4.3.2);
- 2D trajectory tracking while avoiding collisions with randomly placed static obstacles (Sub-section 4.3.2);
- Dynamic collision avoidance with a moving obstacle (Sub-section 4.3.2).

Finally, we present experimental results on a mobile manipulator performing a real manipulation task (Sub-section 4.3.2).



Horizontal Bar A horizontal bar is placed between the start and goal configuration. In contrast to the work in [81], no object detection is required in our approach, as the sensed point cloud is fed directly into the convex region generator. The base's global path consists of a simple straight-line motion to a pose behind the bar. The motion of the manipulator and the generated convex regions for the last link of the kinematic chain are depicted in Fig. 4.4. The advantage of whole-body optimization can be extracted when the horizontal bar is placed at a lower z-positions. In Fig. 4.3, such a situation is visualized for $z = 1.3$. In the visualized case, it is not possible to move underneath the bar with the sequenced approach. On the other hand, the coupled method can navigate safely avoiding collision by moving the arm into an extended position in which the absolute height is smaller than when having it folded.

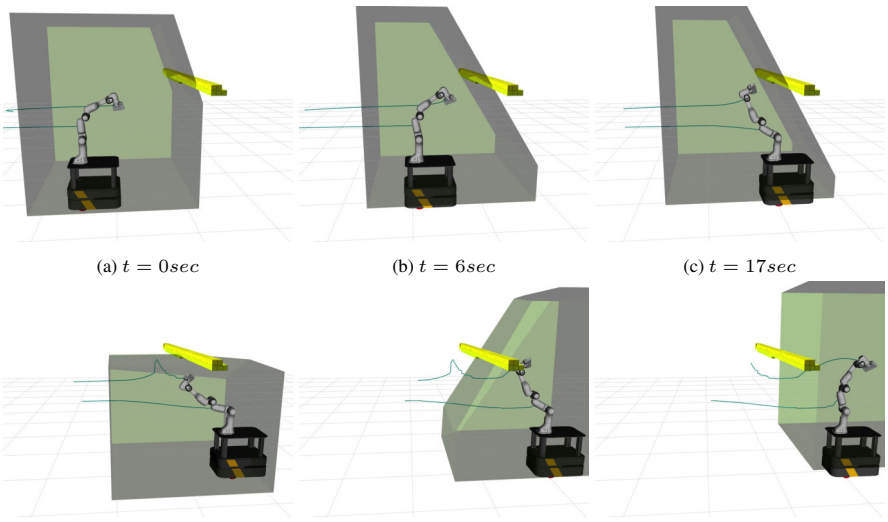


(a) coupled (b) decoupled

Figure 4.3: Advantage of coupled MPC when moving underneath a horizontal bar (orange).

Randomized Obstacles In this scenario, the robot is placed in an unstructured environment with several static obstacles. A global path for the base to reach the goal is computed, but the path is blocked with randomly generated obstacles, uniformly distributed on the intervals $x \in [2m, 5m], y \in [-2m, 2m], z \in [0m, 2m]$. Only those obstacles visible to the LiDAR sensors are available for the global planner which generates a path in the 2D plane for the base's motion. Among the randomly generated cases, only those that are feasible for an MPC trajectory optimizer are considered. Two examples for infeasible cases are depicted in Fig. 4.5.

A successful trajectory of the coupled MPC planner is depicted in Fig. 4.6. A key advantage of our method is that when the environment is densely populated with obstacles, the solving times are not affected when using convex regions to represent the free space, see Fig. 4.7. Explicitly formulating sphere-sphere inequality constraints results in an increase of solving time as the environment becomes more densely populated with obstacles. Note that convex region generation becomes only beneficial as the number of obstacles exceeds a critical value, in this case, for 50 obstacles. Furthermore, parallelizing the locomotion and arm motion allows to reduce the mean overall operational time by 48%, see Table 4.3.



(a) $t = 0sec$ (b) $t = 6sec$ (c) $t = 17sec$

(d) $t = 25sec$ (e) $t = 32sec$ (f) $t = 42sec$

Figure 4.4: Avoiding an horizontal bar. The convex region for the last link of the kinematic chain.

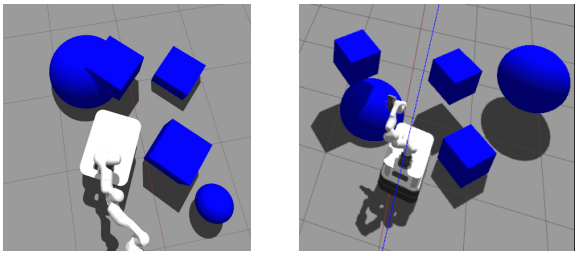
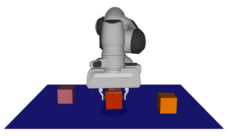


Figure 4.5: Example for infeasible cases, that were excluded from the test set in randomized scenario.

Dynamic Obstacle Here, we evaluate dynamic collision avoidance with a single moving obstacle for different obstacle's velocities. As dynamic object detection and velocity estimation are out of the scope of this work, the state of the obstacle is assumed to be known to the robot during the entire process. In Fig. 4.9, the experiment is visualized. The goal pose (light grey) is to be reached but a moving sphere, conflicting the goal, must be avoided at all time. The proposed MPC formulation's reactivity is investigated using the clearing distance $d_{\text{clear}} = \min_i \left\| \mathbf{p}^{\text{dyn}} - \mathbf{p}_i \right\| - r_i - r_d^{\text{dyn}}$ and the distance to target, $d_{\text{target}} = \|\boldsymbol{z}_{\text{des}} - \boldsymbol{z}\|$, where $\boldsymbol{z}_{\text{des}}$ is composed of the desired base and arm configuration. Different velocities ($\boldsymbol{v}_d^{\text{dyn}}$) and different heights (z_{obs}) of the moving obstacles were investigated. Fig. 4.8 shows that our approach successfully avoids collision with dynamic obstacles when moving towards the goal.

Real-World Experiment We evaluated the presented method in real-world scenarios in a simple pick & place setup. Fig. 4.1 depicts the experimental scenario, where the robot



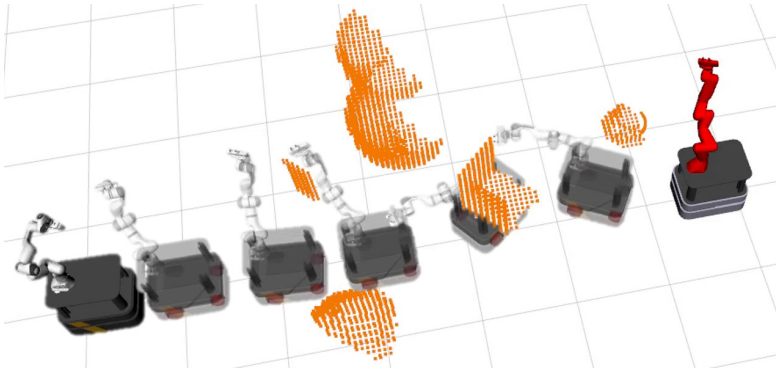
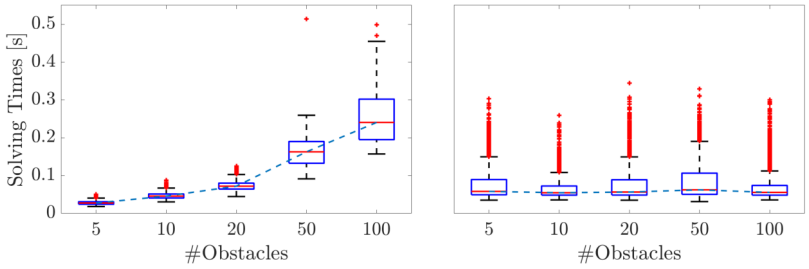


Figure 4.6: Five obstacles avoidance, final configuration in red, occupied voxels in the octomap are represented in orange.



(a) Obstacles as spheres

(b) Convex regions

Figure 4.7: Comparison solver performance for an increasing number of obstacles.

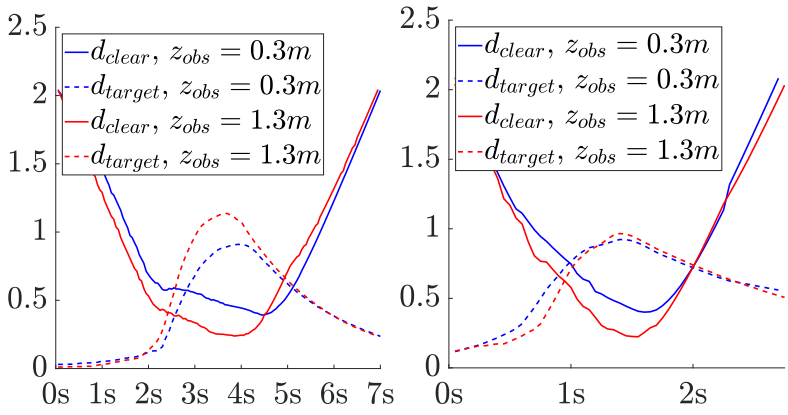
picks up an object on the left (pose in light green) and moves to the target pose (pose in light red) without colliding with the obstacle visualized in light blue. Intermediate poses of the successful trajectory are visualized in Fig. 4.10. A video of the experiment is attached to this work.

4.4. CONCLUSION

This work proposes a whole-body trajectory optimization to navigate in unstructured and simplified dynamic environment safely. Evolution of the environment, i.e., dynamic obstacles, were incorporated, and static collision avoidance is realized by a union of convex regions describing the free space. By representing the free space, rather than individual obstacles, the number of inequality constraints is limited. The scaling of the method for an increasing number of obstacles and its ability to avoid collision with moving obstacles were shown in randomized scenarios. Real-time applicability was demonstrated on a 10-DoF mobile manipulator in a Pick & Place test case. The proposed approach overcomes the limitations of previous works and allows whole-body trajectory optimization in dynamic environments.

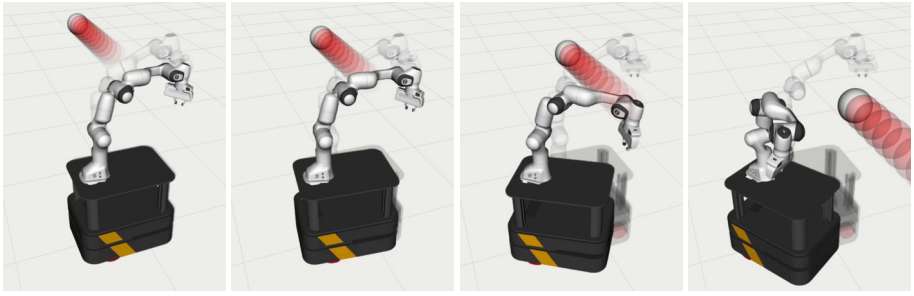
	decoupled	coupled
mean	219.65s	113.822s
std. deviation	24.21s	8.35s
min	199.27s	106.83s
max	270.25s	131.24s

Table 4.3: Compared execution times coupled and decoupled approach for cases that were feasible for both methods.



(a) $\|\boldsymbol{v}_d\| = 0.2m/s$ (b) $\|\boldsymbol{v}_d\| = 0.5m/s$

Figure 4.8: Clearance from moving obstacle and distance to target position for different obstacle velocities.



(a) $t = 0sec$ (b) $t = 1sec$ (c) $t = 2sec$ (d) $t = 3sec$

Figure 4.9: Motion avoiding moving obstacle (red) while attempting to reach target (light grey).

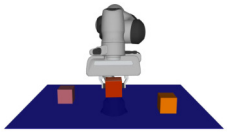


Figure 4.10: Trajectory in mock-up store avoiding obstacles, goal configuration in light grey.

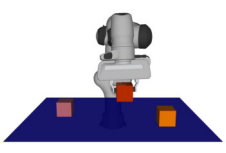
5

Dynamic Optimization Fabrics for Trajectory Generation

The implementation of Model Predictive Control (MPC) demonstrated its capability to generate whole-body trajectories for a mobile manipulator efficiently. Employing free-space decomposition on individual robot links effectively reduced computational costs. However, solving the optimization problem remained computationally expensive, limiting control frequency and yielding non-smooth trajectories. Consequently, this chapter marks a shift away from optimization-based methods towards a geometric approach known as Optimization Fabrics (fabrics). This method allows the encoding of trajectory generation in ordinary differential equations of second order, presenting an alternative to MPC that offers much faster computation cycles and more reactive motion. Although geometric fabrics have proven applicable to real-time trajectory generation, previous works are limited to simple, static environments that are free of local minima and global path planning is thus not required. This chapter introduces the concept of Dynamic Fabrics (DF) to overcome these limitations. Moreover, extensive comparisons between MPC and Static Fabrics (SF) are conducted to substantiate the effectiveness of geometric encoding in trajectory generation. Finally, real-world experiments are conducted, involving both a robot arm and a mobile manipulator, to validate the proposed methodologies.

This chapter is a verbatim copy of the peer-reviewed publication:

- ☞ **M. Spahn**, M. Wisse and J. Alonso-Mora. *"Dynamic Optimization Fabrics for Motion Generation"*. IEEE Transactions on Robotics, 2023.



5.1. INTRODUCTION

Robots increasingly populate dynamic environments. Imagine a robot operating alongside customers in a supermarket. It is requested to perform different tasks, such as cleaning the floor or picking a wide range of products. These different manipulation tasks may vary in their dimension and accuracy requirements, e.g. rotation around a suction gripper does not need to be specified while two-finger grippers require full poses. Thus, it is important for motion planning algorithms to support various goal definitions. Further, the robot is operating alongside humans, it has to constantly react to the changing environment and consequently update an initial plan. As customers move fast, the adaptations must be computed in real time. Therefore, motion planning is often divided into global motion planning [29] and local motion planning, which we will refer to as motion generation in this paper. A global planner generates a first feasible path that is used by a motion generator as global guidance. This paper proposes a novel approach to motion generation, that deals with a variety of different goal definitions.

Motion generation is often solved by formulating an optimization problem over a time horizon. The popularity of this approach is partly thanks to the guaranteed collision avoidance and thus safety [7], [103]. The optimization problem is then assembled from a scalar objective function, encoding the motion planning problem (e.g., the desired final position, path constraints, etc.), the transition function, defining the robot's dynamics, and several inequality constraints, integrating physical limits and obstacle avoidance. Despite abundant

(a) (b)

Figure 5.1: Dynamic fabrics for path (green) following with a non-holonomic mobile manipulator. Dynamic fabrics control all actuators simultaneously to follow the end-effector path while keeping a given orientation and avoiding collision with the environment.

applications of such optimization-based approaches to mobile robots, the computational costs limit applicability when dealing with high-dimensional configuration spaces [104], [105]. Data-driven approaches to speed up the optimization process usually come with reduced generalization abilities, loss of formal guarantees [7] and require prior, often costly, data acquisition. Moreover, due to the scalar objective function, the user must carefully

weigh up different parts of the objective function. As a consequence, optimization-based approaches are challenging to tune and inflexible to generic motion planning problems with variable goal objectives [19], [106].

In the field of geometric control, namely Riemannian motion policies (RMP) and optimization fabrics, all individual parts of the motion planning problem are formulated as differential equations of second order. Applying operations from differential geometry, the individual components are combined in the configuration space to define the resulting motion [16], [18]. This allows to iteratively *design* the motion of the robot while maintaining explainability over the resulting motion [16], [18], [19], [28].

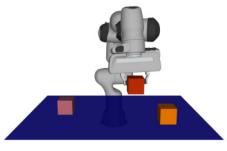
These works on optimization fabrics [18], but also on predecessors, such as RMP [15] and RMP-Flow [16], [17], have shown the power of designing reactive behavior as second-order differential equations. However, integration of dynamic features, such as moving obstacles and path following, have not been proposed nor have the framework been applied to non-holonomic systems. In this article, we exploit relative coordinate systems in the framework of optimization fabrics by introducing the dynamic pullback operation (Eq. (5.1)). This generalization can then integrate moving obstacles and path following. We show that our generalization maintains guaranteed convergence for path following tasks and improves collision avoidance with moving obstacles. Moreover, we propose a method to incorporate non-holonomic constraints. Lastly, we compare a trajectory optimization formulation, namely a model predictive control formulation, with optimization fabrics to provide the reader with a better understanding of key differences between the two approaches. We analyze computational costs and the quality of resulting trajectories for different robots. Several simulated results and real-world experiments show the practical implications of DF. The contributions of this paper can be summarized as:

1. We enable the usage of optimization fabrics for dynamic scenarios. Specifically, we propose time parameterized differential maps using up-to second-order predictor models. As a consequence, this enables the integration of moving obstacles and path following tasks , see Fig. 5.2.
2. We extend the framework of optimization fabrics to non-holonomic robots.
3. We present a quantitative comparison between model predictive control and optimization fabrics. The results reveal that fabrics are an order of magnitude faster, more reliable, and easier to tune for goal-reaching tasks with a robotic manipulator in static environments.

All findings are supported by extensive experiments in both simulation and real-world with a manipulator, a differential drive robot, and a mobile manipulator.

5.2. DERIVATION OF DYNAMIC FABRICS

We extend the framework of optimization fabrics to Dynamic Fabrics (DF). including dynamic environments and path following tasks. We prove that DF converge to moving goals and can be combined with previous approaches in geometric control. This section first introduces the notion of reference trajectory, dynamic Lagrangians and the dynamic pullback. These notations allow then to formulate DF. As DF generalize the concept of optimization



fabrics to dynamic scenarios, we refer to the non-dynamic fabrics as Static Fabrics (SF) to explicitly distinguish between the work presented in [18] and our work.

5.2.1. MOTION DESIGN USING DYNAMIC FABRICS

The method explained in this paper generalizes the concept of SF from [18] and can then be extended from the procedure outlined in Chapter 2. Note that modifications from the original procedure are highlighted in bold.

1. Design path-consistent geometries in a suited, **time-parameterized** (Definition 5.2.2) manifold of the configuration
2. Design corresponding Finsler energies defining the importance metric in this manifold.
3. Energize all geometries with the associated Finsler energies.
4. **If necessary, pull back the energized system from the time-parameterized manifold into the corresponding fixed manifold (Eq. (5.1)).**
5. Pull back the energized system into the configuration space and combine it with all components using summation.
6. Force the system with a **time-parameterized** potential. As a composition of DF, the resulting trajectory converges towards the potential's minimum (Lemma 5.2.11).

In the following, we explain our proposed changes to the framework of SF so that it remains valid in dynamic environments.

5.2.2. REFERENCE TRAJECTORIES

To enable the definition of dynamic convergence and dynamic energy we introduce a reference trajectory that remains inside a domain $\mathcal{X}$ as *boundary conforming*. This term is chosen in accordance to [18, Definition 4.6].

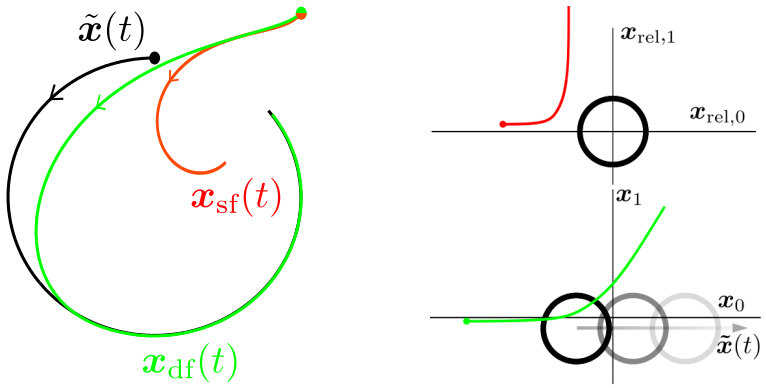
Definition 5.2.1. *A reference trajectory $\tilde{\boldsymbol{x}}(t)$, with its corresponding time derivatives $\dot{\tilde{\boldsymbol{x}}}$ and $\ddot{\tilde{\boldsymbol{x}}}$, is boundary conforming on the manifold $\mathcal{X}$ if $\tilde{\boldsymbol{x}}(t) \in \mathcal{X}, \forall t$.*

In the following, the reference trajectory will be used to define dynamic Lagrangians and dynamic fabrics. In this context, the word 'dynamic' can often be read as 'relative to the reference trajectory'. With the notion of reference trajectories we formulate a mapping to the relative coordinate system.

Definition 5.2.2. *Given a reference trajectory $\tilde{\boldsymbol{x}}$ on $\mathcal{X}$, the dynamic mapping $\phi_d : \mathcal{X} \times \mathcal{X} \to \mathcal{X}_{rel}$ represents the relative coordinate system $\boldsymbol{x}_{rel} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$.*

5.2.3. DYNAMIC PULLBACK

The theory of optimization fabrics also applies to relative coordinates $\boldsymbol{x}_{\text{rel}}$, specifically, specs and potentials can be formulated in moving coordinates. However, there is no theory to combine specs defined in relative coordinates with specs in fixed coordinates. In most cases, individual components of the behavior design are not formulated in the same relative coordinates. Specificially, the configuration space is always static, so we introduce a



(a) Dynamic convergence (b) Dynamic Avoidance

Figure 5.2: The two implications of Dynamic Fabrics. In (a), it can be seen that the trajectory obtained with DF (green) converges towards the reference trajectory (black) while the trajectory with Static Fabrics (red) does not converge. In (b), the top part visualizes collision avoidance as suggested in [18]. Here, the trajectory and obstacle are expressed in a relative system $\boldsymbol{x}_{\text{rel}}$. Using the dynamic pull, Eq. (5.1), this can be transformed into the static reference frame $\boldsymbol{x}$, bottom part. Together with dynamic energization, the framework of optimization fabrics is leveraged for dynamic environments. The motion of the obstacle, $\boldsymbol{x}_{\text{rel}}(t)$ is visualized with an arrow, future positions of the obstacle are shown in lighter color. The resulting trajectory obtained with DF is shown in green.

transformation of a relative spec into the static space $\mathcal{X}$. We call this operation *dynamic pullback*.

$$\text{pull}_{\phi_d}(\boldsymbol{M}_d, \boldsymbol{f}_d)_{\mathcal{X}_{\text{rel}}} = (\boldsymbol{M}_d, \boldsymbol{f}_d - \boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}})_{\mathcal{X}} \tag{5.1}$$

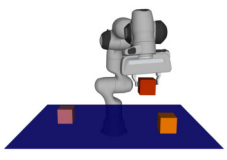
Two specs $\mathcal{S}_{\mathcal{X}_{\text{rel},1}}$ and $\mathcal{S}_{\mathcal{X}_{\text{rel},2}}$ defined in two different relative coordinate systems are then combined by first applying the dynamic pullback to both individually and then applying the summation operation for specs. The dynamic pullback is the natural extension to optimization fabrics for relative coordinate systems. It cannot directly be integrated into the framework of optimization fabrics as it breaks the algebra. In the following, we derive several generalizations so that the theory remains valid even in the presence of reference trajectories for individual components, such as moving obstacles or reference trajectories.

5.2.4. DYNAMIC LAGRANGIANS

Next, we show that energy conservation commutes with the dynamic pullback. This allows us to transfer findings on conservative fabrics to dynamic fabrics. We call a Lagrangian that is defined using relative coordinates a *dynamic Lagrangian* and write $\mathcal{L}_d(\boldsymbol{x}_{\text{rel}}, \dot{\boldsymbol{x}}_{\text{rel}})$. In this relative coordinate system, the dynamic Lagrangian has the same properties as the Lagrangian defined in [18], specifically it induces the Lagrangian spec through the Euler-Lagrange equation, $\partial^2_{\dot{\boldsymbol{x}}_{\text{rel}}\dot{\boldsymbol{x}}_{\text{rel}}}\mathcal{L}_d\ddot{\boldsymbol{x}}_{\text{rel}} + \partial^2_{\dot{\boldsymbol{x}}_{\text{rel}}\boldsymbol{x}_{\text{rel}}}\mathcal{L}_d\dot{\boldsymbol{x}}_{\text{rel}} - \partial_{\boldsymbol{x}_{\text{rel}}}\mathcal{L}_d$, as $(\boldsymbol{M}_{de}, \boldsymbol{f}_{de})$. The system's Hamiltonian $\mathcal{H}_d = \partial_{\dot{\boldsymbol{x}}_{\text{rel}}}\mathcal{L}_d^T\dot{\boldsymbol{x}}_{\text{rel}} - \mathcal{L}_d$ is conserved by the equation of motion as proven in [18].

Applying the dynamic pullback to the dynamic Lagrangian we obtain the transformed Lagrangian $\mathcal{L}_d(\boldsymbol{x}, \dot{\boldsymbol{x}}, \tilde{\boldsymbol{x}}, \dot{\boldsymbol{x}})$ in the static coordinate system.

Theorem 5.2.3. *Let $\mathcal{L}_d(\boldsymbol{x}_{rel}, \dot{\boldsymbol{x}}_{rel})$ be a dynamic Lagrangian and let ϕ_d be the dynamic*



mapping to $\boldsymbol{x}_{rel}$. Then, the application of the Euler-Lagrange equation commutes with the dynamic pullback.

Proof. We will show the equivalence by calculation. As shown above, the induced spec is defined in the relative system as $(\boldsymbol{M}_{de}, \boldsymbol{f}_{de})$. It can be dynamically pulled to form

$$\text{pull}_{\phi_d}(\boldsymbol{M}_{de}, \boldsymbol{f}_{de})_{\mathcal{X}_{\text{rel}}} = (\boldsymbol{M}_{de}, \boldsymbol{f}_{de} - \boldsymbol{M}_{de}\ddot{\tilde{\boldsymbol{x}}})_{\mathcal{X}} \tag{5.2}$$

We can dynamically pull the Lagrangian $\mathcal{L}_d(\boldsymbol{x}_{\text{rel}}, \dot{\boldsymbol{x}}_{\text{rel}})$ to form $\mathcal{L}_d(\boldsymbol{x}, \dot{\boldsymbol{x}}, \tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}})$, where only the first two variables are system variables. Using the generalized Euler-Lagrange equation, the equations of motion of the pulled Lagrangian are obtained as

$$\begin{aligned}
0 &= \frac{d}{dt}\frac{\partial \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}} - \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{x}} \\
&= \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}} \partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}} \partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}} \partial \tilde{\boldsymbol{x}}}\dot{\tilde{\boldsymbol{x}}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}} \partial \dot{\tilde{\boldsymbol{x}}}}\ddot{\tilde{\boldsymbol{x}}} - \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{x}} \\
&= \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\boldsymbol{x}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\boldsymbol{x}}}\frac{\partial \boldsymbol{x}_{\text{rel}}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \\
&+ \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\boldsymbol{x}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\tilde{\boldsymbol{x}}}}\ddot{\tilde{\boldsymbol{x}}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}\frac{\partial \dot{\boldsymbol{x}}_{\text{rel}}}{\partial \dot{\boldsymbol{x}}}\frac{\partial \boldsymbol{x}_{\text{rel}}}{\partial \tilde{\boldsymbol{x}}}\dot{\tilde{\boldsymbol{x}}} \\
&- \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{x}_{\text{rel}}}\frac{\partial \boldsymbol{x}_{\text{rel}}}{\partial \boldsymbol{x}} \\
&= \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}\ddot{\boldsymbol{x}} + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}\dot{\boldsymbol{x}} - \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}\ddot{\tilde{\boldsymbol{x}}} \\
&- \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}\dot{\tilde{\boldsymbol{x}}} - \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{x}_{\text{rel}}} \\
&= \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}(\ddot{\boldsymbol{x}} - \ddot{\tilde{\boldsymbol{x}}}) + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) - \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{x}_{\text{rel}}} \\
&= \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \dot{\boldsymbol{x}}_{\text{rel}}}(\ddot{\boldsymbol{x}} - \ddot{\tilde{\boldsymbol{x}}}) + \frac{\partial^2 \mathcal{L}_d}{\partial \dot{\boldsymbol{x}}_{\text{rel}} \partial \boldsymbol{x}_{\text{rel}}}(\dot{\boldsymbol{x}}_{\text{rel}}) - \partial_{\boldsymbol{x}_{\text{rel}}}\mathcal{L}_d \\
&= \boldsymbol{M}_{de}\ddot{\boldsymbol{x}} + \boldsymbol{f}_{de} - \boldsymbol{M}_{de}\ddot{\tilde{\boldsymbol{x}}}
\end{aligned}$$

The obtained equations of motion match the one obtained by applying the dynamic pullback, see Eq. (5.2). □

Hence, independently of the coordinates, the system conserves the energy $\mathcal{H}_d$ computed with the Hamiltonion in relative coordinates. Next, we adapt the operation of energization to dynamic Lagrangians. Dynamic Lagrangians are a necessary step to allow for collision avoidance with dynamic obstacles in the framework of optimization fabrics. Specifically, the metric for a moving obstacle is computed using the Euler-Lagrange equation in the relative coordinate system. Importantly, in this system, the same energies as with SF can be employed. Using the dynamic pullback, the energy defining the metric for the moving obstacle is then maintained according to Theorem 5.2.3. Concretely, this means that collision avoidance can be achieved in a similar manner as with SF with the added advantage of integrated motion estimates of obstacles.

Proposition 5.2.4 (Dynamic Energization)**.** *Let $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{0}$ be a differential equation and suppose $\mathcal{L}_d$ is a dynamic Lagrangian with the induced spec $(\boldsymbol{M}_{de}, \boldsymbol{f}_{de})$ and dynamic energy $\mathcal{H}_d$. Then the dynamically energized system $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \alpha_{\mathcal{H}_d}\dot{\boldsymbol{x}}_{rel} = \boldsymbol{0}$ with*

$$\alpha_{\mathcal{H}_d} = -(\dot{\boldsymbol{x}}_{rel}^T \boldsymbol{M}_{de} \dot{\boldsymbol{x}}_{rel})^{-1} \dot{\boldsymbol{x}}_{rel}^T (\boldsymbol{M}_{de}(\boldsymbol{h} + \ddot{\tilde{\boldsymbol{x}}}) - \boldsymbol{f}_{de})$$

conserves the dynamic energy $\mathcal{H}_d$.

Proof. From the derivations in [18], we can compute the rate of change of the dynamic energy as $\dot{\mathcal{H}}_d = \dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{de}\ddot{\boldsymbol{x}}_{\text{rel}} + \boldsymbol{f}_{de})$. The equations of motion can be plugged in through the definition of the reference trajectory Definition 5.2.1, $\ddot{\boldsymbol{x}}_{\text{rel}} = \ddot{\boldsymbol{x}} - \ddot{\tilde{\boldsymbol{x}}}$ to obtain:

$$\begin{aligned}
\dot{\mathcal{H}}_d &= \dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{de}(-\boldsymbol{h} - \alpha_{\mathcal{H}_d}\dot{\boldsymbol{x}}_{\text{rel}} - \ddot{\tilde{\boldsymbol{x}}}) + \boldsymbol{f}_{de}) \\
&= \dot{\boldsymbol{x}}_{\text{rel}}^T(-\boldsymbol{M}_{de}\boldsymbol{h} - \boldsymbol{M}_{de}\dot{\boldsymbol{x}}_{\text{rel}}\alpha_{\mathcal{H}_d} - \boldsymbol{M}_{de}\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{f}_{de}) \\
&= \dot{\boldsymbol{x}}_{\text{rel}}^T(-\boldsymbol{M}_{de}\boldsymbol{h} \\
&\quad + \boldsymbol{M}_{de}\dot{\boldsymbol{x}}_{\text{rel}}(\dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{M}_{de}\dot{\boldsymbol{x}}_{\text{rel}})^{-1}\dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{de}(\boldsymbol{h} + \ddot{\tilde{\boldsymbol{x}}}) - \boldsymbol{f}_{de}) \\
&\quad - \boldsymbol{M}_{de}\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{f}_{de}) \\
&= -\dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{M}_{de}\boldsymbol{h} + \dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{de}(\boldsymbol{h} + \ddot{\tilde{\boldsymbol{x}}}) - \boldsymbol{f}_{de}) \\
&\quad - \dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{M}_{de}\ddot{\tilde{\boldsymbol{x}}} + \dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{f}_{de} \\
&= 0
\end{aligned}$$

The energized system conserves the dynamic energy. □

Proposition 5.2.4 allows to combine dynamic components of the motion generator with static components. Effectively, the dynamic component *bends* the underlying geometry according to the motion of the dynamic components (e.g., a moving obstacle).

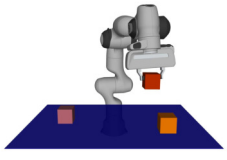
While dynamic Lagrangians and the corresponding energization operation are similar to the methods described in [18], the operation of the standard pull to the dynamically energized system must be slightly modified. Specifically, the reference velocity must be pulled. We show that dynamic energization also commutes with the standard pullback.

Theorem 5.2.5. *Let $\mathcal{L}_d$ be a dynamic Lagrangian to the reference trajectory $\tilde{\boldsymbol{x}}$, and let $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{0}$ be a second order differential equation with a metric $\boldsymbol{M}_d$ such that $\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi$ has full rank that can be written as spec $(\boldsymbol{M}_d, \boldsymbol{M}_d\boldsymbol{h})$. Suppose $x = \phi(\boldsymbol{q})$ is a differential map with $\boldsymbol{J}_\phi$ its Jacobian. Then*

$$\textit{energize}_{\textit{pull}_\phi\mathcal{L}_d}\big(\textit{pull}_\phi(\boldsymbol{I}, \boldsymbol{h})\big) = \textit{pull}_\phi\big(\textit{energize}_{\mathcal{L}_d}(\boldsymbol{I}, \boldsymbol{h})\big), \tag{5.3}$$

when the reference velocity is being pulled as $\dot{\tilde{\boldsymbol{q}}} = \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}}$. $\boldsymbol{J}_\phi^\dagger$ denotes the pseudo-inverse of $\boldsymbol{J}_\phi$. We say that the dynamic energization operation commutes with the pullback transform.

Proof. The commutation can be proven by calculation. First, we compute the right side of the equivalence. According to Proposition 5.2.4, the energized system (that maintains the



dynamic energy $\mathcal{H}_d$) writes as

$$\boldsymbol{M}_d\ddot{\boldsymbol{x}} + \boldsymbol{M}_d\boldsymbol{h} + \alpha_{\mathcal{H}_d}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) = 0,$$

with $\alpha_{\mathcal{H}_d}$ as defined in Proposition 5.2.4. Applying the pull-operation, we obtain

$$\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi\ddot{\boldsymbol{q}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{h} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\alpha_{\mathcal{H}_d}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) = 0. \tag{5.4}$$

As the equation expressed in $\mathcal{X}$, this equation in $\mathcal{Q}$ maintains the energy $\mathcal{H}_d$. Next, we compute the left hand side. The equation of motion of the pulled dynamic Lagrangian $\mathcal{L}_d$ computes as

$$\begin{aligned}\text{pull}_\phi(\boldsymbol{M}_d, \boldsymbol{f}_d) &= \boldsymbol{J}_\phi^T\left(\boldsymbol{M}_d\boldsymbol{J}_\phi\ddot{\boldsymbol{q}} + \boldsymbol{f}_d + \boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}} - \boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}\right)\\ &= \tilde{\boldsymbol{M}}_d\ddot{\boldsymbol{q}} + \tilde{\boldsymbol{f}}_d - \boldsymbol{J}_\phi^T\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}.\end{aligned}$$

The original spec is pulled accordingly

$$\begin{aligned}\text{pull}_\phi(\boldsymbol{M}_d, \boldsymbol{M}_d\boldsymbol{h}) &= \boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi\ddot{\boldsymbol{q}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{h} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}}\\ &= \tilde{\boldsymbol{M}}_d\ddot{\boldsymbol{q}} + \tilde{\boldsymbol{M}}_d\tilde{\boldsymbol{h}}\end{aligned}$$

We energize the pulled system according to Proposition 5.2.4

$$\begin{aligned}&\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi\ddot{\boldsymbol{q}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{h} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}}\\ &+\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi\alpha_{\text{pull}_\phi\mathcal{H}_d}(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\boldsymbol{x}}) = 0,\end{aligned} \tag{5.5}$$

with

$$\begin{aligned}\alpha_{\text{pull}_\phi\mathcal{H}_d} =& -\left((\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})\right)^{-1}\\ &(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\left(\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi(\tilde{\boldsymbol{h}} + \ddot{\tilde{\boldsymbol{x}}})\right.\\ &\left.- \boldsymbol{J}_\phi^T\boldsymbol{f}_d - \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}}\right)\\ =& -\left((\boldsymbol{J}_\phi\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi\boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{M}_d(\boldsymbol{J}_\phi\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi\boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})\right)^{-1}\\ &(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\left(\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{J}_\phi\tilde{\boldsymbol{h}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}\right.\\ &\left.-\boldsymbol{J}_\phi^T\boldsymbol{f}_d - \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}}\right)\\ =& -\left((\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{M}_d(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})\right)^{-1}\\ &(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\left(\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{h} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}\right.\\ &\left.-\boldsymbol{J}_\phi^T\boldsymbol{f}_d - \boldsymbol{J}_\phi^T\boldsymbol{M}_d\dot{\boldsymbol{J}}_\phi\dot{\boldsymbol{q}}\right)\\ =& -\left((\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{M}_d(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})\right)^{-1}\\ &(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger\dot{\tilde{\boldsymbol{x}}})^T\left(\boldsymbol{J}_\phi^T\boldsymbol{M}_d\boldsymbol{h} + \boldsymbol{J}_\phi^T\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}} - \boldsymbol{J}_\phi^T\boldsymbol{f}_d\right)\\ =& -\left((\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{M}_d(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})\right)^{-1}\end{aligned}$$

$$
\begin{aligned}
&\left(\boldsymbol{J}_\phi \dot{\boldsymbol{q}} - \boldsymbol{J}_\phi \boldsymbol{J}_\phi^\dagger \dot{\tilde{\boldsymbol{x}}}\right)^T \left(\boldsymbol{M}_d \boldsymbol{h} + \boldsymbol{M}_d \ddot{\tilde{\boldsymbol{x}}} - \boldsymbol{f}_d\right) \\
=& -\left((\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T \boldsymbol{M}_d (\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})\right)^{-1} (\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T \\
&\left(\boldsymbol{M}_d \boldsymbol{h} + \boldsymbol{M}_d \ddot{\tilde{\boldsymbol{x}}} - \boldsymbol{f}_d\right) \\
=& \, \alpha_{\mathcal{H}_d}
\end{aligned}
$$

Thus, we have shown equivalence between $\alpha_{\mathcal{H}_d}$ and $\alpha_{\text{pull}_\phi \mathcal{H}_d}$. As α is scalar we can can rewrite the energization term in Eq. (5.5) as

$$
\begin{aligned}
&\boldsymbol{J}_\phi^T \boldsymbol{M}_d \boldsymbol{J}_\phi \alpha_{\text{pull}_\phi \mathcal{H}_d}(\dot{\boldsymbol{q}} - \boldsymbol{J}_\phi^\dagger \dot{\boldsymbol{x}}) \\
=& \boldsymbol{J}_\phi^T \boldsymbol{M}_d \alpha_{\text{pull}_\phi \mathcal{H}_d}(\boldsymbol{J}_\phi \dot{\boldsymbol{q}} - \boldsymbol{J}_\phi \boldsymbol{J}_\phi^\dagger \dot{\boldsymbol{x}}) \\
=& \boldsymbol{J}_\phi^T \boldsymbol{M}_d \alpha_{\text{pull}_\phi \mathcal{H}_d}(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}) \\
=& \boldsymbol{J}_\phi^T \boldsymbol{M}_d \alpha_{\mathcal{H}_d}(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}})
\end{aligned}
$$

With the equivalence of the energization terms, we conclude the proof that dynamic energization commutes with the standard pullback. □

5.2.5. DYNAMIC FABRICS

With the previous results, we formulate a new class of fabrics that converge to a reference trajectory. We call this class of fabrics Dynamic Fabrics. First, some notations are introduced to eventually show that dynamically energized specs form dynamic fabrics. Analogously to unbiased specs, we define dynamically unbiased specs (i.e., specs whose solutions do not diverge from the reference $\tilde{\boldsymbol{x}}$ when starting on the reference).

Definition 5.2.6. *A spec is said to be dynamically unbiased with respect to $\tilde{\boldsymbol{x}}(t)$ if $\boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = -\boldsymbol{M}\ddot{\tilde{\boldsymbol{x}}}$, for $\boldsymbol{x}(t) = \tilde{\boldsymbol{x}}(t)$ and $\dot{\boldsymbol{x}}(t) = \dot{\tilde{\boldsymbol{x}}}(t)$.*

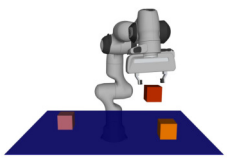
Beside being dynamically unbiased, some specs will converge to the reference trajectory independently from their initial conditions.

Definition 5.2.7. *A spec is dynamically rough with respect to $\tilde{\boldsymbol{x}}(t)$ if all its integral curves $\boldsymbol{x}(t)$ converge dynamically with respect to $\tilde{\boldsymbol{x}}(t)$.*

As for SF, DF can be formed by specs when they are being forced by a potential function $\boldsymbol{\psi}$. Such a forcing potential is generally a function of $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ and has at least one minimum. A spec that converges to a minimum of the forcing potential then forms a dynamic fabrics.

Definition 5.2.8. *A spec forms a dynamically rough fabric if it is dynamically rough with respect to $\tilde{\boldsymbol{x}}(t)$ when forced by a dynamic potential and $\exists t_1 > 0$ such that $\forall t > t_1, \boldsymbol{x}(t)$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for the optimization problem $min_{\boldsymbol{x} \in \mathcal{X}} \psi(\boldsymbol{x}, \tilde{\boldsymbol{x}}(t))$. If a spec does not form a dynamically rough fabric but all its damped variants do, it forms a dynamically frictionless fabric.*

Theorem 5.2.9 (Dynamic Fabrics)**.** *Suppose $S = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$ is a spec. Then S forms a dynamically rough fabric with respect to $\tilde{\boldsymbol{x}}$ if and only if it is dynamically unbiased with re-*



spect to $\tilde{\boldsymbol{x}}$ and it converges dynamically when being forced by a dynamic potential $\psi(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ with $\|\partial_{\boldsymbol{x}}\psi\| < \infty$ on $\mathcal{X}$.

Proof. We can write the corresponding differential equation

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{f} = -\partial_{\boldsymbol{x}}\psi \tag{5.6}$$

Assume that S is dynamically unbiased. Since the spec converges with respect to $\tilde{\boldsymbol{x}}(t)$, we have $\dot{\boldsymbol{x}} \to \dot{\tilde{\boldsymbol{x}}}, \boldsymbol{x} \to \tilde{\boldsymbol{x}}$. Because it is dynamically unbiased we also have $\boldsymbol{f} \to -\boldsymbol{M}\ddot{\tilde{\boldsymbol{x}}}$. Thus, the left hand side of Eq. (5.6), approaches $\mathbf{0}$. Consequently, the right hand side must also approach $\mathbf{0}$ and hence $\partial_{\boldsymbol{x}}\psi \to \mathbf{0}$. The last satisfies the Karush–Kuhn–Tucker (KKT) conditions of ψ.

To prove the converse, assume $\boldsymbol{f}$ dynamically biased. That implies that

$$\exists t > 0, \boldsymbol{f} = \boldsymbol{M}\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{a}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}), \boldsymbol{a}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}) \neq \mathbf{0}.$$

Hence, there exist a $t > 0$ for which the left hand side does not vanish. As ψ satisifies the KKT conditions at $\boldsymbol{x} = \tilde{\boldsymbol{x}}$, its derivative equals zero at $\boldsymbol{x} = \tilde{\boldsymbol{x}}$ which contradicts Eq. (5.6) with $\boldsymbol{M}\boldsymbol{a}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}) = \mathbf{0}$. □

Hence, the spec is required to be unbiased and convergent when forced. While the former can be verified using straight-forward computation, convergence is difficult to verify in the general case.

Lemma 5.2.10 (Dynamically energized fabrics)**.** *Suppose S is a dynamically unbiased energized spec. Then S forms a dynamically frictionless fabric if $\partial_{\boldsymbol{x}}\psi = -\partial_{\tilde{\boldsymbol{x}}}\psi$.*

Proof. The equation of motion for the energized, forced and damped system writes as

$$\ddot{\boldsymbol{x}} + \boldsymbol{h} + \alpha_{\mathcal{H}_d}\dot{\boldsymbol{x}}_{\text{rel}} + \boldsymbol{B}\dot{\boldsymbol{x}}_{\text{rel}} + \partial_{\boldsymbol{x}}\psi = 0 \tag{5.7}$$

The systems energy (dynamic Hamiltonian) is used as a Lyapunov function to show convergence. The rate of change is computed as

$$\begin{aligned}
\dot{\mathcal{H}}_d^{\psi} &= \dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{de}(-\boldsymbol{h} - \alpha_{\mathcal{H}_d}\dot{\boldsymbol{x}}_{\text{rel}} - \boldsymbol{B}\dot{\boldsymbol{x}}_{\text{rel}} - \partial_{\boldsymbol{x}}\psi - \ddot{\tilde{\boldsymbol{x}}}) \\
&\quad + \boldsymbol{f}_{de}) + \dot{\psi} \\
&= -\dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{B}\dot{\boldsymbol{x}}_{\text{rel}} - \dot{\boldsymbol{x}}_{\text{rel}}^T\partial_{\boldsymbol{x}}\psi + \dot{\boldsymbol{x}}^T\partial_{\boldsymbol{x}}\psi + \dot{\tilde{\boldsymbol{x}}}^T\partial_{\tilde{\boldsymbol{x}}}\psi \\
&= -\dot{\boldsymbol{x}}_{\text{rel}}^T\boldsymbol{B}\dot{\boldsymbol{x}}_{\text{rel}}
\end{aligned}$$

As the system energy is lower bounded with $\mathcal{H}_d + \psi \geq 0$ and $\dot{\mathcal{H}}_d^{\psi} \leq 0$, when $\boldsymbol{B}$ stricly positive definite, we must have $\dot{\mathcal{H}}_d^{\psi} \to 0$. Thus, $\dot{\boldsymbol{x}}_{\text{rel}}$ goes to zero. We can conclude that the system is dynamically converging. As it it also said to be dynamically unbiased, the damped energized system forms a dynamic fabric by Theorem 5.2.9. □

Lemma 5.2.11 (Dynamic Lagrangian fabrics)**.** *An unbiased, dynamic Lagrangian spec forms a dynamically frictionless fabric if $\partial_{\boldsymbol{x}}\psi = -\partial_{\tilde{\boldsymbol{x}}}\psi$ holds for the forcing term.*

Proof. The equations of motion induced by the dynamic Lagrangian including damping and forcing are defined by the spec and can be written explicitly as

$$\begin{aligned} \boldsymbol{M}_{\mathcal{L}_d}\ddot{\boldsymbol{x}}_{\text{rel}} + \boldsymbol{f}_{\mathcal{L}_d} + \boldsymbol{B}\dot{\boldsymbol{x}}_{\text{rel}} + \partial_{\boldsymbol{x}}\psi &= 0 \\ \boldsymbol{M}_{\mathcal{L}_d}(\ddot{\boldsymbol{x}} - \ddot{\tilde{\boldsymbol{x}}}) + \boldsymbol{f}_{\mathcal{L}_d} + \boldsymbol{B}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) + \partial_{\boldsymbol{x}}\psi &= 0 \\ \boldsymbol{M}_{\mathcal{L}_d}\ddot{\boldsymbol{x}} - \boldsymbol{M}_{\mathcal{L}_d}\ddot{\tilde{\boldsymbol{x}}} + \boldsymbol{f}_{\mathcal{L}_d} + \boldsymbol{B}\dot{\boldsymbol{x}} - \boldsymbol{B}\dot{\tilde{\boldsymbol{x}}} + \partial_{\boldsymbol{x}}\psi &= 0 \end{aligned} \tag{5.8}$$

In the following we use the Hamiltonian and the potential function as Lyapunov function to show convergence of the damped spec.

$$\begin{aligned} \mathcal{H}_d^{\psi}(\boldsymbol{x}) &= \mathcal{H}_d + \psi \\ &= \partial_{\dot{\boldsymbol{x}}_{\text{rel}}}\mathcal{L}_d^T\dot{\boldsymbol{x}}_{\text{rel}} - \mathcal{L}_d + \psi \end{aligned}$$

The time derivative is composed of the time derivative of the Hamiltonian, $\dot{\mathcal{H}}_e = \dot{\boldsymbol{x}}_{\text{rel}}^T(\boldsymbol{M}_{\mathcal{L}_d}\ddot{\boldsymbol{x}}_{\text{rel}} + \boldsymbol{f}_{\mathcal{L}_d})$, and the time derivative of the forcing potential, $\dot{\psi} = \dot{\boldsymbol{x}}^T\partial_{\boldsymbol{x}}\psi + \dot{\tilde{\boldsymbol{x}}}^T\partial_{\tilde{\boldsymbol{x}}}\psi$. Thus, the system's total energy varies over time:

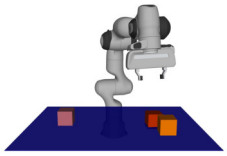
$$\dot{\mathcal{H}}_d{}^{\psi}(\boldsymbol{x}) = (\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T(\boldsymbol{M}_{\mathcal{L}_d}(\ddot{\boldsymbol{x}} - \ddot{\tilde{\boldsymbol{x}}}) + \boldsymbol{f}_{\mathcal{L}_d}) + \dot{\boldsymbol{x}}^T\partial_{\boldsymbol{x}}\psi + \dot{\tilde{\boldsymbol{x}}}^T\partial_{\tilde{\boldsymbol{x}}}\psi$$

Plugging in the equations of motion Eq. (5.8) gives

$$\begin{aligned} \dot{\mathcal{H}}_d{}^{\psi}(\boldsymbol{x}) &= (\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T(-\boldsymbol{f}_{\mathcal{L}_d} - \boldsymbol{B}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) - \partial_{\boldsymbol{x}}\psi + \boldsymbol{f}_{\mathcal{L}_d}) \\ &\quad + \dot{\boldsymbol{x}}^T\partial_{\boldsymbol{x}}\psi + \dot{\tilde{\boldsymbol{x}}}^T\partial_{\tilde{\boldsymbol{x}}}\psi \\ &= -(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{B}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) - (\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\partial_{\boldsymbol{x}}\psi \\ &\quad + \dot{\boldsymbol{x}}^T\partial_{\boldsymbol{x}}\psi + \dot{\tilde{\boldsymbol{x}}}^T\partial_{\tilde{\boldsymbol{x}}}\psi \\ &= -(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}})^T\boldsymbol{B}(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) + \dot{\tilde{\boldsymbol{x}}}^T(\partial_{\boldsymbol{x}}\psi + \partial_{\tilde{\boldsymbol{x}}}\psi). \end{aligned}$$

For $\partial_{\boldsymbol{x}}\psi = -\partial_{\tilde{\boldsymbol{x}}}\psi$ and $\boldsymbol{B}$ strictly positive definite, $\dot{\mathcal{H}}_d$ is strictly negative for $(\dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}}) \neq 0$. Since $\mathcal{H}_d^{\psi}$ is lower bounded as composition of lower bounded function and $\dot{\mathcal{H}}_d{}^{\psi} \leq 0$, $\dot{\mathcal{H}}_d{}^{\psi} \to 0$ and thus, $\dot{\boldsymbol{x}} \to \dot{\tilde{\boldsymbol{x}}}$ and $\boldsymbol{x} \to \tilde{\boldsymbol{x}}$. Hence, the spec converges dynamically with respect to $\tilde{\boldsymbol{x}}$. As the spec is further said to be dynamically unbiased, the damped spec forms a dynamic fabric by Theorem 5.2.9. □

Concretely, Lemma 5.2.11 allows for trajectory following with guaranteed convergence with DF. For example, a reference trajectory for the robot's end-effector is defined as $\tilde{\boldsymbol{x}}(t)$. Then, the potential can be designed as $\psi = \tilde{\boldsymbol{x}}(t) - \boldsymbol{x}$ (respecting the construction rule required for Lemma 5.2.11). In contrast to SF, where the static potential function is simply updated at every time step, DF makes use of the dynamics of the reference trajectory through the dynamic pullback.



5.2.6. CONSTRUCTION PROCEDURE

From the high-level procedure explained in Section 5.2.1, we can derive the algorithm using the formal findings in this section, see Algorithm 1.

Algorithm 1: Motion design with dynamic fabrics

```
Define basic inertia as spec Mq̈ + Mh = 0
for avoidance in avoidances do
    Define differential map between 𝒬 and 𝒳_i: ϕ or ϕ_t
    Design geometry on 𝒳_i: ẍ_i + h_{2,i} = 0
    Design Finsler energy for behavior on 𝒳_i: ℒ_i
    Energize geometry with Finsler energy 5.2.4
    if ϕ is time-parameterized then
        Apply dynamic pullback to energized system
    end
    Apply standard pullback
    Add pulled avoidance component to root fabric
end
Force root system with (time-parameterized) potential
```

Methods to design the individual components, such as geometry and Finsler structures, are introduced in [18]. As these design patterns do not vary for DF, they are not repeated here. In the result section, we show some experimental examples highlighting the comparative advantage of optimization fabrics over model predictive schemes and the advantage of DF over SF for dynamic environments.

5.3. EXTENSION TO NON-HOLONOMIC CONSTRAINTS

Mobile manipulators are often equipped with a non-holonomic base (e.g., a differential drive mobile robot). In contrast to revolute joints for manipulators, non-holonomic bases imply non-holonomic constraints. Based on ideas presented in [60], we propose a method to integrate such constraints in optimization fabrics, including DF.

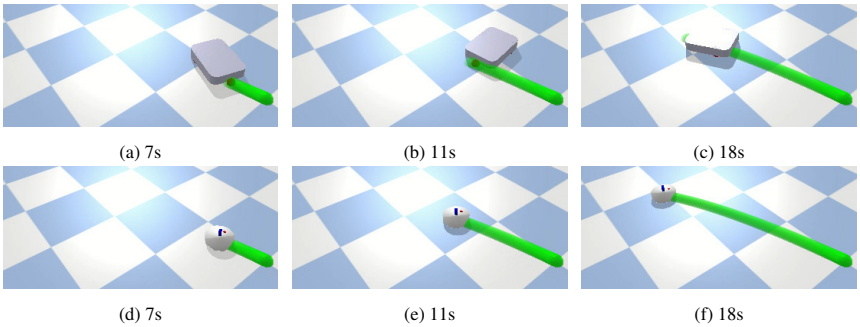
We assume that the non-holonomic constraint at hand can be expressed as an equality of form

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_{\text{nh}} \dot{\boldsymbol{q}}, \tag{5.9}$$

where $\boldsymbol{J}_{\text{nh}}$ is the Jacobian of the constraint, $\dot{\boldsymbol{q}}$ is the velocity of the controlled joints of the system and $\dot{\boldsymbol{x}}$ is the root velocity of the fabric. For a differential drive $\dot{\boldsymbol{x}}$ is the velocity of the system in the Cartesian plane $(\dot{x}, \dot{y}, \dot{\theta})$ and $\dot{\boldsymbol{q}}$ is the velocity of the actuated wheels $(u_{\text{left}}, u_{\text{right}})$. Moreover, we assume that Eq. 5.9 is smooth and differentiable so that we can write

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{\text{nh}} \dot{\boldsymbol{q}} + \boldsymbol{J}_{\text{nh}} \ddot{\boldsymbol{q}}. \tag{5.10}$$

The theory of optimization fabrics allows to pull a tree of fabrics back into one fabric



(a) 7s (b) 11s (c) 18s

(d) 7s (e) 11s (f) 18s

Figure 5.3: Path (green) following with a holonomic and a non-holonomic robot using DF with the extension to non-holonomic robots

expressed in its root-coordinates of form $\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{f} = 0$ with its solution as

$$\ddot{\boldsymbol{x}} = -\boldsymbol{M}^{-1}\boldsymbol{f}. \tag{5.11}$$

Plugging Eq. 5.10 into the root fabric we obtain the non-holonomic fabric of form

$$\begin{aligned} \boldsymbol{M}\boldsymbol{J}_{\text{nh}}\ddot{\boldsymbol{q}} + \boldsymbol{M}\dot{\boldsymbol{J}}_{\text{nh}}\dot{\boldsymbol{q}} + \boldsymbol{f} &= 0 \\ \boldsymbol{M}_{\text{nh}}\ddot{\boldsymbol{q}} + \boldsymbol{f}_{\text{nh}} &= 0 \\ . \end{aligned}$$

Note that $\boldsymbol{M}_{\text{nh}}$ is not necessarily a square matrix and thus not invertible as it was in the original fabric. To find the best actuation for the wheels, we formulate motion generation with fabrics as an unconstrained optimization problem

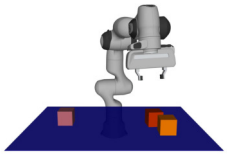
$$\ddot{\boldsymbol{q}}^* = \min_{\ddot{\boldsymbol{q}}} \|\boldsymbol{M}_{\text{nh}}\ddot{\boldsymbol{q}} + \boldsymbol{f}_{\text{nh}}\|_2^2 . \tag{5.12}$$

In this approach, we minimize the error of the final equation. We could equally derive Eq. (5.12) with the objective of minimizing the error between $\ddot{\boldsymbol{x}} = \boldsymbol{J}_{\text{nh}}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\text{nh}}\dot{\boldsymbol{q}}$ and the original fabric's solution $\ddot{\boldsymbol{x}} = -\boldsymbol{M}\boldsymbol{f}$. The mimization of the difference leads to similar result. This optimization problem replaces Eq. (5.11) and is solved by

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{M}_{\text{nh}}^{\dagger}\boldsymbol{f}_{\text{nh}}. \tag{5.13}$$

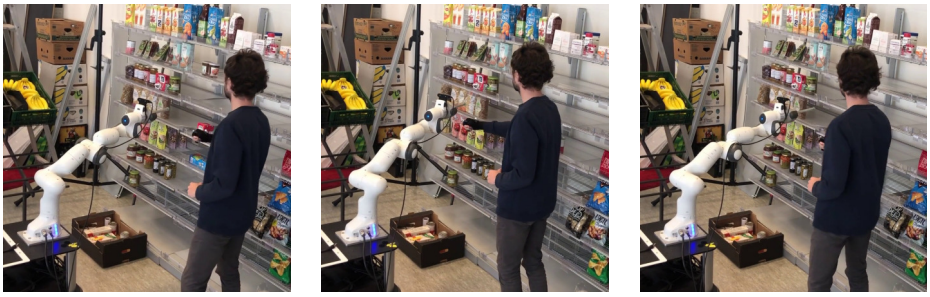
The solutions to this problem makes optimization fabrics, and thus dynamic fabrics, applicable to non-holonomic robots, such as differential drive robots or cars. A qualitative comparison between a trajectory generated for a holonomic and a non-holonomic robot is shown in Fig. 5.3.

The theory of optimization fabrics is built upon energy conservation of artificial energies that design the motion. Eq. (5.12) does not solve the resulting spec exactly, but minimizes the deviation according to the least square objective function. For many kinematic systems, e.g., differential drive model, bicycle model, the non-holonomic constraint additionally reduces the number of degrees of freedom, $\dim q < \dim x$. As a consequence, the least



square solution has a non-zero residuum. Then, some fundamental properties of optimization fabrics, such as energy conservation and convergence can no longer be guaranteed. Despite this theoretical shortcoming, we show that this approach leads to good performance in practical applications.

5.4. EXPERIMENTAL RESULTS



(a) $t = 0$s (b) $t = 2$s (c) $t = 3$s

Figure 5.4: Dynamic Fabrics in the presence of a human. The human hand's state is estimated with a motion capture system. The robot smoothly and in advance avoids the human operator and allows for safe coexistence.

In this section, the performance of optimization fabrics is assessed on various robotic platforms. Although [18] suggested performance benefits over optimization-based methods to local motion planning, no quantitative comparisons have been presented to this date. The scenarios that we have chosen here (especially in the first two experiments) are intentionally simple to identify the specific differences. In the real world experiments, we show the differences on more dynamic scenarios, where the limited frequency of a global planning method, such as RRT, justifies the need for a local planning method. To give a general idea of the performance differences between SF and receding-horizon trajectory optimization, we compare the performance of an MPC formulation, adapted from [69], with SF, as proposed in [18]. The second experiment compares performance between SF and DF for trajectory following tasks. In the third experiment, moving obstacles are added to the scene to form a dynamic environment. Our extension to non-holonomic systems is tested in the fourth experiment. Then, everything is combined in an experiment with a differential drive mobile manipulator. Finally, we present a possible application of a robot sharing the environment with a human. The experiments described here are supported by videos accompanying this paper.

5.4.1. SETTINGS & PERFORMANCE METRICS

We present a detailed analysis of the experimental results for two commonly used setups, namely the *Franka Emika Panda*, a *Clearpath Boxer*, and a mobile manipulator composed of both components see [69]. Note, that these robots are representative of commonly used robots in dynamic environments. The Franka Emika Panda is a 7 degree-of-freedom robot with joint torque sensors, comparable to the Kuka Iiwa. Mobile manipulators equipped with differential drives are widely used by other manufacturers, see Pal Robotics Tiago or the Fetch Robotics Mobile Manipulator.

Compared to [16], we propose a more extensive list of metrics. With regards to safety, we measure the **Clearance**, the minimum distance between the robot and any obstacle along the path. For static goals, solver planner performance is measured in terms of **Path Length**, euclidean length of the end-effector trajectory, and **Time-to-Goal**, time to reach the goal. For trajectory following tasks, this measure is replaced by **Summed Error**, the normed sum of deviation from the desired trajectory. Computational costs are measured by the average **Solver Time** in each time step. Most important, binary success is measured by the **Success Rate**, where failure indicates that either the goal was not reached or a collision occurred during execution. Performance metrics are only evaluated if the concerned motion generator succeeded. More information on the testbed can be found in [107].

In static, industrial environments the time to reach the goal can be considered the one single most important metric, but we argue that dynamic environments require a more nuanced performance evaluation and thus a set of metrics. Intentionally, we do not give general weights to the individual metrics, as their corresponding importance highly depends on the application. As a consequence, we tuned the compared planners in such a way that they reach the goal in a similar time. Note that the general speed for all planners compared in this article can be adjusted by choosing a different parameter setup.

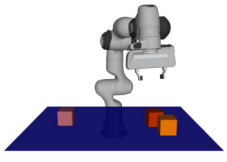
As this work does not focus on obstacle detection, we simplify obstacles to spheres. Thus, we assume that an operational perception pipeline detects obstacles and constructs englobing spheres. The experiments are randomized in either the location of the obstacles, the location of the goal, the initial configuration, or in a combination of all three aspects. For every experiment, the type and level of randomization are stated.

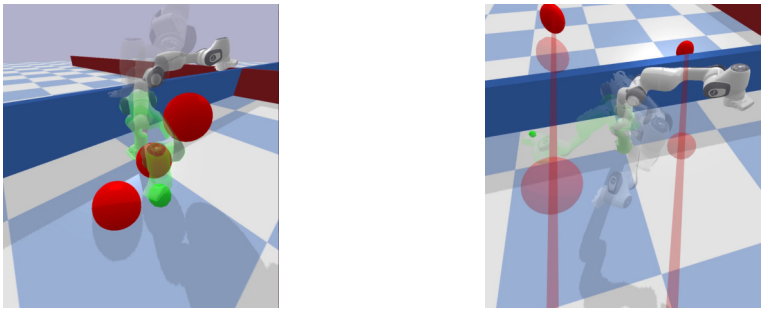
5.4.2. EXPERIMENT 1: STATIC FABRICS VS. MPC

In the first experiment, we compare the performance of an MPC formulation with SF [18], [28]. Compared to the formulation used in [69], we use a workspace goal rather than a configuration space goal, and apply a second order integration scheme so that the control outputs are accelerations instead of velocities. We clarify that the formulation deployed for the following tests is geometric as the model used is a second order integrator and does not include the dynamics of the robots. The main reason lies in the reduced computational costs and the inaccessibility of a highly accurate model [108].

Parameters The low-level controller of the robot runs at 1 kHz. The fabrics are running at 100 Hz and the MPC at 10 Hz. The time horizon for the MPC planner was set to $T = 3$s spread equally over $H = 30$ stages. Based on the findings in [69], we are confident that the MPC planner is close to its optimal settings. Moreover, we used the implementations by [101], [102], which are reported to have improved performance over open-source libraries like *acado*.

Simulation A series of $N = 50$ runs was evaluated with the Panda robot in simulation. Randomized end-effector positions were set for every run, while the initial configuration remained unchanged. One to five spherical obstacles of radius $r = 0.15$ m were placed in the workspace at random. An example setup is shown in Fig. 5.5a. The results are summarized in Fig. 5.6. Solver times with fabrics averaged at 1 ms while the MPC solver





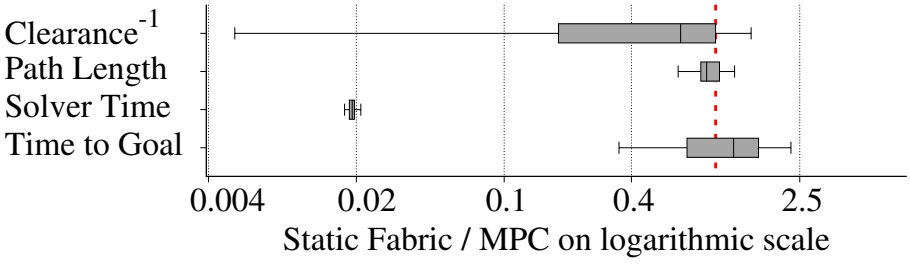
(a) (b)

Figure 5.5: Examples for simulation setups with Panda robot. Initial configuration are shown in white and final configurations in light green. Obstacles are visualized in red. In (a), only static obstacles are considered. In (b), the trajectories of two moving obstacles are visualized in light red.

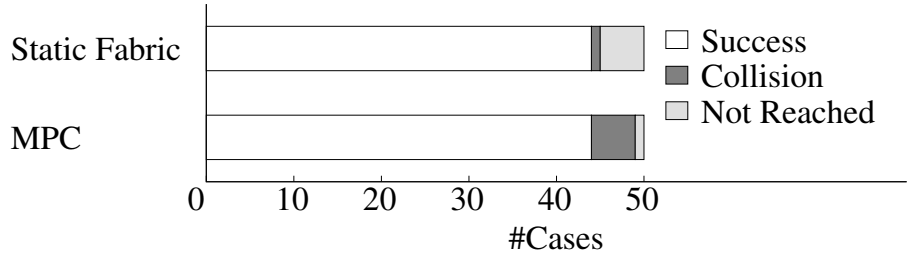
took around 50 ms in every time step. Although the path length is similar with both solvers, the minimum clearance from obstacles is increased with SF (0.183 m) compared to MPC (0.138 m). This means that the trajectories are safer and thus more suitable for dynamic environments. Both motion generation methods fail in 6 cases. However, the SF produce only one collision while MPC creates 5 collisions. The remaining failures are deadlocks. For both methods, deadlocks result from local minima, highlighting the need for supportive global plans. Collisions are caused by numerical inaccuracies, which are generally higher with MPC due to the lower frequency.

Real World For the experiments with the real robot, we limited the number of test runs to $N = 20$. In contrast to the simulated results, MPC has significantly more collisions than SF, Fig. 5.7b. This is likely to be caused by the lower frequency at which the MPC is running. While in simulation the model matches the actual behaviour perfectly and the time interval between two computations can be accuratly predicted, more uncertainty in the model is present in the real world. This leads to prediction errors that cause collisions. For the collision free cases, the real world experiments confirm that optimization fabrics tend to be more conservative with respect to obstacles, see *Clearance* in Fig. 5.7a. Similar to the simulated results, the solving time is reduced by a factor of around 50 with fabrics. This allows to run the planner at a higher frequency and thus generating smoother motions.

Discussion The difference in performance (except for solver time) is likely caused by the different objective metrics. The objective function in the MPC formulation is mainly governed by the Euclidean distance to the goal while control inputs and velocity magnitude are given a relative small weight. Avoidance behaviors, such as joint limit avoidance and obstacle avoidance, are respected through inequality constraints. In contrast, SF design the objective in a purely geometric manner including all avoidance behaviors. Thus the manifold for the motion is directly altered by the avoidance behaviors, i.e., the manifold is *bent* [18] so that the notion of shortest path changes with the addition of obstacles. This shaping of the manifold leads to improved canvergence compared to the combination of



(a) Metrics evaluation for successful experiments



(b) Success results

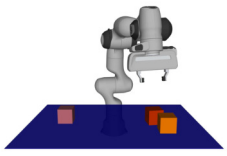
Figure 5.6: Results for randomized motion planning problems with the Panda robot in simulation. Lower values represent an improved performance of SF over MPC.

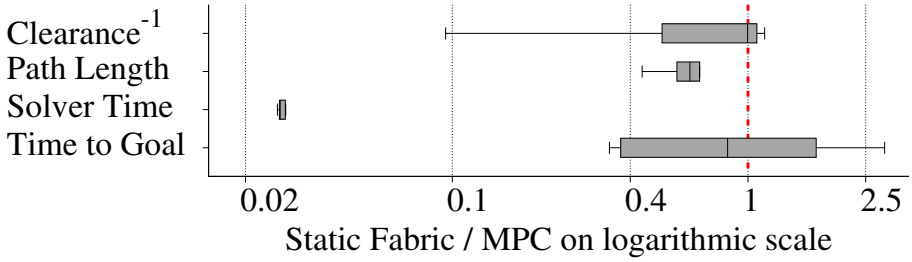
Euclidean distance objective function and inequality constraints used with MPC.

5.4.3. EXPERIMENT 2: STATIC FABRICS VS. DYNAMIC FABRICS

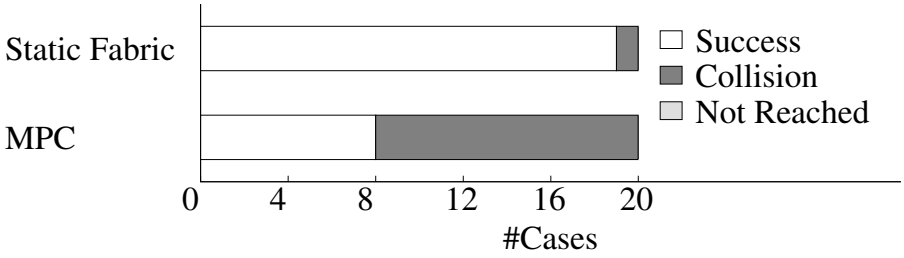
In motion planning for dynamic environments, global and local planning methods work together to achieve efficient and safe motion of the robot. However, SF are not designed to follow global paths. Path following can only be achieved using a pseudo-dynamic approach where the forcing potential is shifted in every time step without considering the dynamics of the trajectory. Therefore, we propose DF to allow smoother path following tasks, where the speed of the goal is also considered during execution. In this second experiment, we investigate how DF compare to SF for path following tasks. Specifically, we show that DF outperform SF when following a path generated by a global planner.

Simulation We evaluated DF on the Panda robot robot in simulation with an analytic, time-parameterized curve and a path generated by a global planner, namely RRT (Fig. 5.8). In the case of the analytic trajectory, the three obstacles were randomized across all runs. For the experiment with the global planner, the goal position and the obstacles were randomized across all runs. A total of $N = 50$ experiments were executed for this experiment. The reduced summed error for dynamic fabrics verifies the theoretical finding that dynamic fabrics can follow paths more closely. The average error over all runs with the analytic trajectory is 0.0792m (DF) and 0.136m (SF), see Fig. 5.10a for the comparison. For the spline path generated with RRT, the average error over all runs is 0.145m (DF) and 0.240m





(a) Metrics evaluation for successful experiments



(b) Success results

Figure 5.7: Results for randomized motion planning problems with the real Panda robot. SF are more conservative around obstacles, improving on safety, while reducing the computational cost by a factor of ≈ 50. As a result of the increased clearance, collisions are more reliably avoided with SF.

(SF), see Fig. 5.10b for the comparison.

Real-World Path following was also assessed with the real Panda robot in similar settings. Quantitative results are only presented for $N = 20$ different paths with splines where up to three obstacles were added to the workspace, see Fig. 5.9. The results in real-world confirm the findings from the simulation. By exploiting the velocity information of the trajectory, the integration error can be effectively reduced, Fig. 5.11. In contrast to the simulation we see a higher fluctuation in solver times, which can be caused by a generally lower capacity of the computing unit on the robot.

5.4.4. EXPERIMENT 3: MOVING OBSTACLES

Next, we compare the different methods in the presence of dynamic obstacles. All experiments in this section consist of at least one moving obstacle that follows either an analytic trajectory or a spline. Here, we use stationary goals to isolate the results from the behavior investigated in the previous section.

Simulation For this series with the simulated Panda robot, only the goal position was randomized. The initial configuration

$$\boldsymbol{q}_0 = [1.0, 0.0, 0.0, -1.5, 0.0, 1.8675]^T,$$

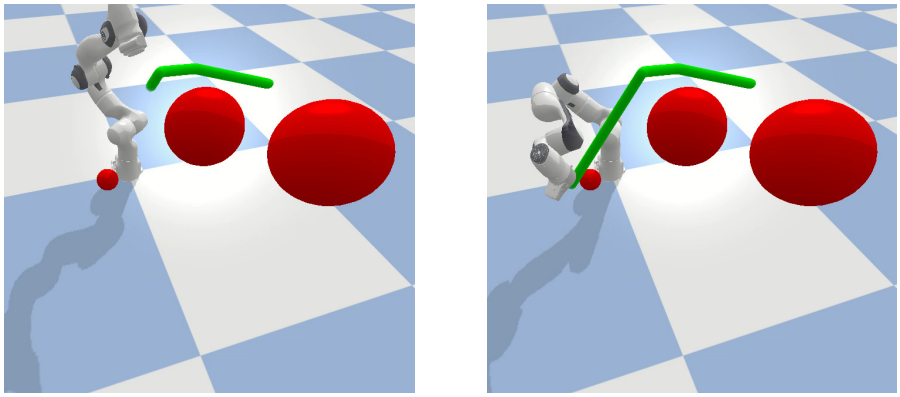
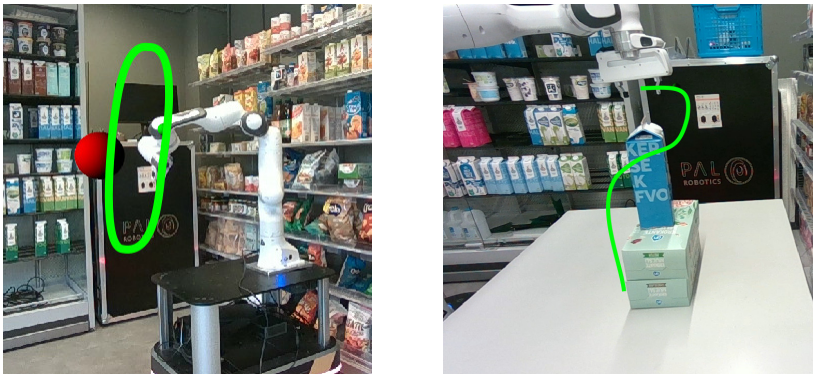


Figure 5.8: Path generated with RRT from OMPL.



(a) (b)

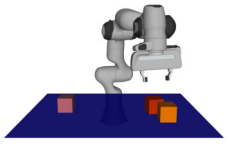
Figure 5.9: Trajectory following tasks with Dynamic Fabrics. In (a), the trajectory is a time-parameterized analytic curve. In (b), the trajectory is described by a spline.

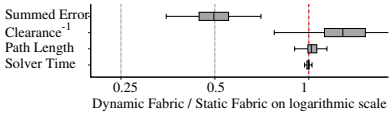
and the two moving obstacles with the trajectories

$$\tilde{\boldsymbol{x}}_{\text{obst1}} = [-1.0 + 0.1t, -0.4, 0.7]^T,$$
$$\tilde{\boldsymbol{x}}_{\text{obst2}} = [-1.0 + 0.2t, 1.0 - 0.1t, 0.3]^T$$

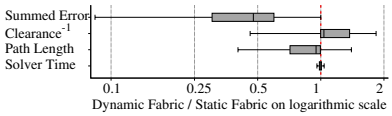
were kept constant throughout all experiments. The environment is visualized in Fig. 5.5b. The comparison between SF and DF shows that DF are more conservative in terms of collision avoidance with dynamic obstacles. Specifically, the distance between the robot and the obstacles is increased (Fig. 5.12a). The success rate with DF compared to SF is significantly improved, see Fig. 5.12b. Thus showing the need for using DF in dynamic environments.

Real-World In a series of $N = 20$ experiments, performance on the real panda arm was assessed. The same trend for more conservative behavior with DF compared to SF can be





(a) Analytic, user-specified global path



(b) Global path generated by RRT using OMPL

Figure 5.10: Comparison between static and dynamic fabrics for trajectory following tasks in simulation. Lower values in a metric indicate that DF performed better than SF.

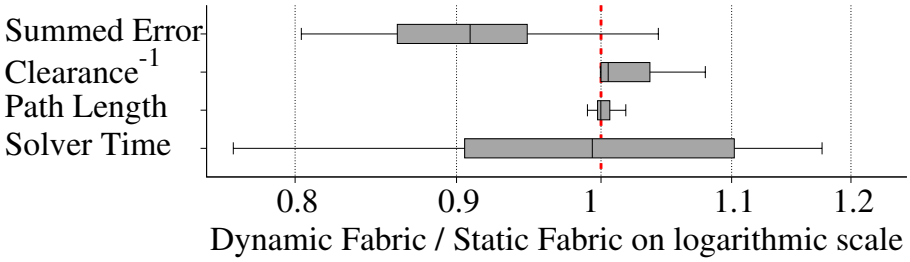


Figure 5.11: Comparison between SF and DF when following a path defined by a basic spline in the real world. The splines and the obstacles are different for the $N = 20$ case. DF achieve lower deviation errors that SF.

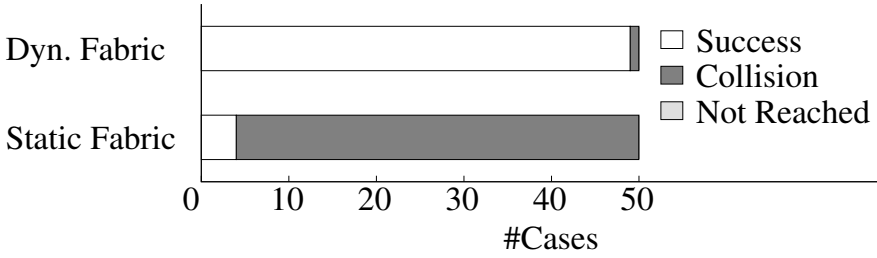
observed, Fig. 5.13. However, DF take longer on average to reach the goal as they keep larger clearance from obstacles. Note that collisions are effectively eliminated with DF compared to SF.

By investigating one example out of the series, see trajectories in Fig. 5.14, the reason for the large number of collisions with SF can be explained. Both methods initially drive the end-effector to the goal position. As the moving obstacle is approaching the robot, the DF are starting to react while SF are not changing its behavior resulting in a very sudden motion at around $t = 30$s. SF treat moving obstacles as pseudo-static (i.e., the position of the obstacle is updated at every time step, but the information on its velocity is discarded). As a result, the relative velocity between obstacle and robot is only a function of the velocity of the robot. Geometries and energies for collision avoidance with fabrics are, by design, a function of this velocity and therefore fail to avoid moving obstacles when the robot moves slowly or not at all. This behavior is most visible when the goal has already been reached but an obstacle is approaching. DF on the other hand take the velocity of the moving obstacles into account and can therefore avoid them.

5.4.5. EXPERIMENT 4: NONHOLONOMIC ROBOTS

Simulation This experiment assesses the performance of the proposed method to compute trajectories for non-holonomic robots with fabrics. Specifically, we run experiments

(a) Metrics evaluation for sucessful experiments



(b) Success results

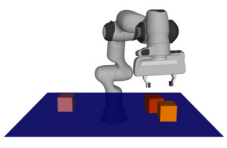
Figure 5.12: Comparison between SF and DF for scenarios with dynamic obstacles. While path length and solver time is not increased, clearance is increased and the time to reach the goal is reduced with DF compared to SF.

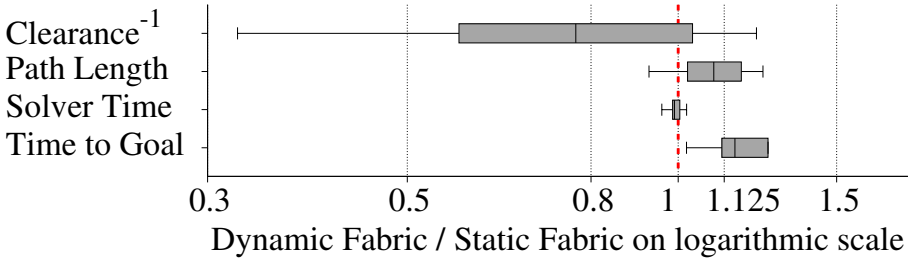
for a *Clearpath Boxer* for position. As for the first experiment, we compare the performance to MPC. In this experiment, the initial position, the goal location, and the position of five obstacles were randomized. The results reveal that our extension of optimization fabrics to non-holonomic robots maintains similar results as with a robotic arm. Specifically, computational time can be reduced to optimization-based methods while maintaining good performance in terms of safety and goal-reaching, Fig. 5.15. We can also observe that success rate with SF is lower compare to MPC due to a high number of unreached goals.

5.4.6. EXPERIMENT 5: MOBILE MANIPULATORS

In the final experiment, we assess the applicability of SF and DF to a non-holonomic mobile manipulator. In an environment that is densely occluded by obstacles, the motion planning problem is defined by a desired end-effector position and additional path constraints (e.g. desired orientation of the end-effector).

Simulation In simulation, we evaluate the performance of our extension to non-holonomic mobile manipulators with SF. In this series, the positions of 8 obstacles are randomized for $N = 50$ cases. The workspace was limited to a 7mx7m square, so that random obstacles are ensured to be actually hindering the motion planner. The results reveal that properties shown in the previous experiments transfer to more complex systems without loss of the computational benefit, Fig. 5.16. In this series, there were 1 unreached goals and 4 collisions which are, similar to the previous experiments, caused by local minima. Local





(a) Metrics evaluation for successful experiments

(b) Success results

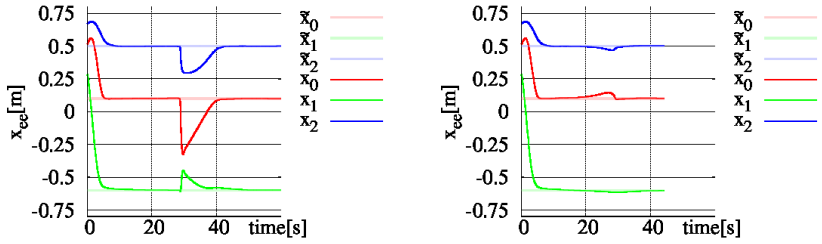
Figure 5.13: Comparison between SF and DF for real-world scenarios with dynamic obstacles.

minima are more likely for mobile manipulators as their workspace is larger. Combining our contributions, DF and the extension to non-holonomic robots, we achieve reactive and safe behavior in dynamic environments. Moving obstacles are avoided in a natural way using our method, see Fig. 5.17.

Real-World We present qualitative results for a non-holonomic mobile manipulator using DF. In Fig. 5.1, the robot follows a trajectory defined by a basic spline, while additionally respecting an orientation constraints on its end-effector and avoiding the shelves and an obstacle on the ground. The end-effector trajectory is plotted in Fig. 5.18.

5.4.7. EXPERIMENT 6: DYNAMIC FABRICS IN SUPERMARKETS

In this experiment, we show qualitatively how DF could be used in collaborative environments where humans and robots coexist. For this experiment, we give the robot a static goal pose similar to a pickup setup. The same environment is shared with a co-worker who restocks a shelf. The right hand of the human is tracked with a motion capture system. The hand is then avoided by the robot using DF, see Fig. 5.4. In this experiment, the minimum distance between the robot and the hand was 0.062m. This real-world experiment showcases potential applications of the proposed method.



(a) Static Fabric

(b) Dynamic Fabric

Figure 5.14: Trajectories for real panda robot in the presence of a dynamic obstacle. DF show a smoother and in-advance reaction to the approaching obstacle while SF can only react in sudden motion.

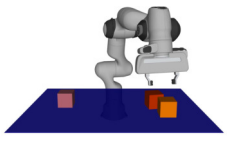
5.5. CONCLUSION

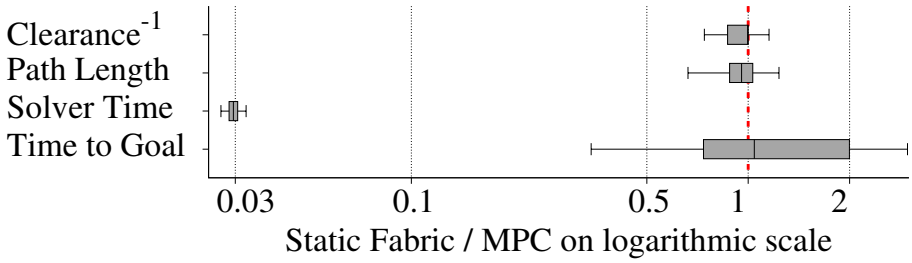
In this paper, we have generalized optimization fabrics to dynamic environments. We have proven that our proposed Dynamic Fabrics are convergent to reference paths and can thus compute motion for path following tasks (Lemma 5.2.10). Besides, we have proposed an extension to optimization fabrics (and thus also DF) for non-holonomic robots. This allows the application of this framework to a wider range of robotic applications and ultimately allows the deployment to many mobile manipulators in dynamic environments.

These theoretical findings were confirmed in various experiments. First, the quantitative comparisons showed that Static Fabrics outperforms MPC in terms of solver time while maintaining similar performance in terms of goal-reaching and success rate. The improved performance with optimization fabrics might be caused by the different metric for goal reaching compared to Model Predictive Control. An integration of non-Riemannian metrics into an MPC formulation should be further investigated in the future.

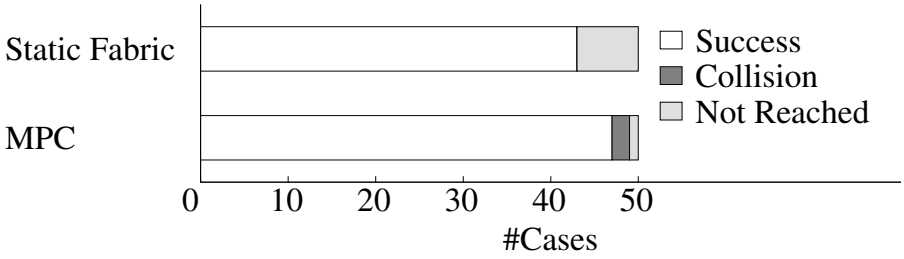
Verifying our theoretical derivations for DF, the experiments showed that the deviation error for path following tasks is decreased compared to SF. Similarly, environments with moving obstacles and humans showed increased clearance while maintaining low computational costs and execution times. Thus, DF overcome an important drawback of SF [18], [28], where collision avoidance with moving obstacle is solved purely by the high frequency at which optimization fabrics can be computed. Moreover, the generalization did not increase the solving time compared to SF. Unlike the original work on optimization fabrics, this generalization allows the deployment to dynamic environments where velocity estimates of moving obstacles are available.

Direct sensor integration in optimization fabrics might be feasible in future works to overcome the shortcomings of perception pipelines for collision avoidance. For the trajectory path tasks in this paper, we used a simple global path generated in workspace. As DF integrate global path in arbitrary manifolds, improving the global planning phase could be further investigated. We expect this to be beneficial when robotics tasks are constantly changing and task planning is required.





(a) Metrics evaluation for successful experiments



(b) Success results

Figure 5.15: Results for randomized cased with the Clearpath Boxer robot. Similar performance in terms of safety and goal-reaching can be combined with very fast computation using optimization fabrics.

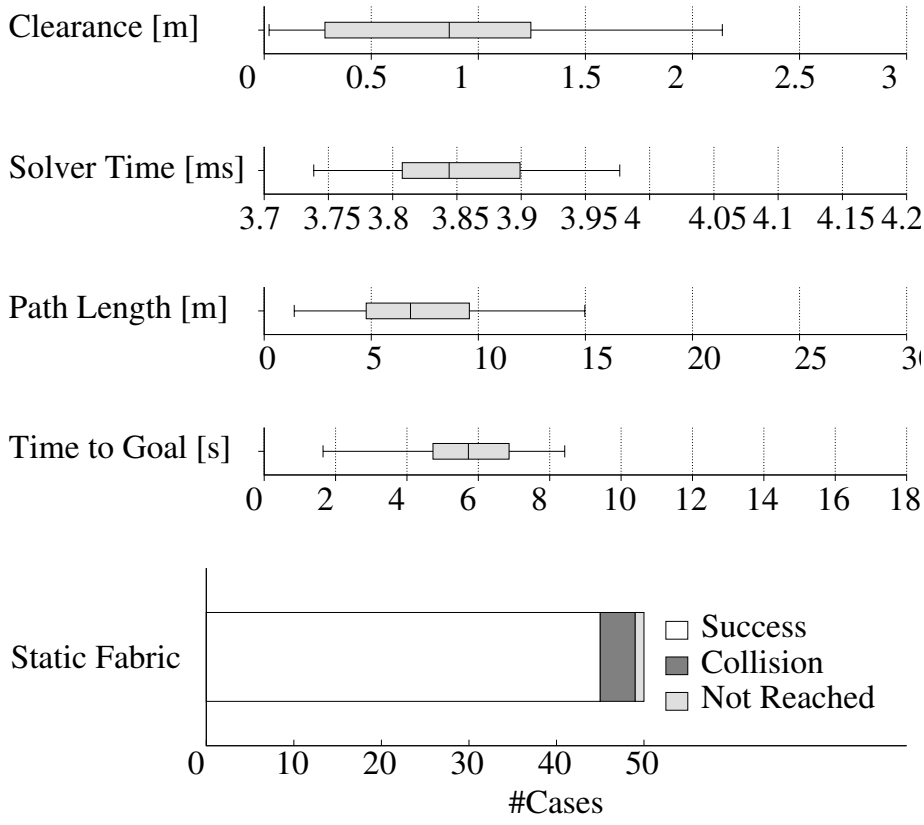
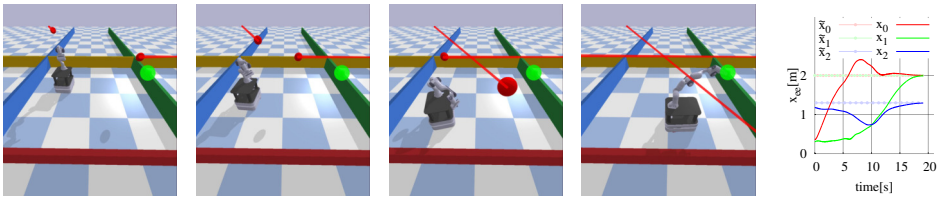
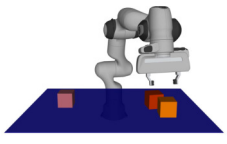


Figure 5.16: Quantitative results with static fabrics for a non-holonomic mobile manipulator in simulation. Fabrics solve planning problems in randomized environments in low planning time. This allows whole-body control and highly reactive behavior.



(a) $t = 0$s (b) $t = 7$s (c) $t = 14$s (d) $t = 20$s (e)

Figure 5.17: Sequence of trajectory computed with DF for a mobile manipulator in simulation with moving obstacles (red sphere with line indicating the past trajectory) and one end-effector goal (green). The trajectory of the end-effector are visualized in (e) as $\boldsymbol{x}$ and the desired end-effector position as $\tilde{\boldsymbol{x}}$.



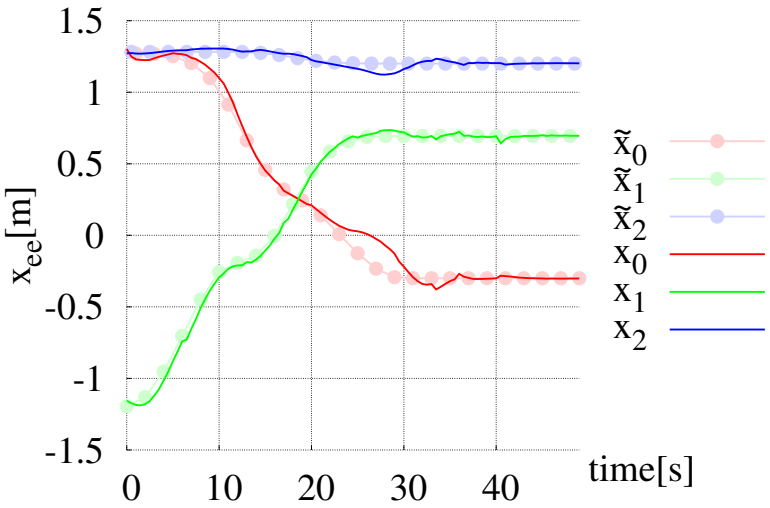


Figure 5.18: Real-world experiment for path following with a mobile manipulator. The global path can be tracked accurately by DF including the extension to non-holonomic robots. The scene is visualized in Fig. 5.1.

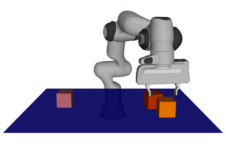
6

Autotuning Symbolic Optimization Fabrics for Trajectory Generation

The previous chapter introduced Dynamic Fabrics (DF) as a generalization of Optimization Fabrics (fabrics) to better cope with dynamic environments for mobile manipulation. Although theoretically powerful, fabrics, and thus also DF, seem to be practically hard to tune correctly. This chapter introduces a symbolic formulation of fabrics to reduce the computational costs at runtime and simplify parameter tuning during execution. We implement a Bayesian parameter tuning method, similar to hyperparameter tuning in machine learning, to reach expert-level performance. We show the effectiveness in real-world experiments and across different robot embodiments.

This chapter is a verbatim copy of the peer-reviewed publication:

- ☞ **M. Spahn** and J. Alonso-Mora. *"Autotuning Symbolic Optimization Fabrics for Trajectory Generation"*. IEEE International Conference on Robotics and Automation, 2023.



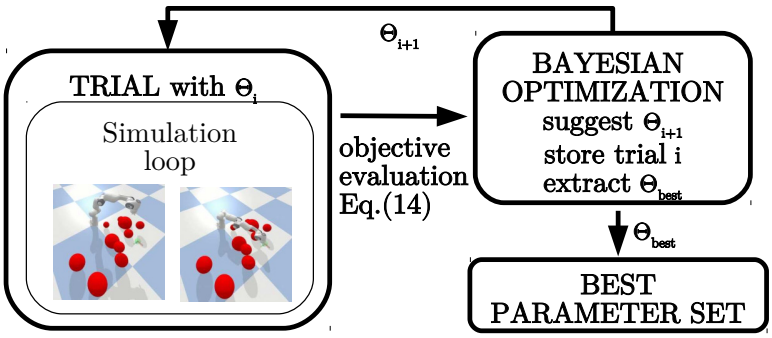


Figure 6.1: Overview of one trial in the tuning pipeline for symbolic optimization fabrics. The objective function is evaluated after an entire trial run is simulated. Using Bayesian optimization, a new parameter set is suggested based on the history of trials. The best parameter set is extracted from all trials.

6.1. INTRODUCTION

Mobile manipulation is the field of robotics concerned with highly capable robots characterized by their locomotion and manipulation ability. Such robots are getting ever more attention as they will be deployed to human-shared environments, like households or warehouses. In such dynamic environments, fast trajectory generation is crucial to avoid collisions and react quickly to changing goal definitions.

Trajectory generation is often addressed by solving an optimization problem that consists of a scalar objective function – the dynamics or transition function – and several constraints. As the degrees of freedom and number of constraints increase, solving that problem in real-time becomes challenging. This is especially limiting in the case of mobile manipulation [8]. Optimization fabrics represent a different approach to the problem, as they formulate trajectory generation as the shortest-geodesic-problem in a manifold of the configuration space [29].

With optimization fabrics, different components, or desired behaviors, such as collision avoidance and joint limit avoidance, are combined using Riemannian metrics. As the structure of the resulting trajectory generation methods remains unchanged across all time steps, it can be composed before runtime, thus saving computational costs during executing. Optimization fabrics, but also their predecessor Riemannian Motion Policies (RMPs), have shown impressive results for several manipulator applications, including dynamic and crowded environments [28], [109], [110]. However, despite their theoretical properties of inherent collision avoidance and convergence, these methods require expertise and intuition to tune individual components to generate smooth and well-behaving trajectories.

Contributions: To address this issue, we formulate optimization fabrics as a **symbolic trajectory generation** method. Precisely, the combination of the individual components (joint limit avoidance, goal reaching, collision avoidance, etc.) is performed in a parameterized way before runtime. Separating composition and evaluation allows for changing the individual parameters at runtime while achieving low computational costs. Additionally, this allows formulating parameter-tuning as a constrained optimization problem. Solving this problem effectively **automates the tuning process** systematically using Bayesian optimization. We show that automated tuning requires only few trials to achieve similar per-

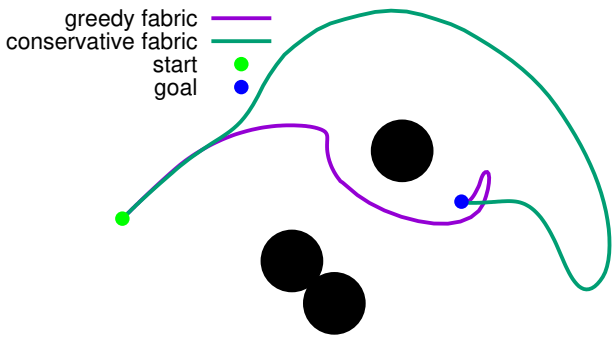


Figure 6.2: Two different parameter sets for optimization fabrics given the same problem. While the greedy tuning is more aggressive (purple), the more conservative tuning results in a smoother trajectory (green).

formance to an expert in the field, and systematically outperforms a randomized parameter setting. Moreover, we show that one parameter tuning generalizes across different robots, to some extent, across different tasks and between simulation and real world. Finally, we demonstrate how **coupled mobile manipulation** with a differential drive can be achieved using autotuned optimization fabrics for in-store order-picking integrating visual servoing.

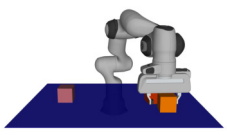
6.2. RELATED WORKS

6.2.1. AUTOTUNING FOR TRAJECTORY GENERATION

Autotuning can be beneficial for trajectory generation when using model predictive control. In [111], an autotuned model predictive controller has outperformed a manual tuned controller of the same kind by 25%. Jointly optimizing parameters and the model of the controller, *AutoMPC* showed the benefit of parameter tuning in the context of simultaneous system identification and control [112]. These methods are explicitly formulated for model predictive control and do not transfer easily to other trajectory generation methods. In contrast, we propose a generic parameter optimization approach to trajectory generation.

6.2.2. HYPERPARAMETER TUNING IN MACHINE LEARNING

Within the machine-learning field, hyperparameter tuning has shown to be highly important for all different kinds of applications [113]–[115]. Parameter optimization aims to minimize training costs while achieving the best possible performance. Hyperparameter tuning is most valuable in extremely costly applications such as reinforcement learning [116]. Generally, two different search algorithms have been investigated: grid search and random search [117]. Current state-of-the-art methods for parameter search are based on random search with a Bayesian optimizer [115], [118]. While the machine-learning community has largely agreed on the importance of parameter tuning, systematic tuning of trajectory generation methods are not well established. In this paper, we showcase, with the example of optimization fabrics, how important parameter tuning is and how trajectory generation can benefit from it.



6.3. OVERVIEW

In this paper, we first recall very briefly the theory of optimization fabrics and the steps to use it for trajectory generation (Section 2.4). Then, we formulate optimization fabrics as a symbolic trajectory generator, so that combining of individual components is only performed once (Section 6.4). Then, we formulate parameter tuning for trajectory generation as a constrained optimization problem and propose Bayesian optimization for effective autotuning (Section 6.5). As an example, we apply this autotuning to symbolic optimization fabrics (Section 6.6), but it is generally independent of the trajectory generator at hand.

6.4. SYMBOLIC FABRICS

A trajectory generator that is based on optimization fabrics is composed of several components, such as collision avoidance, joint limit avoidance, goal attraction, etc. Each component contributes to the resulting optimization fabric through the metric-weighted summation that creates the tree of fabrics. The trajectory generator is parameterized by the individual terms of the components. Here, we lay out the parameterization for collision avoidance, joint limit avoidance, self-collision avoidance, and speed-control. In our framework, the tree of fabrics is generated before runtime as a symbolic expression, to which the parameters are set at runtime. Note that the approach of symbolic pre-solving results in much higher planning frequencies. In the following, we explain the individual parameters that we exposed symbolically. The form of the individual terms is adapted from [18], [28], [109] but written in a symbolic form.

Basic inertia The final tree of fabrics is equipped with a basic inertia metric that indicates how reactive the entire motion is. This basic inertia metric is derived from the symbolic Finsler structure: $\mathcal{L}_e = 0.5 m_{\text{base}} \dot{\boldsymbol{q}}^T \boldsymbol{I} \dot{\boldsymbol{q}}$.

Collision avoidance For collision avoidance, the task manifold $\mathcal{X}$ is defined by the distance function between an obstacle and a robot link. The differential map used is defined as

$$\phi_{\text{i}}(\boldsymbol{q}) = \frac{\|\text{fk}_{\text{i}}(\boldsymbol{q}) - \boldsymbol{x}_{\text{obst}}\|}{r_{\text{obst}} + r_{\text{i}}} - 1,$$

where $\text{fk}_{\text{i}}(\boldsymbol{q})$ is the positional forward kinematics for link i in a configuration $\boldsymbol{q}$, r_{obst} and r_{i} are the radii of the englobing spheres for the obstacle and the link respectively. While this mapping between configuration space and task manifold is different for each obstacle and each collision link of the robot, the geometry and metric are the same for all of them. For the geometry $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{0}$, we use the parameterized forcing term

$$\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{-k_{\text{geo,col}}}{\boldsymbol{x}^{\beta_{\text{geo,col}}}} \dot{\boldsymbol{x}}^2, \tag{6.1}$$

where $k_{\text{geo,col}}$ and $\beta_{\text{geo,col}}$ are parameters of the trajectory generator. Generally, we use k and β for proportional parameters and exponential parameters. The Finsler structure for collision avoidance is parameterized as

$$\mathcal{L}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{k_{\text{fin,col}}}{\boldsymbol{x}^{\beta_{\text{fin,col}}}} \left(-0.5(\text{sgn}(\dot{\boldsymbol{x}}) - 1)\right) \dot{\boldsymbol{x}}^2, \tag{6.2}$$

where $\text{sgn}(\dot{\boldsymbol{x}})$ is the signum-operator returning the sign of $\dot{\boldsymbol{x}}$.

Self-collision avoidance For self-collision avoidance, the task manifold $\mathcal{X}$ is defined similarly to collision avoidance:

$$\phi_{i,j} = \frac{\|\text{fk}_i(\boldsymbol{q}) - \text{fk}_j(\boldsymbol{q})\|}{r_i + r_j} - 1,$$

where $\text{fk}_i(\boldsymbol{q})$ and $\text{fk}_j(\boldsymbol{q})$ are the positional forward kinematics of the two links for a self-collision pair and r_i and r_j are the radii for both englobing spheres. The geometries are defined analogously

$$\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{-k_{\text{geo,self}}}{\boldsymbol{x}^{\beta_{\text{geo,self}}}} \dot{\boldsymbol{x}}^2. \tag{6.3}$$

The Finsler structure for collision avoidance is parameterized as

$$\mathcal{L}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{k_{\text{fin,self}}}{\boldsymbol{x}^{\beta_{\text{fin,self}}}} \left(-0.5(\text{sgn}(\dot{\boldsymbol{x}}) - 1)\right) \dot{\boldsymbol{x}}^2. \tag{6.4}$$

Joint limit avoidance For joint-limit avoidance, two simple differential maps denoting the distance to the joint limits are used, specifically

$$\phi_{\text{limit,i,lower}}(\boldsymbol{q}) = \boldsymbol{q}_i - \boldsymbol{q}_{\text{min,i}}, \forall i \in (1, \ldots, n)$$
$$\phi_{\text{limit,i,upper}}(\boldsymbol{q}) = \boldsymbol{q}_{\text{max,i}} - \boldsymbol{q}_i, \forall i \in (1, \ldots, n).$$

Similar to collision avoidance, we use the parameterized forcing term

$$\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{-k_{\text{geo,limit}}}{\boldsymbol{x}^{\beta_{\text{geo,limit}}}} \dot{\boldsymbol{x}}^2 \tag{6.5}$$

and the Finsler structure

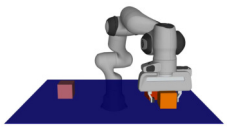
$$\mathcal{L}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{k_{\text{fin,limit}}}{\boldsymbol{x}^{\beta_{\text{fin,limit}}}} \left(-0.5(\text{sgn}(\dot{\boldsymbol{x}}) - 1)\right) \dot{\boldsymbol{x}}^2. \tag{6.6}$$

Speed control As the root of the tree of fabrics is a frictionless fabric, it only converges if damped [18]. Constant damping is sufficient to achieve the theoretical properties that are needed for trajectory generation. However, [18], [27], [28] proposed enhanced damping under the name of *speedcontrol*. We employ the same damping strategy while adding parameterization. The technique is based on a dynamic damping modification based on the distance to the goal. Specifically, the final optimization fabric is damped according to

$$\ddot{\boldsymbol{q}} = -\boldsymbol{h}_2 - \boldsymbol{M}^{-1} \partial_{\boldsymbol{q}} \boldsymbol{\psi} + \alpha_{\text{ex}} \dot{\boldsymbol{q}} - \beta \dot{\boldsymbol{q}},$$

where $\boldsymbol{h}_2$ is the sum of all pulled forcing terms, $\boldsymbol{M}$ is the sum of all metrics of the individual geometries, $\partial_{\boldsymbol{q}} \boldsymbol{\psi}$ is the goal attraction term pulled in the configuration space, α_{ex} is a weighted sum of α_{ex}^0 that maintains constant execution energy without goal attraction and $\alpha_{\text{ex}}^{\boldsymbol{\psi}}$ that maintains constant execution energy with goal attraction:

$$\alpha_{\text{ex}} = \boldsymbol{s}_\eta(\mathcal{L}_{ex}) \alpha_{\text{ex}}^0 + (1 - \boldsymbol{s}_\eta(\mathcal{L}_{ex})) \alpha_{\text{ex}}^{\boldsymbol{\psi}}.$$



Then, β is the damping term, computed as:

$$\beta = \boldsymbol{s}_\beta(\boldsymbol{q})\boldsymbol{B}_{\text{max}} + \boldsymbol{B}_{\text{min}} + \max(0, \alpha_{\text{ex}} - \alpha_{\mathcal{L}_e}),$$

where $\boldsymbol{B}_{\text{max}}$ and $\boldsymbol{B}_{\text{min}}$ are the upper and lower damping values and $\alpha_{\mathcal{L}_e}$ is the energization coefficient maintaining constant system energy (not execution energy) without goal attraction. The switching functions $\boldsymbol{s}_\beta(\boldsymbol{q})$, $\boldsymbol{s}_\eta(\boldsymbol{q})$ are further parameterized as

$$\begin{aligned}\boldsymbol{s}_\beta(\boldsymbol{q}) &= 0.5(\tanh -\alpha_\beta(\|\boldsymbol{q}\| - r_{shift})) + 1\\ \boldsymbol{s}_\eta(\mathcal{L}_{ex}) &= 0.5(\tanh(-0.5\mathcal{L}_{ex}(1 - v_{ex}) - 0.5) + 1),\end{aligned}$$

where r_{shift} determines the distance to the goal at which the switch between $\boldsymbol{B}_{\text{min}}$ and $\boldsymbol{B}_{\text{max}}$ occurs, α_β is the steepness of that switching, $\mathcal{L}_{ex}$ is the user-defined execution energy (usually a simple kinetic energy in joint space) and v_{ex} is the execution energy factor, i.e. it determines the desired speed of motion. For a detailed discussion on speed control with optimization fabrics, we refer to previous works on optimization fabrics [18], [28].

We group all parameters resulting from the symbolic fabrics defined here into a vector of parameters $\boldsymbol{\Theta}$. All parameters are listed in Table 6.1.

6.5. PARAMETER TUNING AS AN OPTIMIZATION PROBLEM

We define parameter tuning as a constrained optimization problem:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} c(\boldsymbol{\Theta}),\ \text{s.t}\ \boldsymbol{\Theta}_{\text{min}} < \boldsymbol{\Theta} < \boldsymbol{\Theta}_{\text{max}}, \tag{6.7}$$

where $\boldsymbol{\Theta}_{\text{max}}$ and $\boldsymbol{\Theta}_{\text{min}}$ are the upper and lower bounds of the parameters. The objective $c(\boldsymbol{\Theta})$ is a function of the parameters specifying the tree of fabrics and can be evaluated after one trajectory planning problem has finished. We call the evaluation of one parameter set a *trial*. Next, we propose an objective function that is flexible as different scenarios may require different parameter tuning.

6.5.1. OBJECTIVE

The objective function $c(\boldsymbol{\Theta})$ is a weighted sum of several metrics, that are invariant to the robot:

$$c(\boldsymbol{\Theta}) = w_{\text{distance}}c_{\text{distance}} + w_{\text{path}}c_{\text{path}} + w_{\text{clearance}}c_{\text{clearance}}. \tag{6.8}$$

c_{distance} accounts for the normalized, summed distance to the goal over one trial and is defined as

$$c_{\text{distance}} = \frac{\sum_{i=0}^{T} \|\boldsymbol{x}_i - \boldsymbol{x}_{\text{goal}}\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}_{\text{goal}}\|}, \tag{6.9}$$

where $i \in [0, T]$ are the discretized time steps and $\boldsymbol{x}_{\text{goal}}$ is the goal of the motion planning problem. c_{path} accounts for the normalized path length over one trial and is defined as

$$c_{\text{path}} = \frac{\sum_{i=1}^{T} \|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|}{\|\boldsymbol{x}_0 - \boldsymbol{x}_{\text{goal}}\|}. \tag{6.10}$$

$c_{\text{clearance}}$ accounts for the average clearance to obstacles over one trial and is defined as

$$c_{\text{clearance}} = \frac{1}{T}\sum_{i=1}^{T} \min_{o^j} \left\| \boldsymbol{x}_i - o_i^j) \right\|, \tag{6.11}$$

where o_i^j is the position of obstacle j at time step i. Each of these terms is evaluated after an entire trial that was obtained by a specific set of parameters.

Algorithm 2: Autotuning for trajectory generators

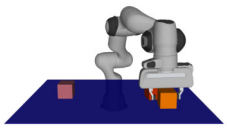
```
Formulate trajectory generator with parameters Θ
Define parameter space by Θ_min, Θ_max
Formulate objective c(Θ)
Initialize objective function estimate c̃(Θ)
for i = 0 to N do
    Suggest parameter Θ_i based on c̃(Θ)
    for t = 0 to T do
        Compute action with parameter set Θ_i
        Apply action to robot
        Store observation relevant for metrics
    end
    Evaluate c(Θ_i)
    Update c̃(Θ)
end
Extract the best parameter set Θ_best
```

6.5.2. BAYESIAN OPTIMIZATION

In the tuning phase, the problem specification for the investigated scenario, e.g., the goal and obstacle positions, across all trials during tuning remains the same while $\boldsymbol{\Theta}$ are optimized according to the objective. To solve the Bayesian optimization we employ the *Tree-structured Parzen Estimator* as it has shown improved performance over grid-search and conventional random search in machine learning applications [118], [119]. To deploy this technique we used Optuna, a hyperparameter optimization framework initially designed for machine learning applications [115]. The general setup for one trial is shown in Fig. 6.1 and the procedure is summarized in Algorithm 2.

6.6. EXPERIMENTAL RESULTS

We showcase our parameter optimization method for symbolic fabrics. The search space for the parameters is summarized in Table 6.1. We first analyze the importance of tuning for optimization fabrics on the performance of trajectory generation. Then, we investigate how tuned parameters can be transferred across different robots (Section 6.6.3), different scenarios (Section 6.6.4), and between simulation and real world (Section 6.6.5).



Parameter	boundaries	type	distribution	manual
m_{base}	$[0, 1]$	float	uniform	0.2
$k_{\text{geo,col}}$	$[0.01, 1]$	float	log	0.03
$k_{\text{geo,limit}}$	$[0.01, 1]$	float	log	0.3
$k_{\text{geo,self}}$	$[0.01, 1]$	float	log	0.03
$k_{\text{fin,col}}$	$[0.01, 1]$	float	log	0.03
$k_{\text{fin,limit}}$	$[0.01, 1]$	float	log	0.05
$k_{\text{fin,self}}$	$[0.01, 1]$	float	log	0.03
$\beta_{\text{geo,col}}$	$[1, 5]$	int	uniform	3
$\beta_{\text{geo,limit}}$	$[1, 5]$	int	uniform	2
$\beta_{\text{geo,self}}$	$[1, 5]$	int	uniform	3
$\beta_{\text{fin,col}}$	$[1, 5]$	int	uniform	3
$\beta_{\text{fin,limit}}$	$[1, 5]$	int	uniform	3
$\beta_{\text{fin,self}}$	$[1, 5]$	int	uniform	3
α_β	$[0, 1]$	float	uniform	0.5
$\boldsymbol{B}_{\text{min}}$	$[0, 1]$	float	uniform	0.01
$\boldsymbol{B}_{\text{max}}$	$[5, 20]$	float	uniform	6.5
r_{shift}	[0.01, 0.1]	float	uniform	0.05
v_{ex}	[1.0, 30]	float	uniform	15.0

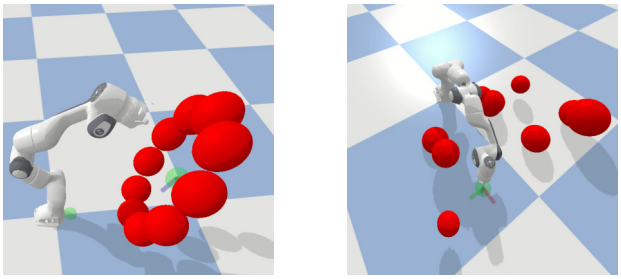
Table 6.1: Search space for parameters. Some parameters are restricted to integers, and for some a log-distribution is applied.

6.6.1. EXPERIMENTAL SETUP

The method was tested in simulation and in the real world on a Panda robot and a mobile manipulator composed of a Clearpath Boxer and a Panda robot. The simulation uses the pybullet physics engine with an interface through OpenAI-gym [120]. The different motion planning goals evaluated in this paper are: (a) reaching an end-effector pose inside a ring of obstacles (Fig. 6.3a) (similar to the experiment in [28]) and (b) reaching an end-effector pose above a surface with random obstacles (Fig. 6.3b). The two scenarios will be referred to as *reaching-in-ring* and *reaching-on-table*, see Section 6.6.1. Unless stated otherwise, the weights are set to $w_{\text{path}} = 0.1, w_{\text{clearance}} = 0.2, w_{\text{distance}} = 0.7$. We also use this weighted sum as the performance metric. While these weights are chosen arbitrarily in this work to demonstrate the usefulness of autotuning, they should be derived from a human evaluator in a more realistic scenario. We refer with *manual* to an expert-tuning, see Table 6.1 for specific parameters. During testing, the trial was randomized with changing obstacles and goals. For autotuning on the robotic arms, we consistently used $N = 60$ trials, although the best parameter set is usually reached earlier, see Fig. 6.4.

6.6.2. IMPORTANCE OF TUNING

We compare the autotuned parameters with seven random parameter sets from the search space and a manually tuned parameter set that we obtained through expertise in previous works like [110]. In this experiment, tuning and testing are performed on the test scenario *reaching-in-ring*. Tuning is crucial for optimization fabrics, as the performance with a random parameter set cannot compete with tuning, Fig. 6.5. This result was expected and



(a) *reaching-in-ring* (b) *reaching-on-table*

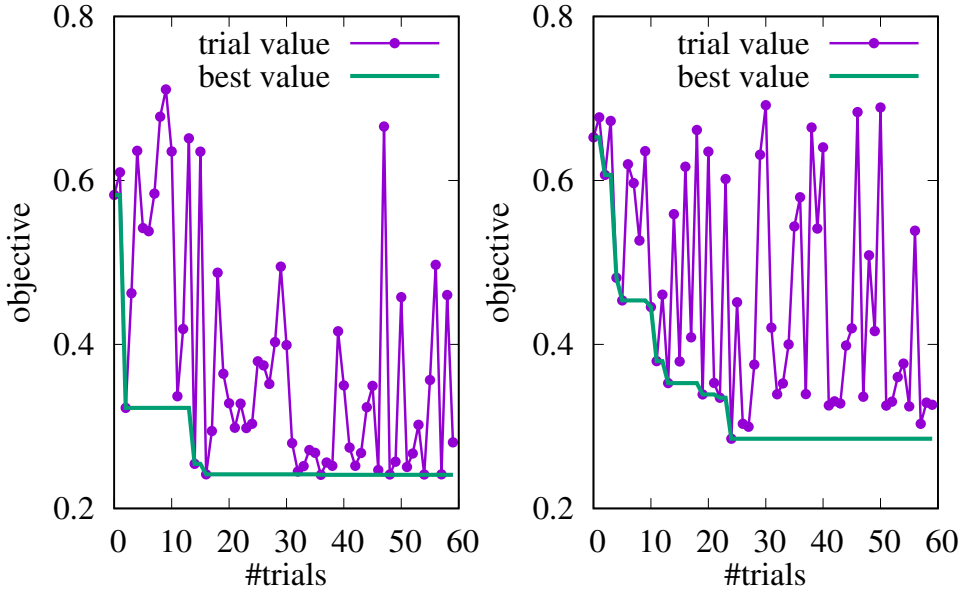
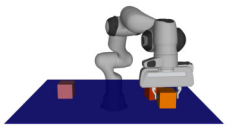


Figure 6.4: Optimization history for simulation (left) and real world (right) for panda robot in *reaching-in-ring* scenario.



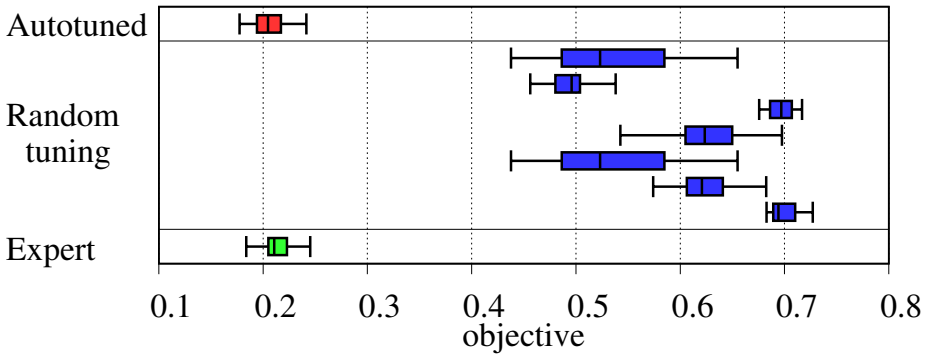


Figure 6.5: Evaluation for scenario *reaching-in-ring* autotuned parameters and compared to random parameter selection and manual tuning. Autotuning is able to systematically outperform random parameter sets and reach expert level tuning.

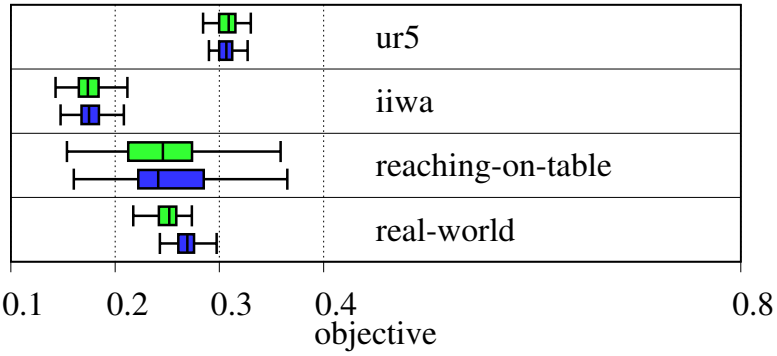


Figure 6.6: The autotuned for the panda robot in simulation for the *reaching-in-ring* scenario on modified scenarios (blue) is compared to autotuned parameter sets obtained on these scenarios directly (green). Exchanging the robot (ur5, iiwa) and changing the scenario (*reaching-on-table*) results in a very small loss in performance, while the loss is higher when parameters are transferred between simulation and real world (real-world).

should only demonstrate that the right parameter set is required to deploy this method. Autotuned parameters reach a similar performance to the expert. This result highlights the importance of tuning for optimization fabrics and shows that autotuning is an effective way to obtain parameter sets for novice users of optimization fabrics.

6.6.3. CROSS VALIDATION: TRANSFER ACROSS ROBOTS

Without any retuning, we deploy the symbolic optimization fabrics planner tuned on the Panda robot on two other robots with similar specifications (Kuka LBR IIwa 7, Universal Robot UR5) and compare the performance with tuning performed on the respective robot. Specifically, we do not change the leaf geometries and energies but change differential maps according to relevant collision links on the robot at hand. From Fig. 6.6, we conclude that tuning is independent of the robot. This can be explained by the fact, that optimization fabrics are a purely geometric approach to trajectory generation and the different dimension of the robots do not change the dynamical system enforced onto the robot.

6.6.4. CROSS VALIDATION: TRANSFER ACROSS SCENARIOS

In the third experiment, we evaluate how well an autotuned parameter set transfers to a different scenario. In the specific example, we use the tuning obtained from the *reaching-in-ring* case and test it on *reaching-on-table*. Performance can be transferred smoothly if the objective remains the same, see Fig. 6.6. However, note that different scenario might require generally slower motion because of a more crowded environment. Such a step would require to retune the parameters according to the new objective.

6.6.5. CROSS VALIDATION: TRANSFER REAL WORLD

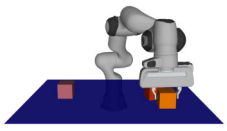
As optimization fabrics are a geometric method [28], they should be independent of the robot embodiment. Relying on the low-level controller. In this paper, we investigate how the performance is affected by the transfer from the simulation environment to the real world. Performance benefits from tuning in the real world highlight that low-level controller differences affect the behavior, see Fig. 6.6. Specifically, the accumulated distance to the goal is increased (0.14m tuned in the real world vs 0.16m tuned in simulation) when tuning is transferred between simulation and real world. Thus, there is added value in tuning in the real world. Our framework offers to quickly tune fabrics in the real-world using the *fabrics-ros-bridge*. With relaxed performance requirements, it is sufficient to tune in simulation.

6.6.6. CROSS VALIDATION: TRANSFER MOBILE MANIPULATOR

Finally, we qualitatively test the performance of the tuning method on a real mobile manipulator with 10 degrees of freedom. After only $N = 30$ trials, the robot was able to perform coupled mobile manipulation based on a visual servoing approach [121]. Symbolic optimization fabrics are especially suited for visual servoing as their symbolic character allows them to constantly update the position of the goal. A video of this experiment is attached to the paper.

6.7. CONCLUSION

We formulated parameter tuning for trajectory generation as a constrained optimization problem. Additionally, we introduced symbolic optimization fabrics that implement optimization fabrics in a parameterized way, for which the general structure is pre-solved. The trajectory generator obtained with this technique is parameterized and achieves low computational costs at runtime. We showed that parameter tuning for symbolic optimization fabrics can be effectively solved using Bayesian optimization. Additionally, we have shown that the tuning generalized across different robots, tasks, and between simulation and the real world. Finally, we qualitatively demonstrated that the method applies to mobile manipulators. While we aim at developing a method-agnostic autotuning framework for motion generation, symbolic optimization fabrics were selected as an example in this work.



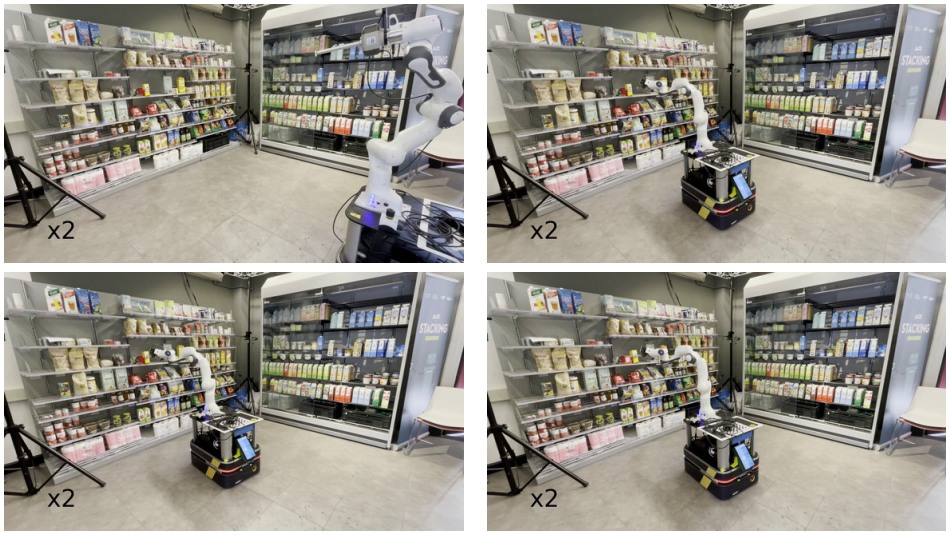


Figure 6.7: Trajectory generation with optimization fabrics for mobile manipulator using visual serving for product picking.

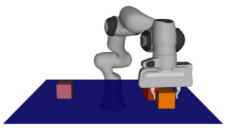
7

Overcoming Explicit Environment Representations with Geometric Fabrics

In the previous chapters, advancements to the framework of Optimization Fabrics (fabrics) have been proposed to allow for the deployment into dynamic environments and the automation of tuning. However, all of these methods, relied on simplistic environment representations, usually composed of primitive shapes. In this chapter, we integrate implicit environment representations known from mobile robotics into the framework of fabrics. We show that low computational costs and the existence of a closed-form solution can be exploited to relax the requirements on the perception pipeline. We quantitatively compare different levels of expressiveness, starting from shape-based collision avoidance and finishing with raw point clouds.

This chapter is a verbatim copy of the submitted work:

- ☞ **M. Spahn** and J. Alonso-Mora. *"Overcoming Explicit Environment Representations with Geometric Fabrics". Submitted to* IEEE Robotics and Automation Letters, 2024.



7.1. Introduction

As robots make their way into human shared environments, fast reactive behavior is needed to ensure that collisions are avoided at all time. Trajectory Generation (TG) is commonly formulated as a receding horizon optimization problem, where the robot's trajectory is optimized over a short time horizon. Such methods are known as Model Predictive Control (MPC) and have shown great success for autonomous vehicles and drones where the dimension of the configuration space remains small. For higher dimensional configuration spaces, e.g. manipulators and mobile manipulators, the computational cost of MPC scale poorly, leading to slower computation cycles and ultimately to a less reactive behavior. Optimization Fabrics (fabrics) offer an alternative to these approaches. Based on differential geometry, policies are composed of several components to form a highly reactive and fast behavior. However, the composition of fabrics of individual obstacle avoidance geometries is limited to simple geometric shapes, such that a differentiable distance function can easily be formulated. This *explicit* environment representation sets a challenging requirement on the perception part of the motion generation pipeline. In this work, we present and analyze three different representations of the environment to overcome this drawback, namely Free Space Decomposition (FSD), Signed Distance Fieldss (SDFs) and raw sensor data, from visual sensors, such as cameras or lidars, into the framework of fabrics. We refer to these representations as *implicit*. Generally, the more implicit an environment representation is, the more computational costs are moved from the perception pipeline to the planner. In the process, we derive essential extensions to the framework and analyze strengths and weaknesses of the individual methods. To summarize, this paper makes the following contributions:

- We integrate implicit representation of the environment into the framework of fabrics, namely FSD, SDF and raw sensor data.
- We derive how numerical gradients can be used for pullback operations which are essential in the composition of fabrics.
- We analyze the strengths and weaknesses of the three representations in different environments, including moving obstacles, and with various robot morphologies.
- We present real-world experiments illustrating the power of our open-source implementation.

Implicit Environment Represenations for trajectory generation While fabrics mainly employ explicit environment representations [18], [110], new TG methods lean towards implicit approaches, see Fig. 7.1. For example, representing the environment's free space as a set of half-planes has proven sucessful for whole-body MPC formulations [69]. In drone flight, a similar concept generates safe flight zones along a global path [123], [124]. In the context of drone flying, SDF has been utilized with MPC in unknown environments [122]. Raw lidar data has been used in combination with Riemannian Motion Policiess (RMPss) showing impressively high frequencies when computation is parallized on GPU [125]. Recent advances in sampling-based MPC, also referred to as Model Predictive Path Integral Control (MPPI), utilizing physics engines for collision avoidance [126], integrate obstacle collisions in cost functions during trajectory rollouts. A similar approach is seen

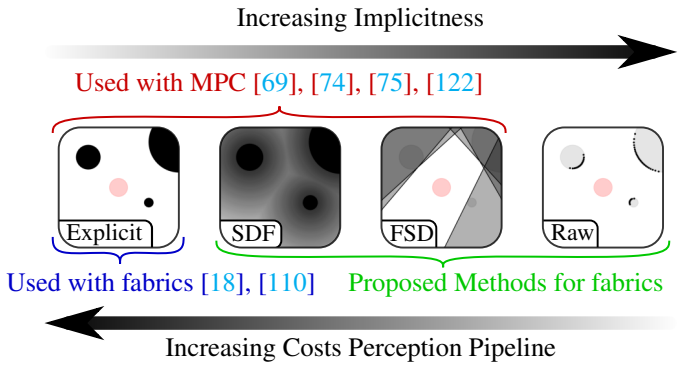


Figure 7.1: Different levels of implicitness for environment representations.

in [127].

7.2. METHODS

As this work does not alter the fundamental principles of fabrics, this section focuses on the integration of three different implicit environment representations into the framework. In the process, we lay out the necessary concepts required for the representations, and provide the tools to combine them with fabrics. The core idea however is identical for all presented methods. Collision avoidance is realized by defining a differentiable mapping ϕ from the configuration space $\mathcal{Q}$ to the distance manifold between the robot and the environment. On that manifold, the desired behavior is encoded using a geometry and a energizing Finsler to form a geometric fabric of form

$$\mathcal{S} = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}\,.$$

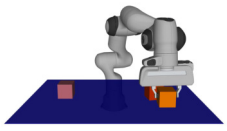
When being combined with other behaviors, this fabric is *pulled* back into the configuration space $\mathcal{Q}$ to be *summed* up with other fabrics. To recall, the *pullback* is defined as

$$\text{pull}_{\phi}(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}} = \left(\boldsymbol{J}_{\phi}^{T}\boldsymbol{M}\boldsymbol{J}_{\phi}, \boldsymbol{J}_{\phi}^{T}(\boldsymbol{f} + \boldsymbol{M}\dot{\boldsymbol{J}}_{\phi}\dot{\boldsymbol{q}})\right)_{\mathcal{Q}}.$$

The difference between the methods presented in this work is the definition of the mapping ϕ and therefore the computation of the gradient $\boldsymbol{J}_{\phi}$. The specifics of the fabric defined on the distance manifold remain unchanged over all methods.

7.2.1. SIGNED DISTANCE FIELDS

Representation Collision avoidance can be realized using SDF [122]. In this approach, the environment is discretized into a grid and the distance to the closest obstacle of the environment is assigned to each voxel in the grid. The distance is zero on the obstacle's surface and in its inside and positive outside the obstacle. In [122], the SDF is computed based on lidar data in combination with continuous mapping, but other ways to generate SDFs are possible, see [128], [129] for some manipulation examples. In this work, we address dynamic environments, therefore the SDF changes over time, see Fig. 7.2 for a two-dimensional example.



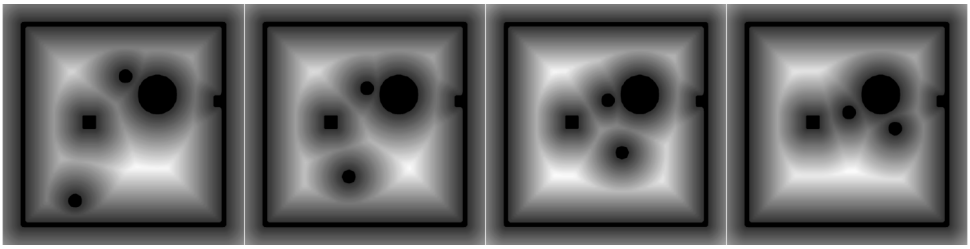


Figure 7.2: Changing Signed Distance Field in 2D.

Integration According to Eq. (2.8), the integration of collision avoidance requires the computation of the gradient $\boldsymbol{J}_\phi$. When the environment is represented as a SDF, the gradient cannot be computed analytically, as it is possible for simple geometric shapes in [27], [110]. To overcome this limitation, we propose to use the numerical gradient as an approximation instead. The SDF is evaluated based on the forward kinematics of the collision link, so $\phi_{sdf}(\text{fk})$ is a function of $\boldsymbol{q}$. As $\partial_{\boldsymbol{q}}\phi_{sdf}$ is not analytically accessible due to the numerical nature of SDFs, we apply the chain rule to obtain

$$\boldsymbol{J}_{\phi,sdf} = \frac{\partial \phi_{sdf}}{\partial \boldsymbol{q}} = \frac{\partial \phi_{sdf}}{\partial \text{fk}} \frac{\partial \text{fk}}{\partial \boldsymbol{q}}.$$

The second term $\partial_{\boldsymbol{q}}\text{fk}$ is the gradient of the forward kinematics, which can be computed analytically. The first part is the gradient of the SDF which can be approximated using finite differences:

$$\left(\frac{\partial \phi_{sdf}}{\partial \text{fk}}\right)_i \approx \frac{\phi_{sdf}(\text{fk} + \Delta_i e_i) - \phi_{sdf}(\text{fk} - \Delta_i e_i)}{2\Delta_i},$$

where Δ is the resolution in the i-th dimension. For the integration into fabrics, the value $\phi_{sdf}(\text{fk})$ and the gradient $\partial_{\text{fk}}\phi_{sdf}$ must be computed at runtime. In contrast, the analytical component $\partial_{\boldsymbol{q}}\text{fk}$ can be precomputed symbolically. Similar to [27], we omit the curvature term $\dot{\boldsymbol{J}}_\phi$ in the pullback operation. When working with multi-link robots, such as manipulators, different manifolds are created for the different collision links. Note, that the same SDF can be used for all collision links.

7.2.2. FREE SPACE DECOMPOSITION

Representation Popular in recent literature [69], [74], [123] is to decompose the environment, with all its obstacles, including dynamic obstacles, into a set of FSD. Then, the workspace is reduced or locally approximated by, typically one, convex regions. The free space is defined by a set of half-planes, each defined by a normal vector $\boldsymbol{n_P}$ and a constant $c_{\boldsymbol{P}}$, see Fig. 7.3. In the following, we describe how the free space decomposition can be computed given a $\mathcal{P}$ of the environment.

The method used in this work is inspired by [123]. We define the $\boldsymbol{x}_{\text{seed}}$, the point in space from which the FSD is computed, as the position of the collision link in question, see Fig. 7.3. Then, $\mathcal{P}$ is sorted according to the euclidean distance to the $\boldsymbol{x}_{\text{seed}}$. Starting with the closest point, a plane $\boldsymbol{P}$ defined by $\boldsymbol{n_P} = \boldsymbol{p} - \boldsymbol{x}_{\text{seed}}$ and $c_{\boldsymbol{P}} = \boldsymbol{p}$ for every $\boldsymbol{p} \in \mathcal{P}$. To

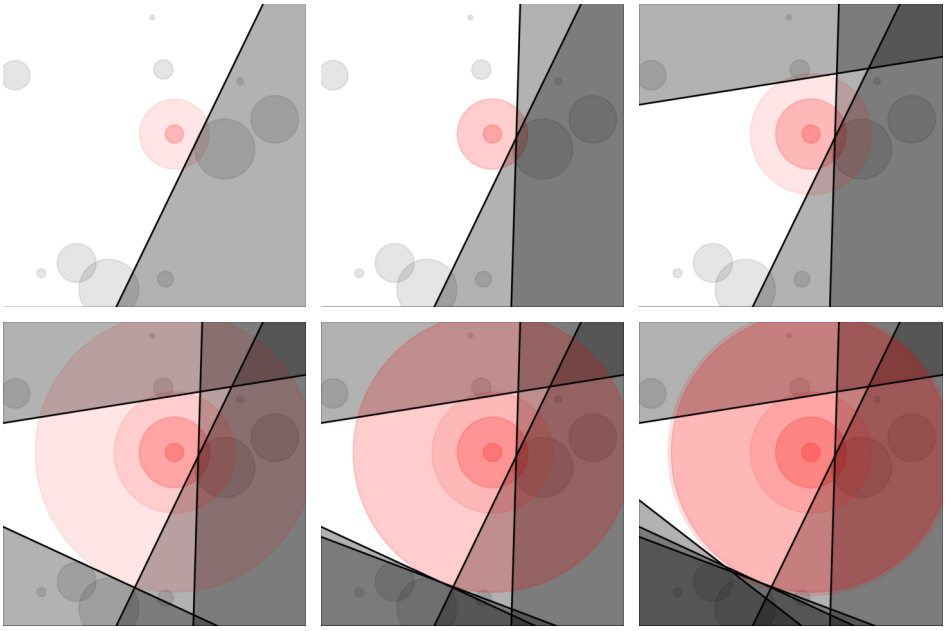


Figure 7.3: Iterative Free Space Decomposition in 2D. The $\boldsymbol{x}_{\text{seed}}$ is shown in red, obstacles in light gray, each $\boldsymbol{P}$ by black plane and the block workspace is shaded in gray. The resulting FSD is the remaining white space.

further speedup the process, every $\boldsymbol{p}$ in $\mathcal{P}$ that is 'behind' an existing $\boldsymbol{P}$ is removed from $\mathcal{P}$. The method results in set $\mathcal{S}_{\text{planes}} = \{\boldsymbol{P}_i, i \in [0, n]\}$, representing the free space around $\boldsymbol{x}_{\text{seed}}$. The algorithm is visualized in Fig. 7.3 for a 2D case.

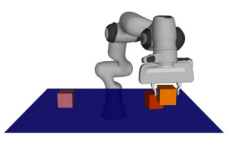
Integration To integrate FSD into the framework of fabrics, we define a differentiable mapping from configuration space $\mathcal{Q}$ to the distance manifold between robot and each constrained plane $\boldsymbol{P}$. Given the fk of a collision link on the robot and the radius of the collision link, the distance to the plane $\boldsymbol{P}$ defined by its normal $\boldsymbol{n_P}$ and the constant $c_{\boldsymbol{P}}$, the distance is computed as:

$$\phi(\boldsymbol{P}, \text{fk}, r_{\boldsymbol{L}}) = \frac{\boldsymbol{n_P}^T \text{fk} + c_{\boldsymbol{P}}}{\|\boldsymbol{n_P}\|} - r_{\boldsymbol{L}}. \tag{7.1}$$

In contrast to SDFs, the gradients, $\boldsymbol{J}_\phi$ and $\dot{\boldsymbol{J}}_\phi$, can be computed analytically as a function of $\boldsymbol{q}$, $\boldsymbol{n_P}$ and $c_{\boldsymbol{P}}$.

7.2.3. RAW SENSOR DATA

Representation As pointclouds generated with either cameras or lidars are usually very large, the direct usage of the raw data is not tractable for most trajectory generation methods. Direct integration refers to create one spherical obstacle for each data point in your raw sensor data. As fabrics are computationally efficient and can handle large amounts of constraints more easily than methods relying on iterative optimization in each time step[110], direct integration becomes feasible. This is also based on the findings by [125] where raw



sensor data was used for drone flying with RMPs. This allows to reduce the amount of preprocessing, limiting the amount of uncertainty and room for error in this step. Here, we explain how raw lidar sensor data can be utilized with fabrics. The same approach can be used to directly integrate pointclouds or occupancy grids.

Integration We assume that a sensor outputs a set of n_{points} points $\mathcal{P}$ in the robot's workspace. We integrate this data directly into fabrics by placing a virtual, spherical obstacle at each $\boldsymbol{p} \in \mathcal{P}$. The map is then defined as

$$\phi_{\text{raw}}(\boldsymbol{p}) = \|\boldsymbol{p} - \text{fk}\| - r_{\boldsymbol{L}} - r_{\text{resolution}},$$

The radius $r_{\text{resolution}}$ of these obstacles must be chosen to reflect the resolution of the point cloud.

7.3. EXPERIMENTAL RESULTS

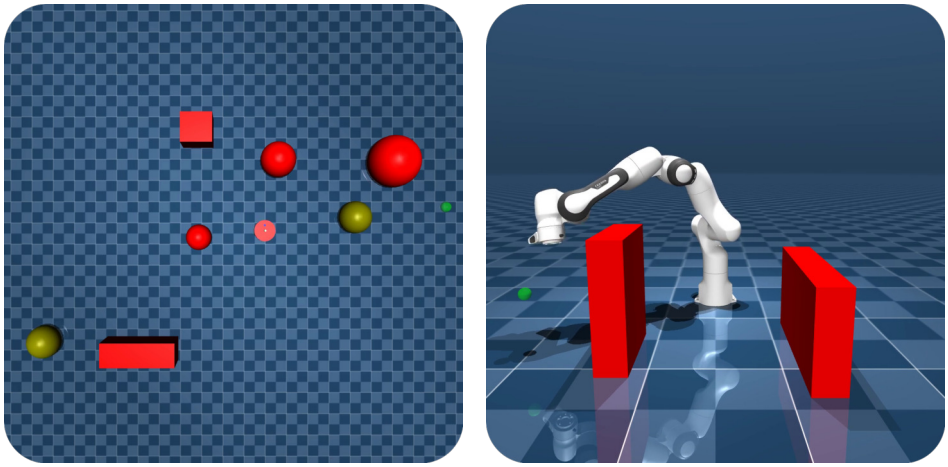
In this section, we explain our implementation of the presented methods and present quantitative comparisons for three simulation environments, namely a holonomic ground robot without and with moving obstacles, see Fig. 7.4a, a robotic manipulator, see Fig. 7.4b. Unless stated otherwise, we evaluate 50 cases with different noise levels σ. The noisy signal is generated using a Gaussian distribution centered around the unoisy sensor data. Specifically, as we use point clouds as inputs to all presented methods, we have $\boldsymbol{n_P}$ 3D points to form a $\mathcal{P}$. The zero-mean, white noise with variance σ is added to each point to form the noisy $\mathcal{P}_\sigma$. This noisy sensor data is then used in the different methods, for computing the FSD, the SDF, and no further processing is done for the raw sensor data.

The performance is measured in terms of success, solver time and execution time to reach the goal. Importantly, the solver time reported in this work does not include the computation of the environment representation. For SDF however, we included the computation of the numerical gradient because it is fabrics-specific. Lastly, we evaluate the methods in the real world using a manipulator, see Fig. 7.12, and a holonomic ground robot, see Fig. 7.5.

Details on implementation The implementation used in this work uses symbolic precomputation of fabrics. In this symbolic interpretation, the composition of the different behaviors is performed before runtime in a symbolic way, see Fig. 7.6. The implementation is identical to the one used in [130]. The code can be found at Geometric Fabrics. The simulation environment as well as the algorithm for computing the FSD, SDF and the raw lidar data can be found as part of urdfenvs. For the real world experiments, we used a ROS bridge and used the same implementation to process the point clouds, generated by either a Velodyne VLP-16 mounted on the robot, see Fig. 7.5, or by an occupancy grid build using the octomap package [100].

7.3.1. GROUND ROBOT

When comparing explicit environment representations with the proposed techniques using noise-free sensor data ($\sigma = 0.0$), SDFs demonstrate the highest success rate. This is likely due to the *guidance* provided by the SDF's gradient information. Interestingly, the success



(a) Ground robot case (b) Robotic manipulator case

Figure 7.4: Simulation cases for the ground robot and the robotic manipulator.

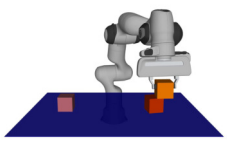
rate for an explicit environment representation increases with the noise level. Potentially, the noisy sensor data is able to push the robot out of local minima, which were reported to be a problem for fabrics in [110]. The more implicit representations suffer from the noise increase, most dramatically for the SDF representation, which becomes unusable around a noise level of $\sigma = 0.1$. When exposing all methods to a dynamic environment (two moving obstacles), the explicit representation has degrading performance, e.g. without noise 42/50 (static) to 38/50 (dynamic), see Fig. 7.8. The implicit representations show a similar success rate across static and dynamic environments. Implicit representations are hardly effected by the dynamic character. This confirms our hypothesis that implicit representations are a good approach in human-shared, changing environments. Moreover, this finding is in line with the findings in [110] on the inability for the explicit representation to avoid moving obstacles.

The goal reaching times remain similar across all approaches, which aligns with expectations as the tuning is based on this criterion, as shown in Fig. 7.9. However, SDF and raw sensor data have some outliers in the time to reach the goal, which is likely due local minima or over-conservative behavior in some cases. While solver computation times are low for all methods (between 0.5 ms and 2.0 ms), utilizing SDF shows the highest computational costs, see Fig. 7.9. Such low solver times allow the usage in real-time applications where high reactivity is required.

7.3.2. ROBOTIC MANIPULATOR

For the experiments with the robotic manipulator, the environment configuration was kept similar to the experiments with the same arm in [65]. The randomized experiments were obtained by randomly selecting goal locations.

High success rates are achieved for all methods in static, noise-free environments, as shown



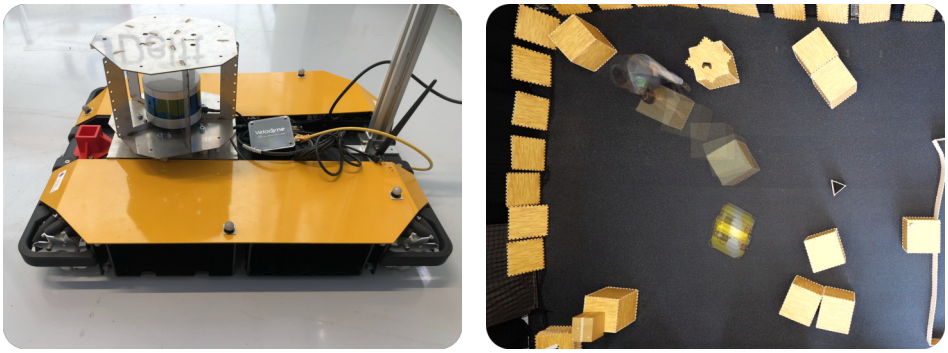


Figure 7.5: Real ground robot used to run the experiments. For the real-world experiment, we used a Clearpath Dingo with a Velodyne VLP16 mounted on top (left). The real-world experiment shows the Dingo navigating through an environment where a human throws in obstacles (right).

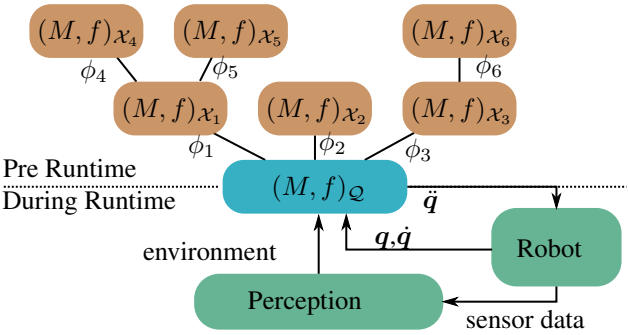


Figure 7.6: Composition of symbolic fabrics and runtime loop.

in Fig. 7.10. As the noise level increases, success rates decrease for all methods. Unlike implicit representations, an explicit environment representation does not suffer from collisions as sensor noise increases, but leads to higher limit-violation rates. Limit-violations are usually caused by a negative $\boldsymbol{x}$ value in the collision spec, leading to infinitely high repulsive accelerations. This is a known issue for fabrics and can be mitigated by using a soft-max function to limit the repulsive acceleration. However, this was not done in this work to keep the comparison fair. However, up to a noise level of $\sigma = 0.02$, success rates are still between 60%(SDF) and 90% (raw sensor data). In terms of execution time, all methods perform similarly, as expected from the tuning process. Solver times are highest for SDF ($\approx$ 25.0 ms) and below 10 ms for the other methods, see Fig. 7.11b. The significantly higher solver times for SDF are due to the computation of the numerical gradient, which is fabrics-specific and thus included in the solver time. This makes all methods suitable for real-time applications.

7.3.3. Real-World Experiments

We tested the methods presented in this paper in the real-world using a Clearpath Dingo and a Franka Emika Panda.

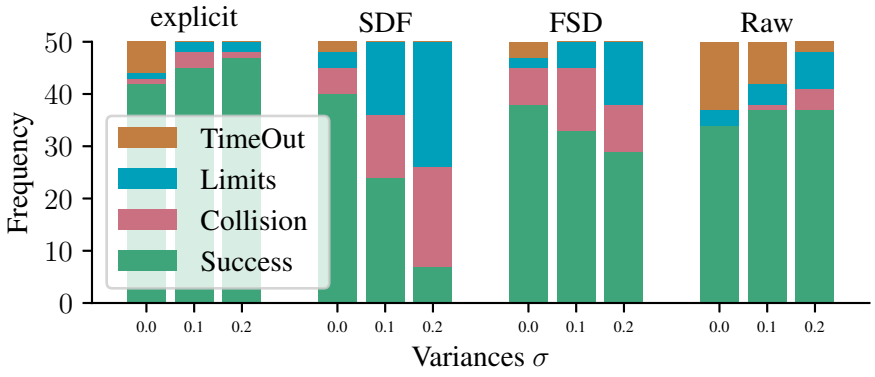


Figure 7.7: Success rates for ground robot in **static** environments for different noise level on sensor inputs.

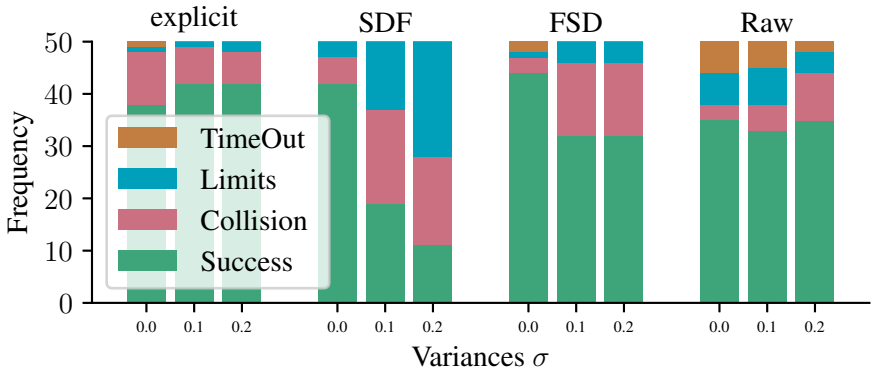
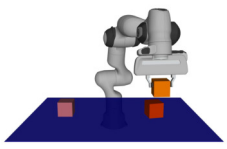


Figure 7.8: Success rates for ground robot in **dynamic** environments for different noise level on sensor inputs.

Dingo For the Clearpath Dingo, we used a Velodyne VLP16 to generate the point cloud data in the ground plane, effectively discarding information for higher z-values. We use a resolution of 1 ray/degree. The method is wrapped into a ros-node where new control actions are commanded at 40Hz. We test the methods in an arbitrary setup environment where obstacles are placed and thrown in front of the robot by a human, see Fig. 7.5. For detailed understanding of the setup, we refer to the accompanied video material. The robot is able to quickly adapt to the obstructions and safely navigates the environment. However, similar to the findings in simulation, when exposed to local minima the robot is not able to escape. This is due to the fact that all methods presented in this paper are highly reactive, exhibiting no time-horizon planning into the future. This is well in line with existing literature on the framework of fabrics and emphasizes the ideal usage of fabrics as a safe medium on which attractor policies can act [131].

Panda For the Franka Emika Panda, we used the octomap package to generate the occupancy grid. The method is wrapped into a ros-node where new control actions are com-



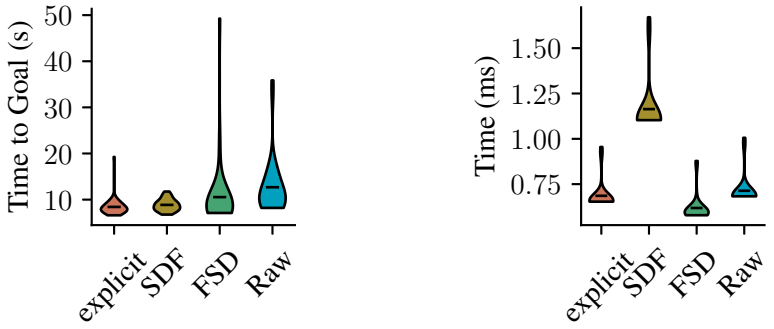


Figure 7.9: Evaluation metrics, goal reaching (left) and solvertimes (right), for the ground robot in simulation.

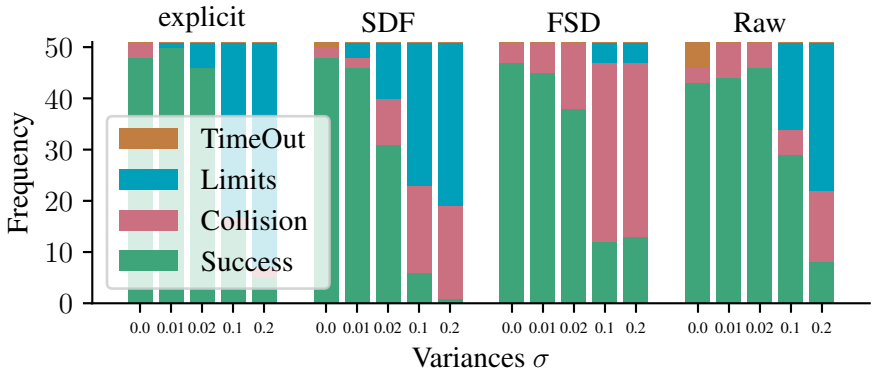
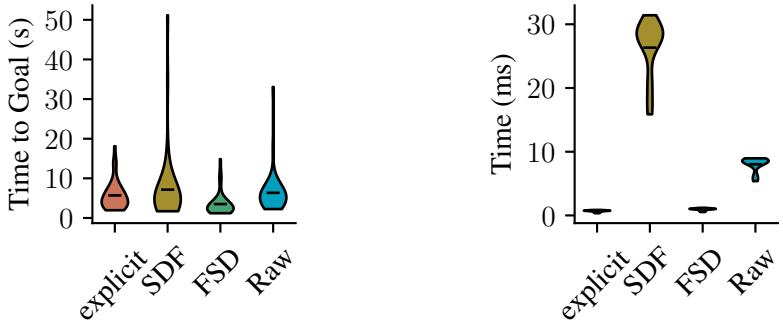


Figure 7.10: Success rates for robotic manipulator in static environments for different noise level on sensor inputs.

manded at 40Hz. We used three collision links on the robots for collision avoidance, see Fig. 7.12 for the realization of FSD. The experiments reveal that implicit environment representations allow for collision-free motion of robotics arms without the need for a complex perception pipeline.

7.4. Conclusion

This paper introduces several approaches to surpass explicit environment representations, employing three distinct implicit representations within the fabrics framework. The study demonstrates that these techniques notably reduce demands and constraints on perception pipelines, while maintaining a low computational load on the planner. Consequently, the proposed methods enable the application of analogous strategies in both dynamic and static environments. The outcomes underline the successful integration of numerical gradients, frequently accessible in trained models, into the symbolic implementation of fabrics detailed in [110]. Additionally, the real-world experiments show that the methods can be transferred physical robots. Future endeavors should concentrate on incorporating more



(a) Goal reaching time

(b) Solvertimes

Figure 7.11: Evaluation metrics for robot manipulator in simulation.

implicit robot representations, e.g. explicit or implicit signed distance fields as outlined in [128], [129], into this framework. While this work refrains from delving into dynamic environmental representations, an exploration of the potential synergies between the implicit representations introduced here and the findings from [110] is a promising avenue for investigation.

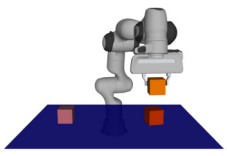


Figure 7.12: Real robotic manipulator used to run the experiments. A Franka Emika Panda is confronted with an unknown shelf environment in a supermarket. Arrows indicate the normals of the planes obtained by the FSD. Different colors indicate different links on the robotic arm.

8

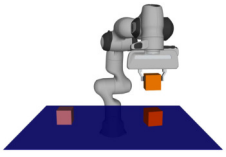
Demonstrating Adaptive Mobile Manipulation in Retail Environments

In the previous chapters on Optimization Fabrics (fabrics), we have suggested a generalization of the theory in the context of dynamic environments, and proposed a symbolic implementation to simplify automated parameter tuning. This chapter highlights that the theory on fabrics is not only theoretically powerful, but also practically realizable. For automated order picking in retail environments, we deploy fabrics as the main Trajectory Generation (TG) method. Insights from visual servoing and learning-from-demonstration are integrated to further highlight the general character of the theory.

This chapter is a verbatim copy of the peer-reviewed publication:

- ☞ **M. Spahn**, C. Pezzato, C. Salmi, R. Dekker, C. Wang, C. Pek, J. Kober, J. Alonso-Mora, C. Hernandez Corbato and M. Wisse. *"Demonstrating Adaptive Mobile Manipulation in Retail Environments"*. Robotics: Science and Systems, 2024.

Statement of contributions: M. Spahn, C. Pezzato and C. Salmi were responsible for the software development and system integration. M. Spahn developed the trajectory generation method, C. Pezzato developed the reactive decision making method, and R. Dekker developed the perception system. The digital twin was implemented by C. Wang. C. Pek, J. Kober, J. Alonso-Mora, C. Hernandez Corbato, and M. Wisse positioned the work in the context of retail environments and provided infrastructure and guidance in all implemantation phases. M. Spahn led the writing process. All authors contributed to the final manuscript.



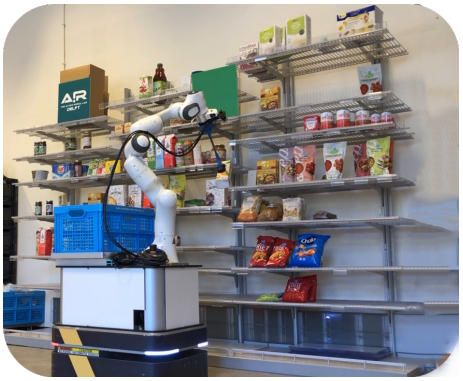


Figure 8.1: We validated our mobile manipulator in our AIRLab lab environment (shown here) and a realistic supermarket environment of a large Dutch retailer.

8.1. INTRODUCTION

Ageing has started to impact the labour markets profoundly, and robotic labour shortage relief is becoming a necessity in all industries [2], [3]. Yet, there are still surprisingly few robots operating outside of structured environments. To bring robots successfully to human-occupied environments, the main challenge still is handling human-instilled disturbances [132], e.g., misplaced products in shelves, newly added products, blocked aisles, or interactions from humans. Such disturbances should be handled adaptively, with little development effort and short response times for natural effective interaction. Focusing on these challenges, this paper demonstrates a combination of two novel methods, one for adaptive decision making and one for rapid motion planning, embedded in a state-of-the-art integrated robot system.

The demonstrator is a mobile manipulator for order picking in realistic supermarkets, see Fig. 8.1. The increase in online shopping induces an equal decrease in store visits. Especially in dense urban areas, these costly but conveniently located stores could have a dual use as distribution centres for flash delivery of online orders [133]. This requires the order picking to occur during opening hours, amongst store customers. We have taken this as our demonstrator scenario, but the methods are generally applicable for any scenario involving human-disturbed mobile manipulation tasks. In such scenarios, we assume that humans can block the robot's path, physically push or hold the manipulator, and move the pick-able items, before, during, or after a pick. Even if items are taken away from the robot's suction gripper, it should recover by picking another item of the same product.

This work presents our mobile manipulation solution in human-shared environments that aims at being adaptive and fault-tolerant. We focus on three main questions that are most relevant when deploying robots in supermarkets: How can we

1. generate *safe* and *robust* trajectories for manipulation?
2. ensure *fault-tolerant* task planning and execution?
3. easily *adapt* the robot to pick new products?

Our solution addresses these questions with specific decisions on the robot's capabilities for decision-making, trajectory generation, and perception. Our contributions are thus an integrated system with adaptive decision-making, fast motion planning and easy-to-use teaching from demonstration.

8.2. RELATED WORK

Robotic mobile manipulation stands as a dynamic and expansive field of research, spurred by diverse potential applications and further fueled by prestigious international competitions, such as DARPA's Robotics Challenge [134], RoboCup@Home [135], the Amazon Picking Challenge [136], and RoboCup@Work [137]. These competitions are tailored to address distinct challenges and performance criteria. Numerous research projects have yielded a significant number of various mobile manipulation platforms. For an exhaustive overview of wheeled mobile manipulation systems and the associated challenges, readers can refer to [138], [139]. In contrast to the aforementioned platforms, we focus on combining commercial off-the-shelf components with little to no modifications to address the specific application.

A supermarket mobile manipulator has been presented in [140] with a special focus on metrics in real-world settings and quantitative field experiments. Similar long-term fetch and carry experiments, yet in different environments, were carried out by Domel et al. [141] in a factory environment and by Stibinger et al. [142] in an outdoor competition to pick up and place simulated construction materials. Instead of relying on a Model Predictive Control formulation, such as [143] for motion planning and control, we deploy a reactive trajectory generation method [61], and a task planning and execution approach that is adaptive in the presence of disturbances. Additionally, we use learning from demonstration to easily teach new products. This approach seems extendable to very different tasks in the long run.

Object picking with manipulators is a well-studied problem. Early approaches rely on engineered components and split detection and grasping into separate tasks. An adapation of STOMP [144] for mobile manipulation was presented in [145]. Here, motions of base and arm are sequenced. Manipulation tasks, including item retrieval, can also be approached from data-driven perspectives. Deep reinforcement learning was used to achieve object picking with a mobile manipulator in non-cluttered environments [146]. In contrast, learning from teleoperation data showed impressive results for dexterous manipulation tasks [147]. The work was extended to mobile manipulation in [148]. While the results are impressive, each task is trained individually, and no safety statements can be made. In contrast, our work relies on engineered components enhanced with learning-from-demonstration techniques to achieve safe and robust mobile manipulation in a supermarket environment.

8.3. SYSTEM OVERVIEW

We briefly introduce the considered supermarket setting and detail our order picking pipeline and system components.

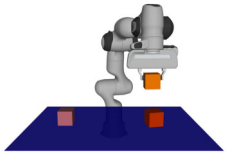


Figure 8.2: Examples of different products considered in this work.

8.3.1. CONSIDERED SUPERMARKET SETTING

Modern supermarkets are characterized by a large range of products, around 100,000 different products per store. Operators usually have access to detailed information of all those products, including mass, geometry, and shelf location in the store. In our demonstration, we assume that the robot can access this database to inform its decision. The large variety of products usually requires specialized grasping strategies per category, e.g., grasping tomatoes is different from grasping a large soft-drink bottle. We focus on a subset of products that can be picked with the suction gripper of our robot (see hardware design in Section 8.3.2), e.g., cans, milk boxes, bottles, or bags of crisps (see Fig. 8.2). The masses of products range from 100g (instant food mixes) to 1.5 kg (soft-drink bottle) and the size are between 10 cm (cans) and 30 cm (soft-drink bottle). We assume that products to pick are visible, accessible from the shelf's front and in the robot's workspace.

For in-store picking, we focus on picking products during opening-hours and favor reliability over execution speed.

8.3.2. HARDWARE

Our mobile manipulator platform is comprised of various hardware components.

Robot The mobile manipulator is composed of two robots, see Fig. 8.3. The moving base is a Clearpath Boxer, differential drive wheeled-robot that can achieve similar speeds to humans while having a relatively small footprint. The robotic arm is a Franka Emika Panda, a serial manipulator with seven degrees of freedom, equipped with torque sensors in every joint that can achieve high accuarcy while being safe to work around, see Fig. 8.1. The attached gripper is a custom 3D-printed suction gripper with two suction cups powered by a industrial vacuum pump.

Perception The base uses a 270 degree Lidar sensor to localize itself and detect dynamic obstacles and humans in the environment. We mounted a Realsense D435 RGBD camera on the wrist link of the arm and use it to detect products and perform visual servoing during picking.

Compute Units We use a total of four compute units to distribute the computational load of individual software components. The first compute unit is the Franka Control Interface controlling the arm. The base's compute unit performs self-localization and collision

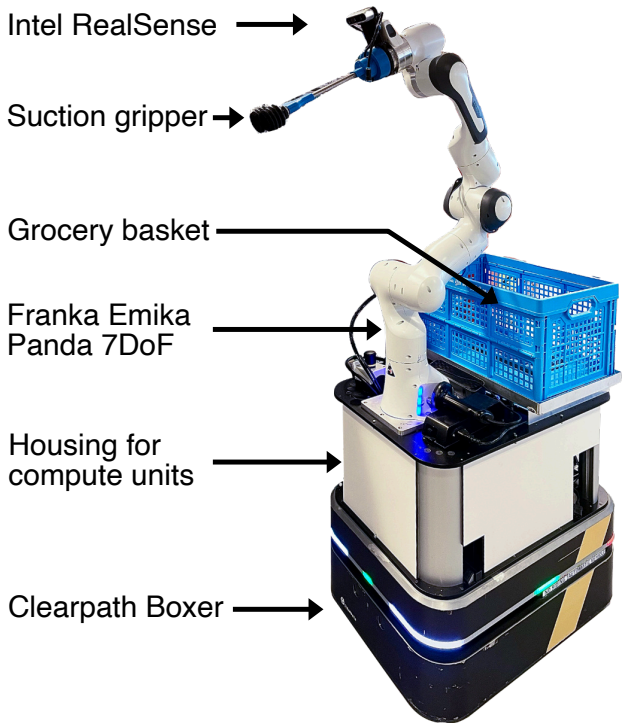
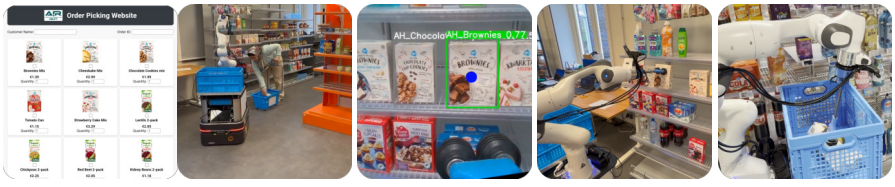


Figure 8.3: Overview of our robot's hardware.

avoidance for the base. The central compute unit is an Intel NUC with an Intel i7 10th generation CPU, running all planning components and the user interface for placing orders. A Dell Alienware laptop mounted on the robot with an RTX 3070 TI GPU runs the perception components. Our two computers are running the Robot Operating System (ROS) and communicate via a network switch.

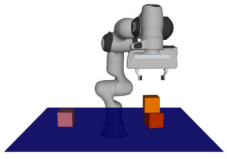
8.3.3. ORDER-PICKING OVERVIEW



(a) Customer order (b) Navigate to shelf (c) Locate item (d) Pick item (e) Place item

Figure 8.4: Overview of ideal flow of symbolic actions to complete an order.

The high-level overview of our order-picking system is illustrated in Fig. 8.4. Customers first place an order via the order placement website. The robot processes the order into a task assignment. For each item, it navigates to the item's shelf, locates it, picks and places



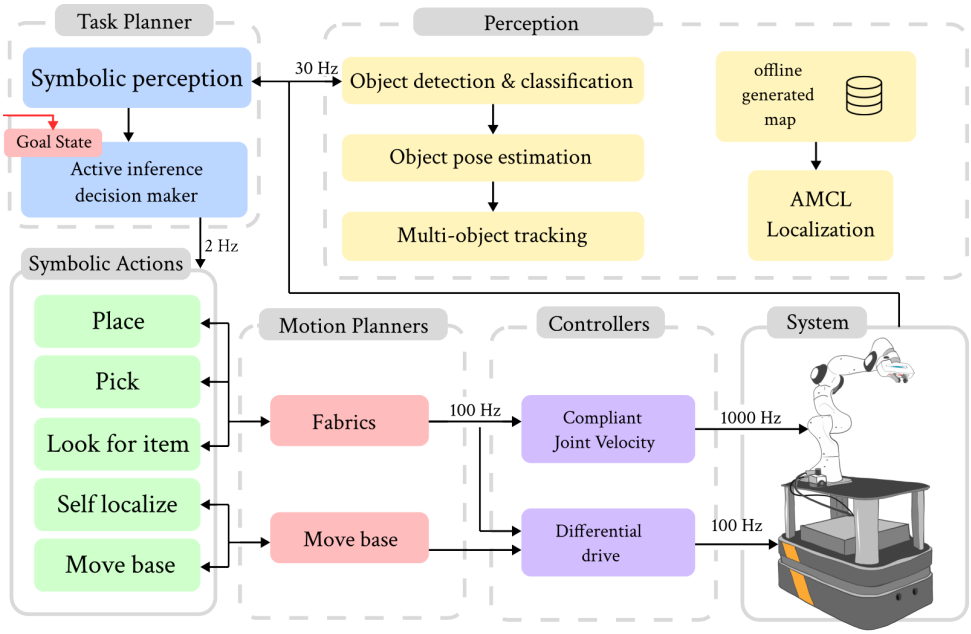


Figure 8.5: Overview of system components.

it in the basket. When the order is completed, the customer can pick up the order from the robot.

8.3.4. System Components

We used the order-picking sequence in Section 8.3.3 to guide our system development, while focusing on adaptiveness to recover from failure and inaccuracies in perception. In the following, we outline the main system components that are visualized in Fig. 8.5 and can be grouped in: (a) task planner, (b) motion planners, (c) low-level controllers, and (d) perception.

After receiving the customer order, the task planner (see Section 8.4) determines the order of picking products by minimizing the robot's travelled distance. The task planner uses a combination of a behavior tree and symbolic state information, such as the robot is holding a product or has arrived at the desired position, with an adaptive inference method to determine the best next symbolic action to execute. We define a *symbolic action* as an elementary robot behavior. Symbolic actions can be as simple as greeting the customer or as complex as picking an item. We use a set of five robot symbolic actions: picking items, placing items, looking for items, localizing the robot, and navigating the robot. Each symbolic action is realized by the motion planning and control components (see Section 8.5). Motion planning is decomposed into path planning and online trajectory optimization for the base and reactive trajectory generation for the arm, augmented with a pseudo-prismatic joint on the base, see Section 8.5. Lastly, the perception component (see Section 8.6) takes care of

item detection and classification and provides item poses to the planning components.

8.4. TASK PLANNING AND EXECUTION

Once the customer submits an order, the task planner creates a plan to collect the items in the order throughout the store and return the shopping basket to the delivery location.

Our decision-making approach to creating these plans and executing them is designed explicitly with failure recovery in mind. It consists of 1) offline plans that leverage the known task structure, and 2) online planning to adapt the action sequence to unforeseen disturbances, following the Active Inference approach in [149].

Active Inference, a neuroscientific paradigm [150], formulates all perception and decision-making in the brain as Bayesian inference, combining prior predictions with novel sensory data. For decision-making, the "prior predictions" are rather *prior preferences*, i.e., desired states, and the probabilistic Bayesian inference is used to determine which symbolic actions have the highest probability of reaching that desired state. In our solution, a sequence of desired states for a task is planned offline and encoded in a Behavior Tree (BT). At runtime, the current desired state (or symbolic goal) is sent to the online active inference planner that computes a symbolic action sequence to transition from the current state to the desired one.

8.4.1. OFFLINE PLANNING

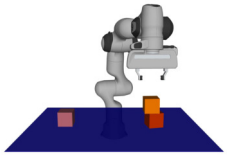
The structure of the task is modelled offline as a plan to be executed using the template BT shown in Fig. 8.6 and expanded by Fig. 8.7, making the robot try to pick every product up to N times, and then deliver the groceries to the delivery location. It specifies an initial welcome to the customer, a “Product Subtree” slot, an active inference node that sets the final task sub-goal of being at the delivery location for the online active inference planner, and a closing message to the customer. For each product, a sub-tree as in Fig. 8.7 is created automatically. Following the order list, every sub-tree for each product is inserted in the overall Behavior Tree (BT) structure from Fig. 8.6, as part of the sequence.

8.4.2. ONLINE PLANNING WITH ACTIVE INFERENCE

The active inference planner (AIP) takes the task sub-goals `isPlaced` and `isAt` as desired item states being placed in the robot’s basket and the robot being at the delivery location, and computes a symbolic action plan based on the robot’s symbolic actions to achieves those states. Our planner uses discrete active inference, which relies on a generative model that contains beliefs about future states and symbolic action plans, where plans that lead to preferred states are more likely. The preferred sequence of symbolic actions is the one with the highest probability of achieving desired states.

Our active inference planner rests on the tuple $(\mathcal{O}, \mathcal{S}, \mathcal{A})$. This is composed of a finite set of observations $\mathcal{O}$, a finite set of symbolic states $\mathcal{S}$, and a finite set of symbolic actions $\mathcal{A}$ that correspond to the robot’s symbolic actions.

The continuous state of the world $x \in \mathcal{X}$ is discretized through a symbolic observer into boolean variables about the relevant states of the world, e.g., item held by the gripper. These discrete observations o are used to build a probabilistic belief about the symbolic current



state, described in Table 8.1.

The AIP computes the posterior distribution over p plans $\boldsymbol{\pi}$ through free-energy minimiza-

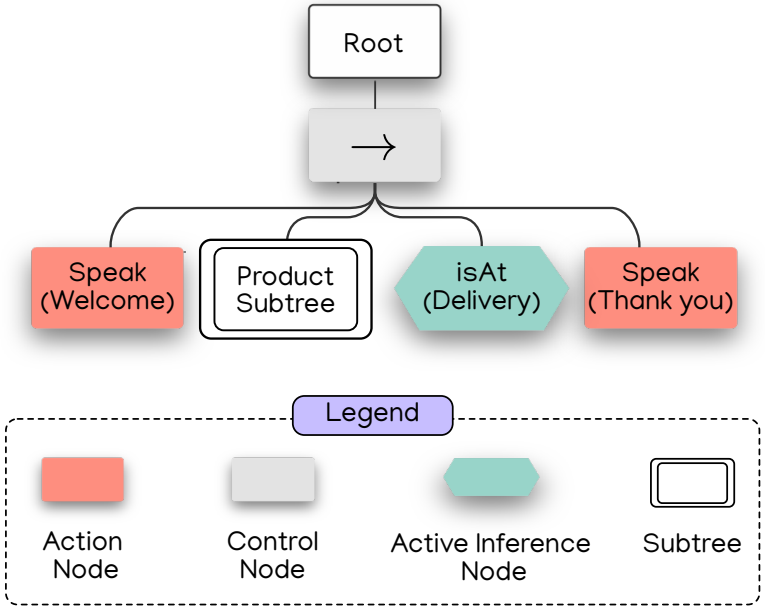


Figure 8.6: Overall BT structure. The symbolic action `Speak` interfaces with the voice module to produce a suitable message for the customers (see Section 8.7).

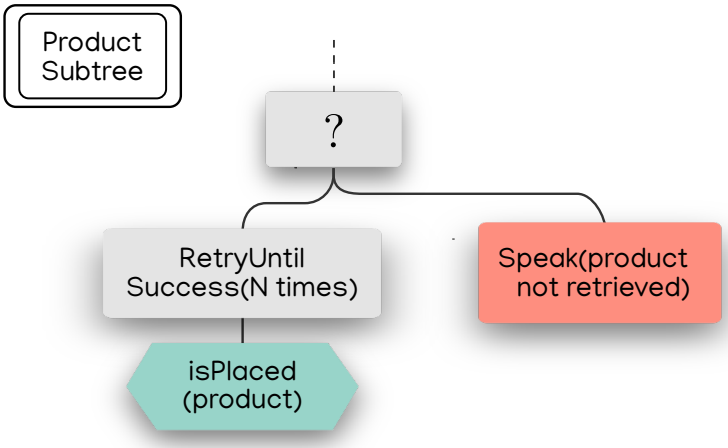


Figure 8.7: Sub-tree structure for placing an item in the basket. The active inference node sets a prior over the state `isPlaced` for a product, triggering the online decision-making. The symbolic action `Speak` produces a voice message to explain the failure in case one happens (see Section 8.7).

tion [149]. The symbolic action to be executed by a robot in the next time step is the first symbolic action of the most likely plan, denoted with $\pi_{\zeta,0}$:

$$\zeta = \max(\underbrace{[\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_p]}_{\boldsymbol{\pi}^\top}), \; a_{\tau=0} = \pi_{\zeta,0}. \tag{8.1}$$

Table 8.1: Notation for belief states. s is the probabilistic belief state and l is the corresponding one-hot encoding

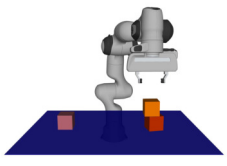
Belief State $\in (0,1)$	**Description**
$s^{(at)}$	Belief about being at the goal location
$s^{(loc)}$	Belief about being self-localized
$s^{(reach)}$	Belief about reachability of an object
$s^{(hold)}$	Belief about holding an object
$s^{(vis)}$	Belief about an object being in sight
$s^{(place)}$	Belief about an object being placed at a location
Boolean State $\in [0,1]$	**Common Name**
$l^{(at)}$	`isAt(goal/obj)`
$l^{(loc)}$	`isLocalized`
$l^{(reach)}$	`isReachable(obj)`
$l^{(hold)}$	`isHolding(obj)`
$l^{(vis)}$	`isVisible(obj)`
$l^{(place)}$	`isPlaced(obj)`

Table 8.2: Notation for symbolic actions

Symbolic Actions	**Preconditions**	**Postconditions**
`selfLoc()`	-	`isLocalized`
`moveTo(goal/obj)`	`isLocalized`	`isAt(goal)/ isReachable(obj)`
`pick(obj)`	`isReachable(obj)` `!isHolding` `isVisible(obj)`	`isHolding(obj)`
`place(obj)`	`isHolding`	`isPlaced(obj)` `!isHolding(obj)`
`look(obj)`	-	`isVisible(obj)`

The combination of offline plans modelled as BT's and online planning using active inference facilitates responsive symbolic action selection for long-term tasks within partially observable and dynamic environments, which is particularly crucial in addressing disturbances in retail settings.

This approach offers the advantage of not having to account for every conceivable contingency and recovery behavior within a BT, and at the same time allows for continuous



online planning. This effectively minimizes computational complexity, enabling the development of a robot capable of adhering to predefined routines while also adapting locally to unforeseen events through real-time online planning with active inference.

8.5. TRAJECTORY GENERATION

This section outlines the various approaches we employ for real time trajectory generation (Section 8.5.1, Section 8.5.2). Furthermore, we explain how the symbolic actions introduced in Fig. 8.5, looking for products (Section 8.5.3), pick (Section 8.5.4) and place (Section 8.5.5) are realized and we explain how they use the trajectory generation approaches.

8.5.1. NAVIGATION OF THE BASE

To navigate the mobile base in the store, we employ the ROS MoveBase framework, configured with A* as the global and Timed-Elastic-Bands as the local planner [151]. We record an environment map including static obstacles prior to deployment. Additionally, we manually define keep-out areas to prevent the robot from going into unsafe or crowded areas, such as checkout zones. The local planner uses the environment map and online lidar sensor information for avoiding collisions with static and moving obstacles, such as humans.

8.5.2. MOTION OF THE ARM

To generate the arm motions (and base during picking), we employ fabrics. Fabrics is based on behavior composition, defined as differential equations in manifolds, which enables safe real-time planning at high frequencies [61], [62], [110]. Since fabrics is a local, reactive trajectory generation method, we require global guidance to perform complex symbolic actions, such as product picking or placing. The global guidance for fabrics consists of a sequence of local goals, where we only continue to the next goal if the previous goal has been reached. In Section 8.7.3 we show how this sequence of goals, i.e., reference trajectory, can be obtained by human teaching. In the following, we briefly explain how fabrics works and how to use it.

METHOD

Requirements such as collision avoidance, joint-limit avoidance or self-collision avoidance are referred to in fabrics as behavioral components. Each component is represented as a differential equation of the form $\boldsymbol{M}(\boldsymbol{x}, \dot{\boldsymbol{x}})\ddot{\boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{0}$ on an appropriate manifold $\mathcal{X}$ of the configuration space $\mathcal{Q}$, where $\boldsymbol{M}$ and $\boldsymbol{f}$ are the importance metric and the forcing term respectively, and $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ are the state, e.g., full configuration of the robot or end effector pose, and its derivatives in the manifold $\mathcal{X}$. By respecting simple construction rules, each behavioral component can be ensured to be an optimization fabric, i.e., a dynamic system that is stable by construction. All components can then be combined in the robot's configuration space by applying the operations of *pullback* and *summation*. In the configuration space, we obtain one optimization fabrics of the form $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f} = \boldsymbol{0}$, where $\ddot{\boldsymbol{q}}$ is the second derivative of the configuration and $\boldsymbol{M}$ and $\boldsymbol{f}$ the resulting importance metric and forcing term, respectively. We define goal states of the robotic arm as a set of constraints $\mathcal{S}_c = \{\mathcal{C}_1, \mathcal{C}_i, \mathcal{C}_n\}$ where each constraint $\mathcal{C}$ is defined by the tuple $\mathcal{C} = (\text{fk}_\text{p}, \text{fk}_\text{c}, \boldsymbol{x})$. Here, fk_p is the forward kinematics to the parent link, fk_c the forward kinematics to the child link, and $\boldsymbol{x}$ is the desired position vector. These constraints are implemented in fabrics as

a forcing term with the differential map $\phi_{\text{goal}} = (\text{fk}_\text{c} - \text{fk}_\text{p}) - \boldsymbol{x}$. On the manifold defined by this differential map, we define the forcing potential ψ that can be pulled at forcing our optimization fabric as $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f} + \partial_{\boldsymbol{q}}\psi = 0$. The final policy is then defined as the solution to the damped differential equation as

$$\ddot{\boldsymbol{q}} = -\boldsymbol{M}^{-1}(\boldsymbol{f} + + \boldsymbol{B}\dot{\boldsymbol{q}} + \partial_{\boldsymbol{q}}\psi), \tag{8.2}$$

where $\boldsymbol{B}$ is a positive definite damping matrix. For further details, we refer readers to [110].

The key advantages of fabrics are their fast computation and flexibility to compose behaviours and define the desired goal in a manifold, rather than being fixed to defining a target configuration or end-effector pose. For example, some tasks may require aligning the end-effector to face a specific point, while the actual position along the line is of little importance, see Fig. 8.9. Similarly, some products, like a can, can be grasped from many directions and one may only need to specify a subset of desired grasping poses, e.g., the grasp height and that the grasp should be perpendicular to the vertical axis, but not the specific approach direction.

SAFETY

An important aspect of applications of robotics solutions to human-shared environments is *safety* and failure-free operation during an extended amount of time. For the latter, the most important aspect is that joint-limits and self-collision is avoided at all times, because these can lead to hardware shutdowns. In fabrics, these constraints are achieved by defining a joint-limit avoidance and self-collision avoidance components as described above, making these failures virtually impossible. Safety, however, is more complex to achieve, as it requires an accurate environment model and a reliable prediction of humans and their individual joints. In this work, we instead opted for safety through compliance during arm motion, i.e., the robot is compliant to external forces. This is achieved by tracking the desired acceleration output from Eq. (8.2) with a low-level controller that outputs the torques for the individual joints. Specifically, we use a PI controller that tracks the velocity that is obtained by integrating the desired acceleration. This approach allows ensuring that collisions are non-harming to the robot and its environments.

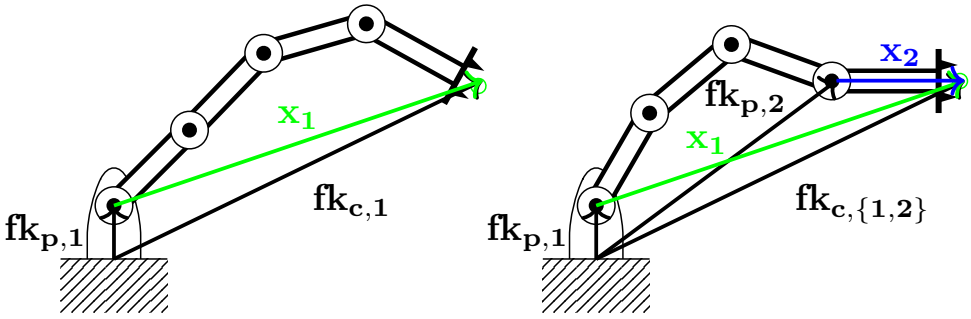


Figure 8.8: Goal constraints for fabrics. In (a), the only constraint is defined by $\mathcal{C}_1 = (\text{fk}_{\text{p},1}, \text{fk}_{\text{c},1}, \boldsymbol{x}_1)$. In (b), a second constraint is added as $\mathcal{C}_2 = (\text{fk}_{\text{p},2}, \text{fk}_{\text{c},2}, \boldsymbol{x}_2)$ to align the end-effector horizontally.

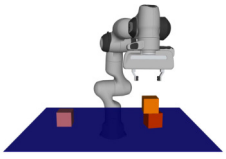


Figure 8.9: Illustration of the orientation constraints for trajectory generation for look-for-product.

USAGE

As an example, a reaching problem, where the end-effector should be at a certain position is defined by $\mathcal{C}_{\text{ee}} = (\text{fk}_0, \text{fk}_{\text{ee}}, \boldsymbol{x}_{\text{ee}})$, where fk_0 is the forward kinematics to the base link of the robot and $\boldsymbol{x}_{\text{ee}}$ the desired position of the end-effector in the base link frame.Fig. 8.8 shows two examples of composing goal states by constraints.

A problem we encountered when picking products with only the arm, is that the arm's workspaceis limited w.r.t. the shelf's size. This in combination with variability in base position or product location, often resulted in the product being out of reach or requiring difficult arm configurations close to joint limits. Therefore, during picking, we augment arm motion by integrating the forward motion of the base as a pseudo-prismatic joint to the kinematic chain. This can be easily done in fabrics by appending the base motion to the state vectors $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$, see Fig. 8.13.

8.5.3. LOOK FOR A PRODUCT

Looking for the product is triggered as soon as the robot base is in front of the shelf that is expected to have the desired product. The camera frame is then located at a position $\boldsymbol{x}_{\text{camera}}$. The camera must be pointed towards the expected product location, defined as $\boldsymbol{x}_{\text{item}}$. We can model this goal as two constraints. First, the camera link should not move from its current location, thus $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{camera}}, \boldsymbol{x}_{\text{camera}})$. Secondly, the camera should face the product location. We compute the ray connecting the camera and product location $\boldsymbol{x}_{\text{ray}} = \boldsymbol{x}_{\text{item}} - \boldsymbol{x}_{\text{camera}}$ to define $\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{\text{ray}})$ that aligns the end-effector with the defined ray $\boldsymbol{x}_{\text{ray}}$, see Fig. 8.9.

8.5.4. PICKING OF A PRODUCT

To define a goal for picking products we use a combination of three constraints. First, the position of the vacuum cup is determined by the position of the product, thus $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{suction-cup}}, \boldsymbol{x}_{\text{suction}}, w_{\text{suction}})$. Secondly, we limit ourselves to picking products from the shelf, thus constraining the end-effector to be perpendicular to the shelf by defining

$\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{\text{flange,end-effector}})$. Lastly, our gripper is composed of two suction cups, which, depending on the product should align with a specific angle for executing the most reliable grasp. The desired alignment defines our third constraint for the picking as $\mathcal{C}_3 = (\text{fk}_{\text{suction1}}, \text{fk}_{\text{suction2}}, \boldsymbol{x}_{\text{alignment}})$. Although it is possible to program picking, including approach and retreat, as a sequence of this set of constraints, it is difficult to capture all the nuances of picking in the code. Therefore, we make use of a human operator to teach the robot the best trajectory to reliably pick specific products, following the learning-from-demonstration paradigm [152]. Teaching has proven an effective way to generate sequences of the previously mentioned constraints thus encoding the human understanding of the picking problem into recorded trajectories. We explain the process of recording and playing back trajectories in detail in Section 8.7.3.

8.5.5. PLACING OF A PRODUCT

Placing the product consists of four phases. The robot navigates to a configuration to its right or to its left depending on whether it was a right-sided pick or a left-sided pick. Then, it moves above the crate facing downwards. This is defined by two constraints, $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{0,end-effector}, 1)$ and $\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{flange,end-effector})$. The product is placed by moving the arm downwards until an external force, from touching the crate's bottom or an already placed product, is detected. This triggers the gripper release and goal change to the homing position ready for the next product.

8.6. PERCEPTION

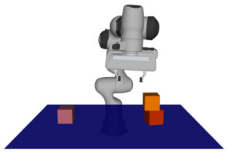
The three perception components are shown in Fig. 8.5.

- **Object detection and classification:** Generates 2D object proposals and classifies them in a binary way based on a provided target class, resulting in the proposals being classified as either target class or not. Object detection and classification are realized using two different models. Both models are fine-tuned on a supermarket dataset, but do not require retraining to add new products, as we will explain in the following.

- **Object pose estimation:** Uses 2D object proposals in combination with a depth image to convert them to 3D. To estimate the orientation around the z-axis, we use a plane-fit of the pointcloud frustum. That first order approximation of the surface of the products front has proven to be a reliable approach, even for non-planar surfaces, see Fig. 8.2.

- **Multi-object tracking:** To track the objects over time, we use a set of Kalman Filters, one per object. The object proposals are assigned to Kalman Filter tracks using the Hungarian Algorithm [153], [154].

Because a supermarket has a large and often changing set of products, the main requirement for our perception pipeline is that it should be easily adaptable. This observation from retail environments led us to the constraint on the perception pipeline that it should not require retraining when new products are added. Such a constraint can be addressed using, so-called, *few-shot* models. In the following, we describe the details of our object detection and classification method.



The method utilizes YOLO for object detection, relying on product details for object classification. Additionally, our system allows adding products using a single or few images.

This dynamic addition of new products is possible trough a few-shot model we dub ProtoProductNet. This model is based on ProtoNet [155]. ProtoNet matches query images to target classes by their distance in feature space. For each target class a prototype is constructed that is essentially the mean of the features of a number of example images of this class. By matching query images to target prototypes, ProtoNet essentially learns to encode features that classify similarity between query images and target classes. Because this model picks random query- and target classes from a dataset for every iteration, ProtoNet learns a general feature extraction strategy that is invariant of the actual class. This is important for adding new products, as classifying new products is as easy as providing the model with new target images.

The exact implementation of ProtoNet we use is based on P>M>F [156]. P>M>F shows that in few-shot learning pre-training (P) is more important than meta-training (M), which is in turn more important than fine-tuning (F). For the best results the authors of P>M>F suggest using ProtoNet with a Vision Transformer pre-trained with DINO [157] as the feature extractor and meta-training it with a small learning rate.

However like most few-shot classification models, ProtoNet assumes that query images can only be one target class. Not only would comparing a query image to all supermarket products increase inference time, attributing it to a likeliest product is unsafe. If our product detector misidentifies a human as a product, the classifier must correctly recognize that and not classify it as the most likely product.

ProtoProductNet makes exactly this possible. It uses a ViT pre-trained with DINO to extract image features, and predicts if those features are part of a target prototype based on their cosine distance. ProtoProductNet then passes this cosine distance through a linear layer combined with a sigmoid function to translate it to a confidence score. If query images have a confidence < 0.5, they are considered not the target class. Using a sigmoid function however leads to a loss of relational information between classes. In contrast to ProtoNet, that predicts only the most likely target class among a set of target classes with a softmax function, a sigmoid predictor only uses the cosine distance per class to make predictions. As this mechanic is an important reason why ProtoNet works so well, ProtoProductNet will also be allowed to choose the likeliest from a number of prototypes. This means that next to a target prototype, a number of helper prototypes will be chosen. When classes are likelier to be a helper class then the target class, they are considered to be not the target class. As classes that are close together in feature space are harder to distinguish, it makes only sense to choose helper prototypes that are close to the target prototype.

8.7. Interaction and Teaching Capabilities

To quickly adapt our robot to new store environments and products, we created four interfaces for operators: a digital twin for remote monitoring and control, adding product classes to the perception, trajectory teaching mode, and audio explanations of the robot's symbolic actions.

Figure 8.10: Overview of the remote monitoring and control system. a) Laptop-based remote interface for monitoring and control system; b) Visualization of the robot within the actual retail environment; c) Tablet interface for on-the-go monitoring and task programming.

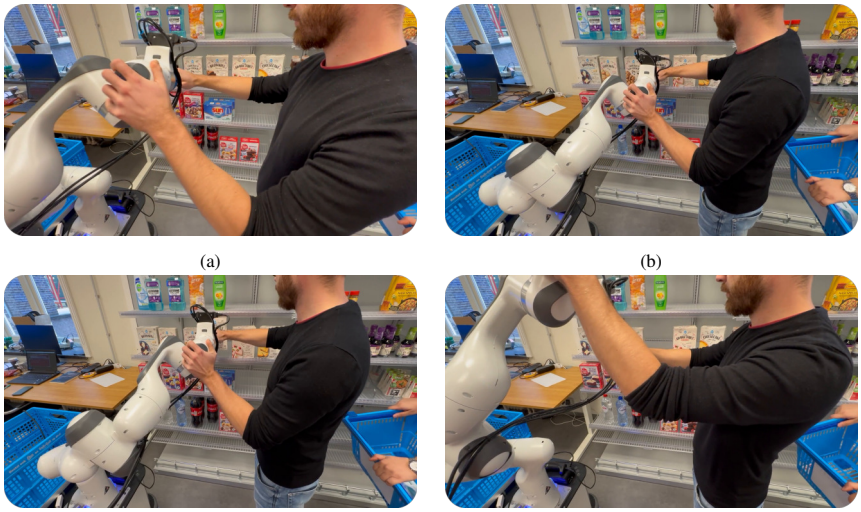


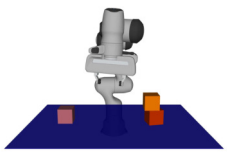
Figure 8.11: The human operator can easily 'teach' the robot a new picking strategy by moving the arm, thus passing implicit knowledge to the robot.

8.7.1. DIGITAL TWIN FOR REMOTE MONITORING AND CONTROL

Herein, we introduce a digital twin mechanism to support remote monitoring and control of a mobile manipulator in a retail setting, as shown in Fig. 8.10. By scanning the environment in three dimensions, we construct a virtual model that accurately represents the workspace. The robot, when operational in a supermarket, is connected to this digital twin through Wi-Fi or 4G, enabling operators to monitor its status and issue commands remotely. The addition of a tablet interface allows for flexible monitoring and control by on-site staff, who can easily adjust the robot's course or teach it new tasks as needed.

8.7.2. INTERACTIVELY ADDING PRODUCT CLASSES TO PERCEPTION

Section 8.6 explains ProtoProductNet, our adaptation to the state-of-the-art ProtoNet, to make the few-shot learning approach scalable for the supermarket environment. To add new unseen product classes to ProtoProductNet, we developed a custom user interaction for human operators. The interaction contains the following steps 1) the operator uses a barcode scanner, attached to the robot, to scan the new product 2) the operator puts the



product in view in front of the robot's camera 3) a GUI with the view of the camera pops up on the screen and the operator interactively drags a box around the new product. The robot should already start detecting the product, as can be verified by a bounding box appearing on screen. To further enhance the detection the operator can add images from different angles of the product. The cropped images selected by the operator are saved locally and combined with the product's barcode. If the same barcode is encountered in future orders, our perception pipeline will now know how to classify the product accurately, without re-training the network.

8.7.3. TEACHING GRASPING TRAJECTORIES TO THE ROBOT

As outlined in Section 8.5, fabrics require global guidance to effectively execute complex symbolic actions that are essential for some products, see Fig. 8.2. To simplify the process of obtaining this guidance in the form of trajectories, we leverage human expert demonstrations, effectively teaching the robot. We distinguish between two phases for teaching, the *recording* and the *playback*. For both phases, we assume that the product is visible and detected by the robot, such that we can compute a transformation between the root link and the product.

RECORDING

When recording a trajectory, we first reduce the stiffness of the robot to the bare minimum, such that it can easily be pushed around by the human operator. Then, the human operator can activate recording by pressing a button on our tablet interface. From that moment onwards, the state values $\boldsymbol{x}$ for the constraints defined for picking in Section 8.5.4 are recorded, see Fig. 8.11. The state of the gripper, active or non-active, and whether a product is attached to the gripper are also recorded. The generated sequence of constraint values and gripper states is stored as a reference trajectory.

We additionally record the transformation matrix between the root link of our kinematic chain and the product to be picked. That allows us to later generalize the recording to different product poses by applying a rigid body transformation to the trajectory.

PLAYBACK

During trajectory playback, we loop through the recorded goals sequentially, continuing when a desired goal accuracy has been reached. In contrast to the recording part, where we exclude motion of the base, during playback the base motion is activated, see Fig. 8.13. To account for different product poses between recording and playback, we transform the goals on the fly based on the product pose estimate, see Section 8.6, using the following transform:

$$ {}_{\text{item},r}\boldsymbol{T}^{\text{item},p} = \left({}_{\text{base}}\boldsymbol{T}^{\text{item},r}\right)^{-1}_{\text{base}}\boldsymbol{T}^{\text{item},p}, $$

where ${}_{\text{base}}\boldsymbol{T}^{\text{item},r}$ is the transformation matrix between the manipulator's base link and the product during recording and ${}_{\text{base}}\boldsymbol{T}^{\text{item},p}$ is the transformation matrix between the manipulator's base link and the product during playback, see Fig. 8.12. Using this continual feedback we effectively employ a visual servoing [158] approach and are robust against changes in product location during the pick. In addition to fabrics goals, the recording also contains information about the state of the vacuum pump. This state information is replicated during

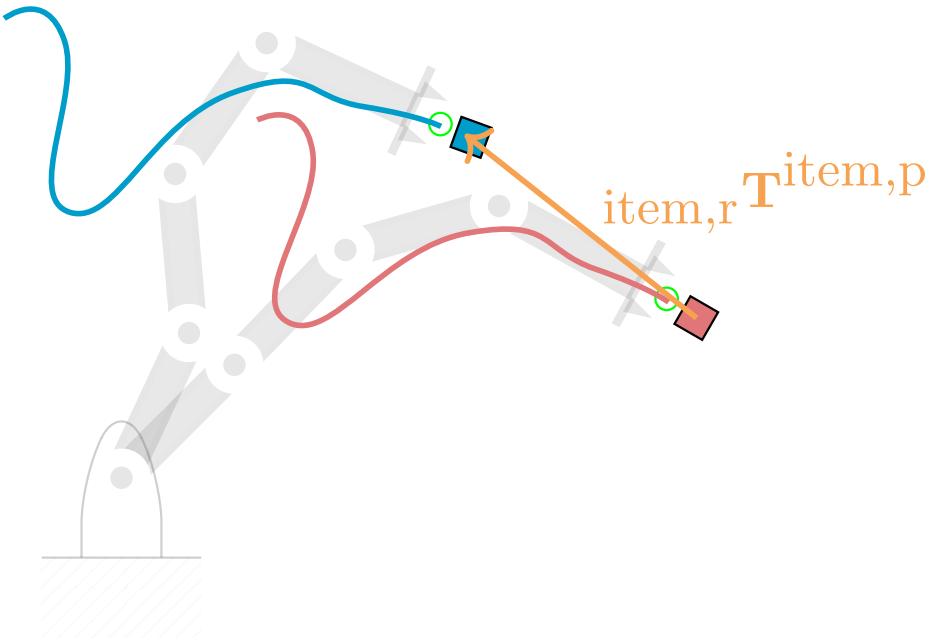
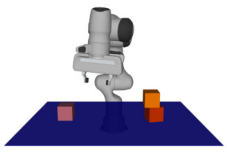


Figure 8.12: To generalize to different item poses, recorded trajectories (red) are transformed based on the the transformation between the item's pose during recording and during playback (orange). The new trajectory (blue) is then tracked using our trajectory generation method.

playback, and used to know when a product should have been attached. In the playback routine for picking we then modify the fabrics goal if a product is not yet attached where it is expected. The goal is modified to effectively push further into the shelf along the z-axis of the nozzle head, until a product is attached, or a maximum threshold is reached.

8.7.4. GENERALIZABILITY

To reduce the number of taught trajectories, we rely on generic trajectories. Our gripper design led to a *horizontal* and a *vertical* pick trajectory, where the two suction cups are either aligned horizontally or vertically. As most considered products have a planar surface, we can use these two trajectories for most products. These trajectories are replaced by product specific trajectories in case of repeated failures. For example, unconventional bottle shapes require modified trajectories. Similar to existing works on learning-from-demonstration [159], we argue that non-experts can, over time, create an increasingly complete trajectory database for all products to further improve performance. Importantly, general trajectories have proven to be sufficient for most of our products. Note that all trajectories are robot agnostic and only gripper specific, so we expect them to be transferable to other robots with similar gripper designs. The generalizability of our approach relies on the transformation of trajectories according to the item's pose. The robot's workspace is a natural limitation, as items placed outside the workspace (i.e. on the lowest or highest shelf or at the back on the shelf) are kinematically unreachable and thus, not resolvable by our teaching approach.



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

Figure 8.13: During playback, fabrics actively use the base's forward motion as a prismatic joint to compensate for misplacement during navigation.

8.7.5. AUDIO FEEDBACK

During the robot's operation, we are interested in providing audio explanations of the robot's actions as feedback to operators, e.g., to monitor the robot and to be notified about failures. We do this by 1) generating compact prompts of the robot's action and state and 2) using Large Language Models (LLMs) to generate short informative explanations that are played via the robot's speaker.

Prompt generation We leverage the structure of the generated BT's and symbolic state information (see Section 8.4) to generate prompts for an LLM. For every sub-tree in the generated BT, we automatically add explanation nodes that generate prompts for symbolic actions and items. An explanation is described by its name a formulated as a verb in the present continuous form, e.g., $a = \texttt{placing}$. The item's name i is taken from the product database, e.g., $i =$ *Whole Milk*. We generate string prompts of the form pr:

$$pr := \textit{action} \;\; a \; i \; c,$$

where $c \in \{\texttt{running}, \texttt{failed}, \texttt{completed}, \texttt{retry}\}$ is the status returned from the sub-tree. An example for a generated prompt is

"*action* `picking Whole Milk retry`".

Explanation generation We generate explanations by prompting an LLM on the fly with our generated prompts. To this end, we provide the LLM with a context describing that the robot is deployed as an order picking robot in a supermarket with five symbolic actions and that the task is to generate a concise explanation of the prompt to operators. During operation of the robot, we simultaneously generate the explanations and play them back via the robot's speaker. For instance, for the prompt "*action* `picking Whole Milk retry`", we generate the explanation: "Oops! It seems like I had a little trouble placing the Whole Milk into my basket. No worries, I'll give it another try and make sure it goes in smoothly this time".

8.8. VALIDATION

To evaluate the performance and how well our robot adapts, we validated our robot in picking customer orders in two realistic environments. Our driving validation questions were:

Table 8.3: Number of attempted, succeeded and recovered symbolic action attempts for picking and placing (the more complex symbolic actions). Note, that a recovery is defined as a successful execution of a symbolic action after a failure.

	AIP-goal		
	attempted	succeeded	recovered
AIRLab	81	50	9
Realistic store	151	90	26
	picking		
	attempted	succeeded	recovered
AIRLab	65	45	9
Realistic store	153	98	25
	placing		
	attempted	succeeded	recovered
AIRLab	46	43	2
Realistic store	91	90	6

1. What is the success rate of our robot?
2. What are causes for failures of the robot?
3. How well did the robot recover from disturbances?

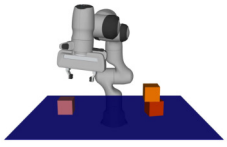
We first describe the two validation environments, followed by summarizing the robot's performance and how it recovered from introduced disturbances such as misplaced items. A video showcasing our robot can be found in the paper's supplementary material.

8.8.1. VALIDATION ENVIRONMENTS

We evaluated our robot in two different, realistic but controlled environments:

AIRLab (see Fig. 8.1) Our *AI for Retail* (AIRLab) environment is a university laboratory at TU Delft that resembles parts of real supermarkets of our Dutch retail partner. AIRLab has OEM shelves and products. We used this environment to develop and validate our system during development.

Realistic store We also validated our robot in a realistic store layout of our Dutch retail partner, used by their development teams for testing before moving into their real stores. For confidentiality, we cannot show this store. The main differences to AIRLab are a larger number of products in the shelves that are also more densely packed, similar to the real stores of our partner. Moreover, the shelves also contained product information tags. We prepared one full day in this store to validate our system, including creating a map, scanning available products, and connecting to the product database.



8.8.2. COMPARISON TO TELEOPERATED SYSTEMS

A direct comparison to a human picker seems of little use as the current stage of the system is not competitive in terms of capabalities and speed to a human picker. Instead, we compare the autonomy of the picking strategy to a teleoperated version of it. The focus on the picking for this lies in its important contribution to the overall success rate and execution time, see Fig. 8.16a. For this study, the robot starts facing the shelf where the product is expected. The teleoperator has access to the camera image from the camera mounted on the end effector. Control is limited to Cartesian movements of the end effector, base forward motion and vacuum activation. Then, the teleoperator has access to the same controls available to the robot in autonomous mode. Therefore, the symbolic actions evaluated in this study are the *look-for-a-product* and *picking-of-a-product*. This limited study was performed for a subset of five different products from the set used in the demo stores. Teleoperation results in similar execution times than the autonomous mode, see Fig. 8.14. This indicates that the execution times is likely limited by the hardware setup, including sensors and actuators, and not by the modules responsible for the autonomous behavior. However, we acknowledge that the teleoperation study is limited in scope and that more capable teleoperation setups, see for example [160], might substantially outperform the autonomous mode.

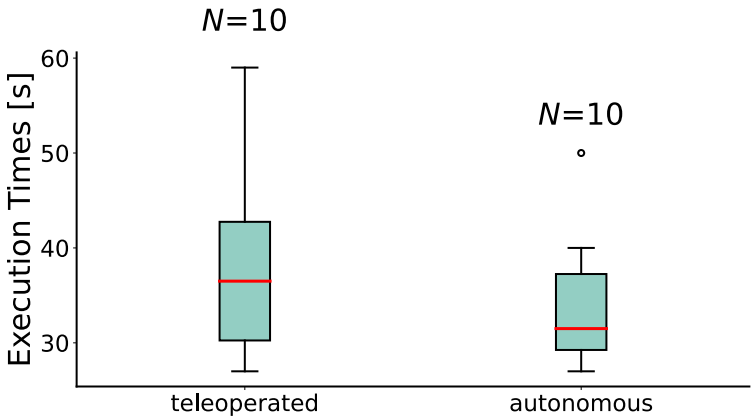


Figure 8.14: Execution times for the combination of *look-for-a-product* and *picking-of-a-product* between teleoperation and autonomous mode.

8.8.3. PERFORMANCE

Performance is evaluated by success rate and execution times. Orders are divided into *success*, i.e. the entire order was successfully collected and returned to the client, *partial success*, i.e., at least one product was not collected and at least one was collected, and *failure*, i.e., no product was collected. This is visualized by the inner ring in Figs. 8.15a and 8.16a. We also inform about the failure reasons and the number of products a successful order contained (outer ring in Figs. 8.15a and 8.16a). For each symbolic action, we report execution times and how many failures were recovered by the adaptive task assignment method, see Table 8.3. As the symbolic action remains active throughout the entire treatment of one product, there lower bound is the sum of the execution times of the other symbolic actions.

8.8.4. SUCCESS RATE AND RECOVERIES

Our evaluation in a lab-like environment reveals that we can achieve a success rate of about 60% for few-items orders, see Fig. 8.15a. Additionally, we observe that most failures are caused during 'picking'. In Table 8.3, we see that recoveries, i.e., a symbolic action failed at least ones before it succeeded, were common. Thus, it shows that decision-making that is able to deal with disturbances is essential in this sort of application. Investigating the execution times, it can be seen that 'picking' is also the symbolic action that takes most time in collecting an item, roughly 50s on average between starting the grasp at its completion, see Fig. 8.15b.

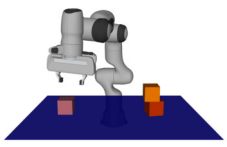
A remarkable property of our system is its reliability and fault tolerance at a very low computational cost. Specifically, apart from the perception, all the components, including the decision-making and the trajectory generation, are running on the Intel NUC with an Intel Core i7-10710U, a low-power CPU that is roughly 80% worse than the compute unit used in [140] according to `www.cpubenchmark.net`. This relies on our multi-level approach to adaptability and recovery from disturbances and runtime uncertainties:

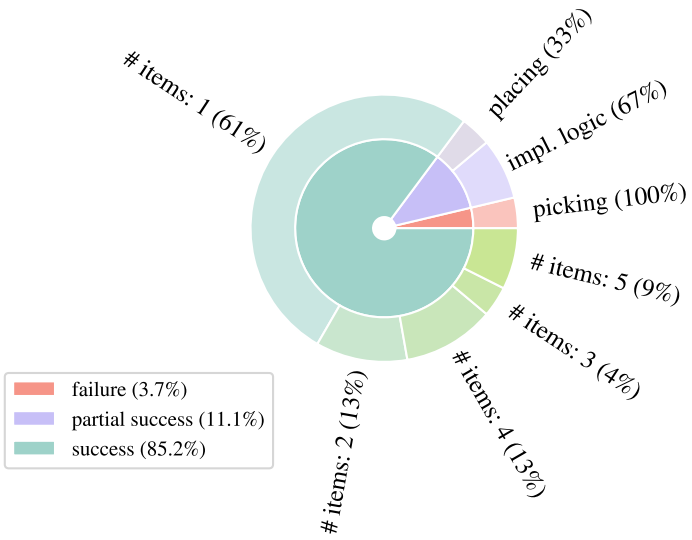
Skill level Our Skills are adaptive to disturbances such as sensor noise, to which our object recognition is robust, or physical disturbances. Examples of the latter are the ability of our compliant arm control to accommodate someone holding it —e.g., if an operator identifies an issue with the item being picked by the robot and wants to take it from the robot— or the visual serving enabled by our object detection and trajectory generation that continuously adapts the motions in case the position of the object changes in the field of view of the robot.

Task execution level If the adaptability of the symbolic actions falls short of accounting for a disturbance, e.g., the operator took the item from the robot. It is now out of its field of view; failing to detect the item, our extremely reactive online planner would generate an alternative sequence of symbolic actions to achieve the desired intermediate subgoal belief state of the item being in sight, resulting in the trajectory generation component smoothly transitioning to a trajectory for the end effector to look for another instance of that item in the shelf. The formulation of the online planning problem in terms of desired states instead of symbolic actions results in a failure recovery behaviour that is easier to scale since there is no need to re-write an entire application-specific logic, which is the case in solutions based on state machines, but one can extend the definition of the planning problem with new states and eventually new symbolic actions if new symbolic actions are developed for the robot.

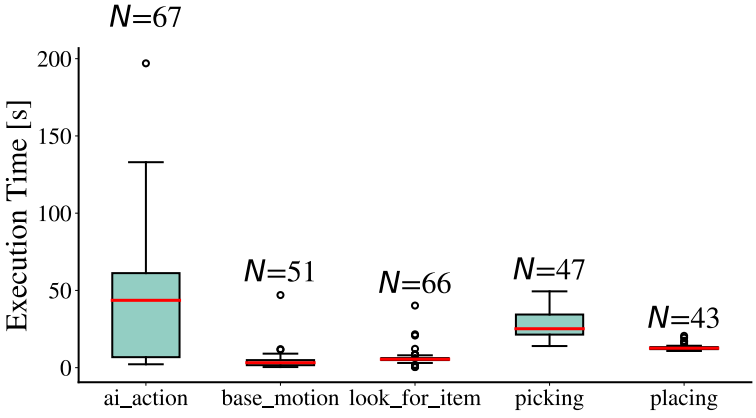
Task plan level The BT structure with pre-defined recoveries retries a subgoal, e.g., getting an item into the basket, up to three times when it fails. This ensures a reasonable trade-off of reliability and performance, e.g., most of the time the second attempt to pick and item was enough. If the third attempt fails, most likely, the item can not be grasped. This heuristic is computationally simple and easily adjusts to new items, e.g., allowing more attempts for incredibly challenging items.

The evaluation in the realistic store environment shows that the system can be deployed to





(a) Success-rate across order sizes and failure causes.



(b) Execution times of symbolic actions in seconds. Note that symbolic actions remain active until the desired state, i.e., product in basket, is reached or a failure occurs. In that case, the symbolic action is re-attempted.

Figure 8.15: Success-rate and action execution times in AIRLab environment. A total of $N = 27$ were performed.

Failure case	Potential causes
Product knocked over during pick	Inaccuracy in product detection or trajectory following, resulting in insufficient vacuum seal
Collision with shelf	Changes in environment due to shelf railing, price tags or discount tags
Product outside reachable space	The arm cannot physically reach the bottom or top shelf
Collision with surrounding products on the shelf	Products are differently positioned than during taught behavior
Vacuum gripper fails to attach	Factors like product size, weight, shape and material can cause vacuum suction to be insufficient

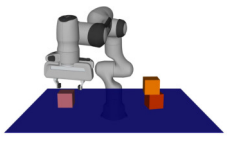
Table 8.4: Qualitatively evaluated list of potential failure cases

a human-shared environment without a major loss of performance, see Fig. 8.16a. This test environment also confirms that most reliability issues are caused by the picking action.

8.9. LESSONS LEARNED AND KEY TAKEAWAYS

Throughout our project, we have gained valuable insights that inform our approach to deploying robotic systems effectively. These lessons, drawn from hands-on experience, highlight key considerations and strategies we believe essential for successfully implementing robotic software solutions.

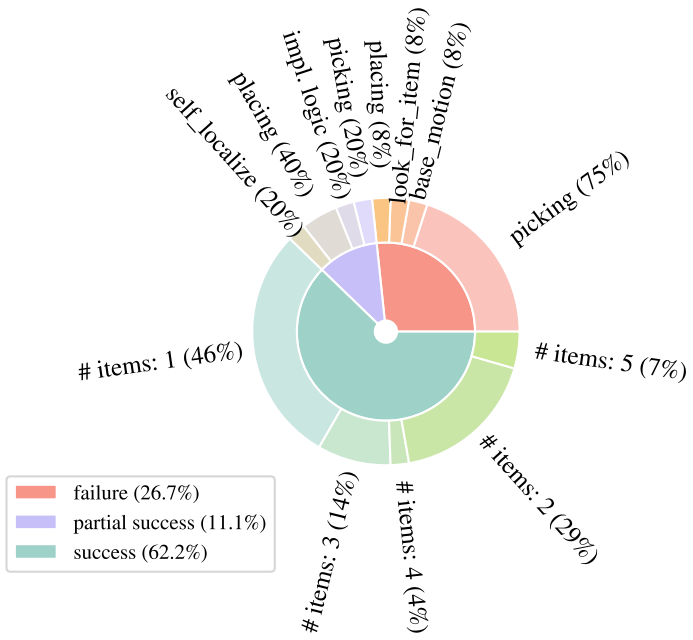
- **Human expert trajectories are an efficient way of encoding grasping strategies**: Through the recording of human expert picking trajectories, we could address a significant portion of collision avoidance challenges and grasping strategies for specific products. This approach effectively allows for the encoding of per product strategies regarding grasp approach, location, and retrieval in a far more streamlined manner compared to traditional hard-coded behaviors. We showed that default trajectories generalize for various similarly shaped products, so that the number of trajectories can be much lower than the number of different products. Although the adaptation of recorded trajectories proved successful, we require more complex methods to mitigate remaining failure cases.
- **Accurate product detection and continuous visual feedback are crucial**: Accurate product detection and pose estimation emerged as critical requirements within the confines of supermarket environments because of the small size of certain products and the lack of clearance. Continuous visual feedback, particularly through visual servoing techniques, played a pivotal role by enabling real-time tracking of products and refining pose estimations as the robotic arm approached the target object.
- **Vacuum grippers may fail with light and small products**: We underestimated the inherent difficulty in effectively picking very light and small products. This challenge highlights the need for alternative gripping mechanisms or specialized approaches tailored to handling such delicate items.



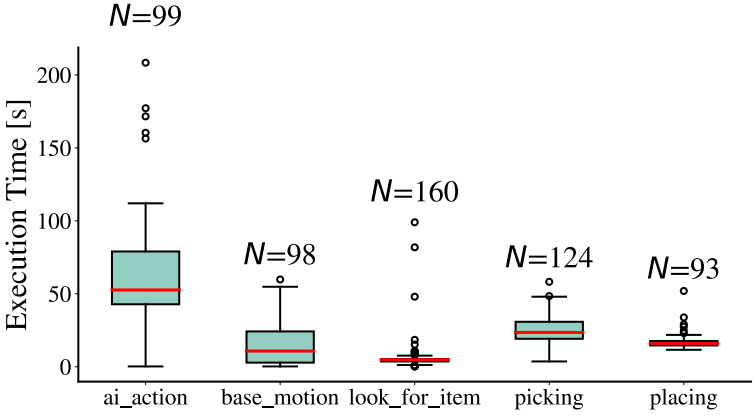
- **Grasping angles heavily influence seal integrity and stability**: We put particular emphasis on selecting optimal grasping angles to ensure both seal integrity and product stability during suction grasping. Preferably, the grasp is positioned on a product's flat side to establish a secure seal while minimizing the risk of product displacement or toppling. A slight angle of approach that gently presses the product against the surface further enhances stability.
- **Compliant robots are key in dynamic environments**: Compliance is essential when safely operating rigid robotic arms in dynamic environments and alongside humans. To guarantee collision avoidance with humans requires accurate human detection and intent detection. As this is highly complex, we opted for safety through compliance during arm motion and rely on raw lidar data during base motion. Beyond safety, compliance offers inherent forgiveness in the event of grasping failures. Together with our adaptive online task planner, this combination enables our robot to recover or retry autonomously, minimizing human interventions.
- **Whole body control enhances efficiency**: Whole body (or semi-whole body) control simplifies the task of picking products from diverse shelves within a supermarket setting. The expanded configuration space with the additional degrees of freedom significantly enhances planning efficiency and reliability, thereby streamlining operations.
- **Rapid and iterative software development is imperative**: Rapidly iterating our software directly on the physical robot was a critical success factor for us. Swift iterations serve as a reliable indicator of the eventual outcome. Furthermore, our realistic rest lab environment with anticipated operational conditions enhanced overall system robustness.
- **Lack of high quality mobile manipulators hinders progress**: Contrary to prevailing notions, the development of mobile manipulators present substantial challenges, particularly in research. Existing solutions remain scarce, and researchers are often forced to deal with inaccurate and unreliable mobile bases and robotic arms with short battery lives. Similarly, versatile gripper design is an unsolved research topic. This underscores the need for continued exploration and innovation in this domain to bridge existing gaps and facilitate rapid advancements in robotic research.

8.10. CONCLUSION

In this DEMO-paper, we present our approach to order-picking in human-shared retail environments. In contrast to previous approaches for that application [140], we deploy a failure-robust task-planning method that can recover from failure of the individual sub-modules and propose a way to simplify the 'programming' of the robot by leveraging teaching. We show that a significantly simpler robot design than [140] can be successful at the task of in-store order-picking. The approach was evaluated in a realistic store environment and in a lab-like environments without much modifications. In the future, we must address the challenge of interacting with more complex-shaped items, such as fruits and vegetables. That requires the integration of either a gripper switching method or a more intricate gripper design, as suggested in [140].

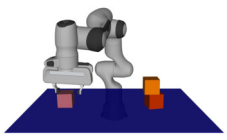


(a) Success-rate across order sizes and failure causes.



(b) Execution times of symbolic actions in seconds. Note that symbolic actions remain active until the desired state, i.e., product in basket, is reached or a failure occurs. In that case, the symbolic action is re-attempted.

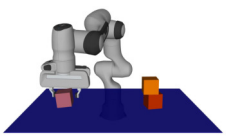
Figure 8.16: Success-rate and action execution times in realistic store environment. A total of $N = 45$ were performed.



9

Conclusion and discussion

The final chapter of this thesis summarizes and discusses the main findings. We discuss the limitations of Optimization Fabrics (fabrics) and propose potential future directions of research. Specifically, we point the reader to lines of research that could be integrated to further promote the framework. Finally, we discuss common questions from the public when it comes to Trajectory Generation (TG) and robotics in general. The chapter is closed by a vision on the future of robotics by the author.



9.1. CONCLUSION

This thesis was stated in the context of labor shortage in the global North and the need for further automation to combat demographic change. In particular, we focused on the problem of TG for mobile manipulators in human-shared environments. This thesis proposed several TG methods to allow the deployment of mobile manipulators outside their classical cages. The core methods are centered around a geometric interpretation of TG for manipulators and mobile manipulators. Specifically, we extended the framework of fabrics to more dynamic environments, proposed a symbolic implementation to allow for fast parameter tuning at runtime, and demonstrated its effectiveness in a real demonstration case. Along the way, we showed how Model Predictive Control (MPC) can be used for mobile manipulators and why geometric approaches tend to outperform optimization-based methods for these systems. All results were validated extensively in simulation and verified in real-world experiments. In the following, we summarize the key findings of this thesis.

9.1.1. MPC FOR MOBILE MANIPULATORS

Chapter 4 applied the widely used method of Free Space Decomposition (FSD) to mobile manipulators which led to several improvements. First, it allowed to control the system in a coupled way reducing overall execution time, similar to what had been done in [81] with dynamic weighing. Second, it reduced the required accuracy of the perception pipeline as raw point clouds or occupancy grids can be used. Third, we showed that reducing the number of constraints achieved by using FSD allows for substantially faster computation times that do not depend on the number of obstacles in the environment, e.g. a solver time decrease of a factor 3 for 100 obstacles. In Chapter 5, we additionally showed that geometric approaches tend to outperform MPC formulations despite weaker theoretical guarantees.

9.1.2. DYNAMIC FABRICS

Chapter 5 generalized the framework of fabrics to dynamic environments. We showed that fabrics can be formulated on moving reference frames, in such a way that trajectory following and collision avoidance with dynamic obstacles can be combined with other, static components to the TG problem. Specifically, the *dynamic pullback* operator brings a dynamically defined component into the static manifold of the configuration space while making the motion of the frame a runtime variable of the root geometry. This chapter showed that reference velocity integration is required to handle moving obstacles. Additionally, integration of global path planning in arbitrary manifolds is possible to the generalization. In particular, this chapter proved that convergence to reference trajectories can be guaranteed. Finally, this chapter presented results on trajectory following with a mobile manipulator and collision avoidance with moving humans in the workspace. Additionally, we provided a required extension to fabrics for non-holonomic systems, such as differential drive robots.

9.1.3. SYMBOLIC FABRICS

Chapter 6 highlighted the existence of a closed-form solution to TG when formulated as a geometric problem. This property is exploited in a purely symbolic implementation of fabrics which allows for solver times around 1ms. This chapter additionally proposed to use a Bayesian optimization to tune parameters for fabrics. The experiments showed that

expert-level tuning can be reached within 50 iterations, i.e. 50 full runs in simulation. Additionally, the results revealed that parameter sets can be transferred between similar embodiments. Finally, qualitative results on a real robot are presented confirming the applicability to both manipulation and mobile manipulation.

9.1.4. IMPLICIT ENVIRONMENT REPRESENTATIONS

Chapter 7 proposed the integration of several implicit environment representations into the framework of fabrics. Specifically, we showed how Signed Distance Fields (SDF) can be integrated using numeric gradients and how FSD can be used using plane constraints for which closed-form distance functions can be computed. Additionally, we compared these two representations with the usage of raw point clouds. The experiments reveal that success rates for all proposed implicit representations remain high (>80%) when transitioning from static to dynamic environments while performance of explicit representations drops substantially. When used with a robotic manipulator, success rates of raw point clouds is highest up to a noise level of $\sigma = 0.02$. For all methods and both embodiments, solvertimes remain below 30ms allowing for reactive deployment.

9.1.5. REAL-WORLD DEMONSTRATION

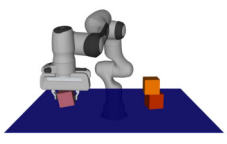
Chapter 8 demonstrated that fabrics are not a mere theoretical framework but can be used in real-world manipulation demos. In the combination with a modern decision-making framework, and state-of-the-art perception, we demonstrated how fabrics can be used to generate trajectories in complex environments with high-dimensional robots. Besides, it was shown that different paradigms from TG in manipulation, such as learning-based methods and redundancy resolution, can be easily integrated into the framework.

9.2. DISCUSSION

This thesis focused on different methods for TG that are capable to generate motion for mobile manipulators in human-shared environments. That included the adaption to the widely used method of MPC to whole-body control for mobile manipulators. The main part of this thesis was dedicated to a geometric interpretation of TG. Although several advancements for bringing mobile manipulators to human-shared environments were made, we acknowledge that seeing robots in our daily lives will still be a rare sight for the years to come. In the following, we discuss the limitations of the proposed methods and highlight important missing pieces to unlock the full potential of mobile manipulators in human-shared environments.

9.2.1. FABRICS OR MPC?

Most readers are likely familiar with Model Predictive Control and may have seen its application to, for example, autonomous driving or mobile manipulation, and some experienced roboticists may be aware of the strengths and weaknesses of MPC. In contrast, fewer readers may have heard of fabrics before. Often times, both methods are put in the same category, because MPC is based on an *optimization* which is part of the name of Optimization Fabrics (fabrics). However, the methods are fundamentally different, and it is important to analyze the differences and similarities between these methods. MPC is centered around the



formulation of a constrained optimization problem. The constraints are the dynamics of the system and avoidance behaviors, such as collision avoidance or joint limit avoidance. The problem is solved up to a defined optimality criteria. In general, the optimization problem is non-linear and non-convex. In contrast, fabrics is a purely geometric method. It is based on the formulation of a smooth manifold in the configuration space of the robot. Then, the solution to the TG problem is solution to the geodesic equation where the manifolds metric is non-Riemannian in general. Importantly, the entire problem is encoded in the metric by iteratively adding avoidance behaviors, such as collision avoidance or joint limit avoidance. As the trajectory is as consequence of solving the geodesic equation in each time step, a closed-form solution is available for explicit behaviors as discussed in this thesis. This fundamentally different view on TG has severe implications on practical implementations.

Reactivity One important aspect of TG in human-shared environments is the ability to react fast to changing environments. The *speed* of reaction of different approaches is captured in what we call *reactivity* and is uniquely defined by the frequency at which a TG problem can be deployed, which is a direct consequence of the computational complexity of the method.

Due to the closed-form solution, computing actions can be achieved in roughly 1ms for fabrics on a 7-Degree of Freedom (DoF) manipulator. Therefore, fabrics can arguably be considered a controller, that captures geometric environment aspects. In contrast, the solution to the optimization problem in MPC is generally not available in closed-form. Therefore, the optimization problem has to be solved in each time step. Additionally, the problem is usually non-convex making the solution costly. In numbers, when encoding the same TG problem in an MPC formulation as in fabrics, solver times between 10ms and 100ms are realistic. That substantially reduces reactivity of the system making it less suitable for fast changing environments.

Global optimality While reactivity is an important aspect to collision avoidance in dynamic environments, convergence to the goal state is equally important when it comes to task success of manipulation tasks. In MPC formulations, future states are considered when constructing the optimization problem, thus the evolution and, importantly, goal approaching is optimized over the entire horizon. In the extreme, an infinitely long horizon and a global solver would lead to a globally optimal solution. This is practically impossible to achieve, such that the horizon is usually limited to a few stages, and the remaining stages are approximated by a terminal cost. Nevertheless, MPC formulations practically show strong behavior in terms avoidance of local minima. In contrast, fabrics is a purely local method, with no consideration of future states. It relies purely on the shape of the manifold to guide the motion. This shape however is not guaranteed to be free of local minima. In fact, we argued in Chapter 5 that local minima are likely to occur in cluttered environments. This is a severe limitation which must be addressed by either global guidance as we suggested, or by further analyzes of the generated manifolds.

Motion design We call *motion design* the process of encoding behaviors into the TG problem. Practically, motion design describes the process for a user to encode different

desired behaviors into the TG problem, such that the robot behaves as intended by the application. In MPC formulations, the behavior is mainly influenced by the objective function and its weighing factors if multiple objectives are present. This scalar encoding often proves to be difficult, because individual objectives are highly coupled. In contrast, fabrics allow for *iterative* motion design. Specifically, the defining manifold can be iteratively enhanced, such that individual components can be tested and tuned separately. For example, in the experiments of this thesis, we usually defined behaviors in order of decreasing importance, starting with joint limit avoidance and ending with goal-attraction. That makes the method user-friendly and individual components exchangeable if an application might require that. Moreover, individual behaviors can be complex, such as collision avoidance defined by SDF as integrated in Chapter 7, or goal reaching using deep neural networks, as demonstrated in [161].

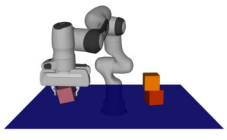
In summary, fabrics and MPC are two fundamentally different approaches to TG. MPC is usually used as an approximation of the infinite time-horizon optimal control problem. Using the terminal cost appropriately, MPC formulations lead to reactive behaviors that are able to avoid local minima up to a certain extent. In contrast, fabrics excels in reactivity and motion design and should often be regarded as an informed controller. That comes at the cost of global properties which must be taken into account when using the method in cluttered environments. In this thesis, we showed how global guidance can be integrated using either conventional path planning (Chapter 5) or learning-from-demonstration methods (Chapter 8).

9.2.2. THE FUTURE OF FABRICS

After having discussed differences and similarities between MPC and fabrics, we name the core limitations of our proposed methods and point the reader to potential future directions of research.

Continuous time A practical limitation of fabrics –as for any other geometric method defined in the continuous time domain– is that the method is not directly applicable to real-world systems. Specifically, the method has to be discretized and approximated leading to inaccuracies when tracking continuous trajectories. A thorough analysis on convergence of the discretization when applied to real robots is missing to this point.

Tuning Despite improved design-invariant properties on stability and convergence compared to Riemannian Motion Policies (RMPs), the tuning of fabrics requires some intuition that usually comes from experience. We aimed to tackle that problem in Chapter 6, but further research should be conducted to simplify the tuning process. Moreover, all behaviors used in this thesis are hand-crafted, in Chapter 6 up to certain level of parameterization. Hand-crafting TG methods is time-consuming and does not seem to be the right approach in the long run. Especially considering the success of learning-based methods in other fields. Therefore, we argue that fabrics should be seen as a compact, yet rich, encoding of TG that serves only as the building structure for nuanced, learned behaviors. By ensuring that deep neural networks exhibit the properties required by the theory of fabrics, arbitrarily complex behaviors can be constructed and importantly combined into a single policy. This approach



has been experienced with in [161] and in Chapter 8 of this thesis.

Geometric algebra When putting a method into the context of geometric interpretation, it is natural to ask whether other mathematical concepts can be applied to the problem. Recently, the field of geometric algebra has gained attention in the robotics community, because it allows to express different geometric shapes and operations in a single structure, see [162] for example. While low-order geometric algebras can unify translation and rotation, higher order geometric algebras can be used to unify different attractors, as they are required by fabrics. For example, attractors in this thesis are limited to sets of translations, e.g. an end-effector pose can be encoded as three translational attractors. Higher order algebra allow encoding attraction to a circle, line, pose or sphere as an object of the same algebra [162]. From a conceptional point of view, this seems the more compact understanding of attractors, and from a practical point of view, it reduces the amount of parameters required. Future research in fabrics should therefore consider the integration of higher order geometric algebras.

Fabrics in the context of sampling intensive methods In this thesis, we used fabrics as a standalone method for TG, but we argue that the compact encoding as a dynamical system can be deployed as a nuanced low-level controller [61], [163]. That interpretation offers a direct integration with methods that are based on intensive sampling for either learning or online-sampling. For one example, fabrics can be used as a safety filter during training and/or during testing. Acting as a filter, fabrics can ensure that joint limits are respected, self-collision is avoided and that action spaces defined in task spaces are tracked in joint space. Especially the latter is a common assumption for reinforcement-learning agents, see [164], [165] for examples. In this setting, fabrics replace the simplistic low-level controller that is already needed to track end-effector actions. Similar to goal definitions in higher-order algebras, see Section 9.2.2, goal attraction and/or damping in the final policy equation

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f} + \partial_{\boldsymbol{q}}\psi + \boldsymbol{B}\dot{\boldsymbol{q}} = 0$$

could be computed based on visuals inputs, such as images or point clouds. By doing so complex behaviors can be achieved without the need for hand-crafting complicated objectives. This direction of research has already been experimented with in [163], [166]. Finally, the low computational costs of fabrics can be exploited for forward rollouts in sampling-based methods. The approach of forward rollouts is often used when objectives are non-convex or gradients are unknown. Then sampling is beneficial and can lead to high-quality trajectories, see [126]. In this context, fabrics can be used as an initial guess to form the mean of the sampling distribution. Potentially, that can lead to a reduced sample size. Similarly, pure fabrics policies can be used to generate non-optimal trajectories for various tasks that are subsequently used for training.

Fabrics for manipulation All approaches presented in this work treated collisions as harmful and integrated the environment in avoidance behaviors. Collision avoidance is critical for safety in human-shared environments, but contacts are not *per-se* harmful, in fact, we, humans, make contact very often to interact with the environment or to better understand characteristics of the tasks we want to achieve. Therefore, one clear direction of future

research is the application of fabrics for actual manipulation tasks. One way is the presented approach in Chapter 8 where fabrics are used to generate trajectories that actually make contact. However, the contact was not actually considered in the TG problem. Similarly in terms of contact integration is the work of [166]. Trying to lift the assumptions that contacts occur at *close-to-zero*-velocity, recent works integrate collision, or positively phrased contact, into the control formulation, see for example [167]–[169]. These approaches integrate the impact-awareness into a task-space controller. As fabrics can be interpreted as a generalization of Cartesian Impedance Control (CIC), an extension for impact-aware fabrics seems possible and should be investigated in the future.

9.2.3. CONSIDERATIONS AND VISION

After having discussed the advantages and limitations of methods related to the chapters of this thesis, I also want to give a broader perspective on robotics. In particular, I want to shed light on some reasons why robots have not entered human shared environments in a more widespread manner. To make this section a bit more lively, I shall ask some questions, that are often asked by the public, and try to answer them in a more philosophical way.

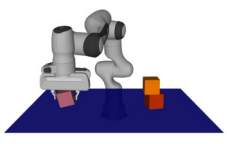
Why does the robot move so slowly?

This is the question that every roboticist – especially those working on TG – has to endure when showing their work to the public. While most, including the author, are quick to point out that the robot moves slowly to ensure safety, the true reason is more complex. Execution speed is dependent on (a) the ability to move fast, (b) the ability to execute trajectories accurately when moving fast, (c) the ability to make state estimates fast, (d) the ability to compute trajectories fast, (e) the effect of speed of motion on the environment, and, of course, (f) safety considerations. Let me treat these points in turns:

(a) When a robot's motors are not powerful enough to move fast, the robot will move slowly. This is obvious and should not be used as an excuse, because all robots used in this thesis are physically capable of moving much faster than shown in the experiments.

(b) Moving fast however is not sufficient as accuracy is important in manipulation tasks. It turns out that modern hardware is capable of speeding up moving between waypoints defined in the configuration space. After all, robots in cages are moving extremely fast with super-human accuracy, so this cannot be the reason for slow motion either.

(c) When deployed in human-shared environments however, robots cannot simply play back sequences of waypoints in an open-loop manner. Instead, the environment has to be constantly perceived and the state of the robot must be estimated at the same speed. It is difficult to achieve required update frequency with computer vision, but even state estimates related to the robot itself fail to be accurate at high speeds. For example, the joint velocities of the Panda robot used primarily in this thesis are not accurate at high speeds. That proves to be highly problematic when using reactive TG methods. In this thesis, the problem was usually addressed by low-pass filtering the joint velocities which on the flip side introduces a delay of the signal. Either the estimates must be improved without introducing delays or TG methods must be robust against noise estimates. Inaccurate, slow or delayed sensing seems an important reason for slow motion in human-shared environments.



(d) The ability to compute trajectories fast is discussed in Section 9.2.1 and I believe that fabrics is a good approach to address slow computations causing slow-moving robots. However, more widely used methods do not offer this speed and are therefore contributing to slow motion.

(e) What is often under-appreciated by the public is that speeding up motions has a direct effect on the interactions with the environment. For example, when grasping objects with a vacuum gripper, as done in Chapter 8, the motion must be slow when approaching an item to avoid it tipping over. This logical connection is often considered trivial by the public, but is very difficult to encode in a robot's behavior. This leads to a tradeoff between speed of motion and success rate, where the latter is usually favored.

(f) And finally, speed of motion and safety are in direct conflict. We put fast robots in cages because their kinetic energy is dangerous to humans. Having the same kind of robots in human-shared environments naturally results in slow motions to compensate for the lack of the cage as a safety measure. Unless, we start relying on soft robots to imitate human hardware, this conflict seems hard to impossible to resolve.

Why does the robot choose this highly complicated way of moving?

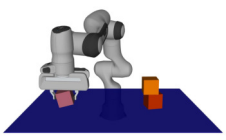
This question often arises when a robot seems to take an unconventional path to reach its destination. The issue hinges on the notion of what is considered *natural*, which essentially boils down to what humans perceive as the best route. In the context of robotics, the optimal path is typically defined as the shortest path in the configuration space, which poses significant challenges due to its high dimensionality and the constraints imposed by obstacles. Consequently, finding this shortest path becomes a difficult task, often addressed through sampling-based methods like Rapidly-exploring Random Trees (RRT), which iteratively sample configurations until a viable sequence of configurations leading to the goal is discovered [29]. However, this approach only guarantees the shortest connection between subsequent samples, not the entire sequence, often resulting in abrupt and jerky motions. Despite efforts to refine these paths, such as through smoothing techniques [170], the fundamental challenge lies in the nature of planning within the configuration space, which tends to yield paths that appear unconventional to human observers. In contrast, the methods outlined in this thesis avoid long-term planning within the configuration space, instead relying on purely reacive TG and global path planning in the workspace, which generally yields paths that appear more natural. Specifically, techniques like shaping the configuration manifold seem to resemble to human path-planning processes [171].

Why does the robot fail at this simple task?

This is not directly a question of the TG problem at hand, but it is often intertangled with it. It touches upon task and motion planning, a widely studied field of robotics these days, see [172] for a recent survey. To answer this question, one has to appreciate the ease of humans to combine several sensors, e.g. vision, feel, sound, etc. with general understanding of physics and highly precise, yet compliant actuation. Some works suggest that the sensors of robots are fundamentally limiting the set of tasks a robot could do [173]. Additionally, our cognitive abilities to understand failure cases, make long term plans and to transfer knowledge are not met by robots. Until great advancements are made, the public will

continue wondering why robots fail at simple tasks.

While our modern societies are desperate for automation in human-shared environments, it is often disappointing to observe the current state of the art in robotics. From an intellectual point of view however, this is truly exciting, as the opportunities to shape robotics seem endless. And, from a philosophical point of view you may ask: *Isn't it quite reassuring that human abilities are still superior to those of robots when it comes to human-shared environments?*



BIBLIOGRAPHY

[1] M. Ince Yenilmez, "Economic and social consequences of population aging the dilemmas and opportunities in the twenty-first century", *Applied Research in Quality of Life*, vol. 10, pp. 735–752, 2015.

[2] J. McGrath, *Report on labour shortages and surpluses*. Publications Office of the European Union, 2021.

[3] V. Astrov, S. Leitner, R. Grieveson, D. Hanzl-Weiss, I. Mara, and H. Weinberger-Vidovic, "How do economies in EU-CEE cope with labour shortages?", wiiw Research Report, Tech. Rep., 2021.

[4] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision", in *International Conference on Machine Learning*, PMLR, 2023, pp. 28 492–28 518.

[5] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, "Training compute-optimal large language models", *arXiv preprint arXiv:2203.15556*, 2022.

[6] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review", *Computational intelligence and neuroscience*, vol. 2018, 2018.

[7] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

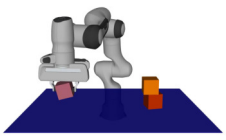
[8] M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled mobile manipulation via trajectory optimization with free space decomposition", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 12 759–12 765.

[9] F. Noroozi, M. Daneshmand, and P. Fiorini, "Conventional, heuristic and learning-based robot motion planning: Reviewing frameworks of current practical significance", *Machines*, vol. 11, no. 7, p. 722, 2023.

[10] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning", *IEEE transactions on systems, man, and cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[11] Y. K. Hwang, N. Ahuja, *et al.*, "A potential field approach to path planning.", *IEEE transactions on robotics and automation*, vol. 8, no. 1, pp. 23–32, 1992.

[12] N. Hogan, "Impedance Control: An Approach to Manipulation: Part II — Implementation", *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 8–16, Mar. 1985.



[13] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in motion optimization", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4202–4209.

[14] H. Klein, N. Jaquier, A. Meixner, and T. Asfour, "On the design of region-avoiding metrics for collision-safe motion generation on riemannian manifolds", in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2346–2353.

[15] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, *Riemannian motion policies*, Jul. 25, 2018.

[16] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmp flow: A computational graph for automatic motion policy generation", in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, Springer, 2020, pp. 441–457.

[17] ——, "Rmpflow: A geometric framework for generation of multitask motion policies", *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 968–987, 2021.

[18] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, *Optimization fabrics*, Aug. 21, 2020.

[19] M. Xie, K. Van Wyk, A. Li, M. A. Rana, Q. Wan, D. Fox, B. Boots, and N. Ratliff, "Geometric fabrics for the acceleration-based design of robotic motion", 2021.

[20] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning", *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1305–1317, 2014.

[21] J. Mišeikis, P. Caroni, P. Duchamp, A. Gasser, R. Marko, N. Mišeikienė, F. Zwilling, C. De Castelbajac, L. Eicher, M. Früh, *et al.*, "Lio-a personal robot assistant for human-robot interaction and care applications", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5339–5346, 2020.

[22] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths", in *Field and Service Robotics: Results of the 7th International Conference*, Springer, 2010, pp. 69–78.

[23] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer, 2019, vol. 49.

[24] J. M. Lee and J. M. Lee, *Smooth manifolds*. Springer, 2012.

[25] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[26] S. Lang, *Fundamentals of differential geometry*. Springer Science & Business Media, 2012, vol. 191.

[27] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, *Generalized nonlinear and finsler geometry for robotics*, Jul. 2, 2021.

[28] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, and N. D. Ratliff, *Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior*, Jan. 18, 2022.

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

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 1985, pp. 500–505.

[31] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning—a review", *Frontiers in Robotics and AI*, vol. 7, p. 590 681, 2020.

[32] A. Albu-Schaffer and G. Hirzinger, "Cartesian impedance control techniques for torque controlled light-weight robots", in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 1, 2002, pp. 657–663.

[33] L. van der Spaa, G. Franzese, J. Kober, and M. Gienger, "Disagreement-aware variable impedance control for online learning of physical human-robot cooperation tasks", in *ICRA 2022: IEEE International Conference on Robotcs and Automation*, 2022.

[34] S. Hjorth, J. Lachner, A. Ajoudani, and D. Chrysostomou, "Enabling passivity for cartesian workspace restrictions", English, in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, United States: IEEE, Jan. 2024.

[35] J. Lachner, F. Allmendinger, S. Stramigioli, and N. Hogan, "Shaping impedances to comply with constrained task dynamics", *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2750–2767, 2022.

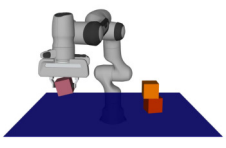
[36] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "Ilosa: Interactive learning of stiffness and attractors", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7778–7785.

[37] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[38] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields", in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2008, pp. 91–98.

[39] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments", in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 44, IFAC Secretariat, 2011, pp. 6842–6850.

[40] S. Du, W. Shang, S. Cong, C. Zhang, and K. Liu, "Moving obstacle avoidance of a 5-dof robot manipulator by using repulsive vector", in *2017 IEEE International Conference on Robotics and Biomimetics, ROBIO 2017*, vol. 2018-Janua, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1–6.



[41] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities", in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 3109–3116.

[42] J. Haviland and P. Corke, "A purely-reactive manipulability-maximising motion controller", *arXiv preprint arXiv:2002.11901*, 2020.

[43] ——, "Neo: A novel expeditious optimisation algorithm for reactive motion control of manipulators", en, *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1043–1050, Apr. 2021.

[44] T. Urakubo, "Stability analysis and control of nonholonomic systems with potential fields", *Journal of Intelligent & Robotic Systems*, vol. 89, no. 1-2, pp. 121–137, 2018.

[45] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance", *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[46] F. Zhang, N. Li, T. Xue, Y. Zhu, R. Yuan, and Y. Fu, "An improved dynamic window approach integrated global path planning", in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2019, pp. 2873–2878.

[47] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments", in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[48] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance", in *Springer Tracts in Advanced Robotics*, vol. 70, 2011, pp. 3–19.

[49] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots", in *Springer Tracts in Advanced Robotics*, vol. 83 STAR, 2012, pp. 203–216.

[50] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance", *Autonomous Robots*, vol. 32, pp. 433–454, 2012.

[51] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models", *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[52] L. Huber, J.-J. Slotine, and A. Billard, "Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds", *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3113–3132, 2022.

[53] ——, "Avoidance of concave obstacles through rotation of nonlinear dynamics", *IEEE Transactions on Robotics*, vol. 40, pp. 1983–2002, 2024.

[54] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey", *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[55] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors", *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[56] R. Pérez-Dattari and J. Kober, "Stable motion primitives via imitation and contrastive learning", *IEEE Transactions on Robotics*, 2023.

[57] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[58] A. Li, C.-A. Cheng, M. A. Rana, M. Xie, K. Van Wyk, N. Ratliff, and B. Boots, *Rmp2: A structured composable policy class for robot learning*, Mar. 10, 2021.

[59] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[60] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural autonomous navigation with riemannian motion policy", in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 8860–8866.

[61] N. Ratliff and K. Van Wyk, "Fabrics: A foundationally stable medium for encoding prior experience", *arXiv preprint arXiv:2309.07368*, 2023.

[62] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, 2022.

[63] I. R. Manchester and J.-J. E. Slotine, "Control Contraction Metrics: Convex and Intrinsic Criteriafor Nonlinear Feedback Design", en, *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 3046–3053, 2017.

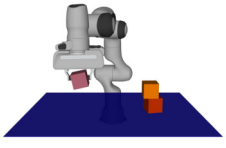
[64] B. Yi, R. Wang, and I. R. Manchester, "On necessary conditions of tracking control for nonlinear systems via contraction analysis", in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 2000–2005.

[65] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation", in *Conference on Robot Learning*, PMLR, 2022, pp. 750–759.

[66] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality", en, *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[67] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization", en, *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[68] S. S. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations", *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265–293, 1988.



[69] M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled mobile manipulation via trajectory optimization with free space decomposition", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 12 759–12 765.

[70] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control", in *49th IEEE Conference on Decision and Control (CDC)*, IEEE, 2010, pp. 6137–6142.

[71] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles", *Control Engineering Practice*, vol. 61, pp. 307–316, 2017.

[72] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[73] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model", *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2018.

[74] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments", in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 1934–1940.

[75] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments", *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[76] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot", *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2016.

[77] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression", *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[78] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[79] S. Ide, T. Takubo, K. Ohara, Y. Mae, and T. Arai, "Real-time trajectory planning for mobile manipulator using model predictive control with constraints", in *URAI 2011 - 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2011, pp. 244–249.

[80] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constraint-based model predictive control for holonomic mobile manipulators", in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1473–1479.

[81] G. Buizza Avanzini, A. M. Zanchettin, and P. Rocco, *Constrained model predictive control for mobile robotic manipulators*, 2018.

[82] S. M. LaValle, “Planning algorithms”, *Planning Algorithms*, vol. 9780521862, pp. 1–826, 2006.

[83] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, “Continuous-time gaussian process motion planning via probabilistic inference”, *International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[84] D. J. Webb and J. Van Den Berg, “Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics”, in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 5054–5061.

[85] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, “Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation”, *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 277–283, 2019.

[86] J. J. Kuffner and S. M. La Valle, “Rrt-connect: An efficient approach to single-query path planning”, in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[87] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles”, *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[88] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, “Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning”, in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 5113–5120.

[89] M. Stilman, “Global manipulation planning in robot joint space with task constraints”, *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

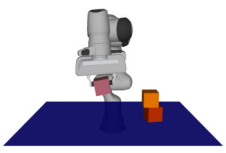
[90] D. Berenson, S. Srinivasa, and J. Kuffner, “Task space regions: A framework for pose-constrained manipulation planning”, *International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[91] Z. Kingston, M. Moll, and L. E. Kavraki, “Exploring implicit spaces for constrained sampling-based planning”, *International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[92] A. H. Qureshi, J. Dong, A. Baig, and M. C. Yip, “Constrained motion planning networks x”, *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 868–886, 2021.

[93] S. Thakar, L. Fang, B. Shah, and S. Gupta, “Towards time-optimal trajectory planning for pick-and-transport operation with a mobile manipulator”, in *IEEE International Conference on Automation Science and Engineering*, vol. 2018-Augus, IEEE Computer Society, 2018, pp. 981–987.

[94] S. Thakar, P. Rajendran, V. Annem, A. Kabir, and S. Gupta, “Accounting for part pose estimation uncertainties during trajectory generation for part pick-up using mobile manipulators”, in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1329–1336.



[95] A. Jain and C. C. Kemp, "Behavior-based door opening with equilibrium point control", *RSS Workshop Mobile Manipulation in Human Environments*, 2009.

[96] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 1799–1806.

[97] C. Li, F. Xia, R. Martin-Martin, S. Savarese, R. Martín-Martín, and S. Savarese, "Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators", 2019.

[98] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds", in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, pp. 797–803.

[99] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning", in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 6015–6022.

[100] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees", *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[101] A. Domahidi and J. Jerez, *Forces professional*, 2014–2019.

[102] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: An efficient implementation of interior-point... methods for multistage nonlinear nonconvex programs", *International Journal of Control*, pp. 1–17, 2017.

[103] D. Hrovat, S. Di Cairano, H. Tseng, and I. Kolmanovsky, "The development of model predictive control in automotive industry: A survey", in *2012 IEEE International Conference on Control Applications*, 2012, pp. 295–302.

[104] M. Bednarczyk, H. Omran, and B. Bayle, "Model predictive impedance control", in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 4702–4708.

[105] S. Richter, S. Mariéthoz, and M. Morari, "High-speed online mpc based on a fast gradient method applied to power converter control", in *Proceedings of the 2010 American Control Conference, ACC 2010*, IEEE Computer Society, 2010, pp. 4737–4743.

[106] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, "Automatic tuning for data-driven model predictive control", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7379–7385.

[107] M. Spahn, C. Salmi, and J. Alonso-Mora, "Local planner bench: Benchmarking for local motion planning", *arXiv preprint arXiv:2210.06033v1*, 2022.

[108] C. Meo, G. Franzese, C. Pezzato, M. Spahn, and P. Lanillos, "Adaptation through prediction: Multisensory active inference torque control", *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 1, pp. 32–41, 2022.

[109] M. Xie, K. Van Wyk, A. Li, M. A. Rana, Q. Wan, D. Fox, B. Boots, and N. Ratliff, *Geometric fabrics for the acceleration-based design of robotic motion*, Jun. 25, 2021.

[110] M. Spahn, M. Wisse, and J. Alonso-Mora, "Dynamic optimization fabrics for motion generation", *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2684–2699, 2023.

[111] A. Loquercio, A. Saviolo, and D. Scaramuzza, "Autotune: Controller tuning for high-speed flight", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4432–4439, 2022.

[112] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, "Automatic tuning for data-driven model predictive control", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China: IEEE, May 30, 2021, pp. 7379–7385.

[113] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice", *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[114] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*, ser. The Springer Series on Challenges in Machine Learning. Springer International Publishing, 2019.

[115] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework", in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[116] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning", *arXiv preprint arXiv:1611.01578*, 2016.

[117] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.", *Journal of machine learning research*, vol. 13, no. 2, 2012.

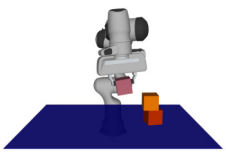
[118] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization", *Advances in neural information processing systems*, vol. 24, 2011.

[119] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020", in *NeurIPS 2020 Competition and Demonstration Track*, PMLR, 2021, pp. 3–26.

[120] M. Spahn, *Urdf-environment*, https://github.com/maxspahn/gym_envs_urdf, version 1.0.0, 2022.

[121] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing", in *Springer Handbook of Robotics*, Springer, 2016, pp. 841–866.

[122] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.



[123] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments", *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[124] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments", in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 1934–1940.

[125] M. Pantic, I. Meijer, R. Bähnemann, N. Alatur, O. Andersson, C. Cadena, R. Siegwart, and L. Ott, "Obstacle avoidance using raycasting and riemannian motion policies at khz rates for mavs", in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1666–1672.

[126] C. Pezzato, C. Salmi, M. Spahn, E. Trevisan, J. Alonso-Mora, and C. H. Corbato, *Sampling-based model predictive control leveraging parallelizable physics simulations*, 2023.

[127] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, "Curobo: Parallelized collision-free robot motion generation", in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8112–8119.

[128] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized deep signed distance fields for reactive motion generation", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6673–6680.

[129] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control", *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2023.

[130] M. Spahn and J. Alonso-Mora, "Autotuning symbolic optimization fabrics for trajectory generation", in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 11 287–11 293.

[131] K. Van Wyk, A. Handa, V. Makoviychuk, Y. Guo, A. Allshire, and N. D. Ratliff, *Geometric Fabrics: A Safe Guiding Medium for Policy Learning*, arXiv:2405.02250 [cs], May 2024.

[132] D. Rodriguez-Guerra, G. Sorrosal, I. Cabanes, and C. Calleja, "Human-robot interaction review: Challenges and solutions for modern industrial environments", *IEEE Access*, vol. 9, pp. 108 557–108 578, 2021.

[133] M. Kronmüller, A. Fielbaum, and J. Alonso-Mora, "Online flash delivery from multiple depots", *Transportation Letters*, pp. 1–17, 2023.

[134] E. Guizzo and E. Ackerman, "The hard lessons of DARPA's robotics challenge [news]", *IEEE Spectrum*, vol. 52, no. 8, pp. 11–13, 2015.

[135] T. Wisspeintner, T. Van Der Zant, L. Iocchi, and S. Schiffer, "RoboCup@ Home: Scientific competition and benchmarking for domestic service robots", *Interaction Studies*, vol. 10, no. 3, pp. 392–426, 2009.

[136] C. H. Corbato, M. Bharatheesha, J. Van Egmond, J. Ju, and M. Wisse, "Integrating different levels of automation: Lessons from winning the amazon robotics challenge 2016", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4916–4926, 2018.

[137] G. K. Kraetzschmar, N. Hochgeschwender, W. Nowak, F. Hegger, S. Schneider, R. Dwiputra, J. Berghofer, and R. Bischoff, "RoboCup@ Work: Competing for the factory of the future", in *RoboCup 2014: Robot World Cup XVIII 18*, Springer, 2015, pp. 171–182.

[138] M. Sereinig, W. Werth, and L.-M. Faller, "A review of the challenges in mobile manipulation: Systems design and RoboCup challenges.", *Elektrotech. Informationstechnik*, vol. 137, no. 6, pp. 297–308, 2020.

[139] S. Thakar, S. Srinivasan, S. Al-Hussaini, P. M. Bhatt, P. Rajendran, Y. Jung Yoon, N. Dhanaraj, R. K. Malhan, M. Schmid, V. N. Krovi, *et al.*, "A survey of wheeled mobile manipulation: A decision-making perspective", *Journal of Mechanisms and Robotics*, vol. 15, no. 2, p. 020 801, 2023.

[140] M. Bajracharya, J. Borders, R. Cheng, D. Helmick, L. Kaul, D. Kruse, J. Leichty, J. Ma, C. Matl, F. Michel, C. Papazov, J. Petersen, K. Shankar, and M. Tjersland, "Demonstrating mobile manipulation in the wild: A metrics-driven approach", in *Robotics: Science and Systems XIX*, ser. RSS2023, Robotics: Science and Systems Foundation, Jul. 2023.

[141] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments", *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, 2017.

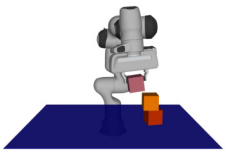
[142] P. Štibinger, G. Broughton, F. Majer, Z. Rozsypálek, A. Wang, K. Jindal, A. Zhou, D. Thakur, G. Loianno, T. Krajník, and M. Saska, "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2595–2602, 2021.

[143] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, "Model predictive robot-environment interaction control for mobile manipulation tasks", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 1651–1657.

[144] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning", *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4569–4574, 2011.

[145] R. Bormann, B. F. de Brito, J. Lindermayr, M. Omainska, and M. Patel, "Towards automated order picking robots for warehouses and retail", in *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings 12*, Springer, 2019, pp. 185–198.

[146] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning", *Sensors*, vol. 20, no. 3, p. 939, 2020.



[147] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware", *arXiv preprint arXiv:2304.13705*, 2023.

[148] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation", *arXiv preprint arXiv:2401.02117*, 2024.

[149] C. Pezzato, C. H. Corbato, S. Bonhof, and M. Wisse, "Active inference and behavior trees for reactive action planning and execution in robotics", *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1050–1069, 2023.

[150] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: A process theory", *Neural computation*, vol. 29, no. 1, pp. 1–49, 2017.

[151] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies", *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[152] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, and J. Kober, "Interactive imitation learning in robotics: A survey", *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.

[153] E. Hamuda, B. Mc Ginley, M. Glavin, and E. Jones, "Improved image processing-based crop detection using kalman filtering and the hungarian algorithm", *Computers and electronics in agriculture*, vol. 148, pp. 37–44, 2018.

[154] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system", in *2016 6th international conference on system engineering and technology (ICSET)*, IEEE, 2016, pp. 109–115.

[155] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning", *CoRR*, vol. abs/1703.05175, 2017.

[156] S. X. Hu, D. Li, J. Stühmer, M. Kim, and T. M. Hospedales, *Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference*, 2022.

[157] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, *Emerging properties in self-supervised vision transformers*, 2021.

[158] M. Kmich, H. Karmouni, I. Harrade, A. Daoui, and M. Sayyouri, "Image-based visual servoing techniques for robot control", in *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2022, pp. 1–6.

[159] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration", *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[160] S. Behnke, J. A. Adams, and D. Locke, "The $10 million ana avatar xprize competition: How it advanced immersive telepresence systems", *IEEE Robotics & Automation Magazine*, vol. 30, no. 4, pp. 98–104, 2023.

[161] M. Xie, A. Handa, S. Tyree, D. Fox, H. Ravichandar, N. D. Ratliff, and K. Van Wyk, "Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration", in *Conference on Robot Learning*, PMLR, 2023, pp. 1355–1367.

[162] T. Löw and S. Calinon, "Geometric algebra for optimal control with applications in manipulation tasks", *IEEE Transactions on Robotics*, 2023.

[163] K. V. Wyk, A. Handa, V. Makoviychuk, Y. Guo, A. Allshire, and N. D. Ratliff, *Geometric fabrics: A safe guiding medium for policy learning*, 2024.

[164] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, "Self-supervised policy adaptation during deployment", *arXiv preprint arXiv:2007.04309*, 2020.

[165] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 611–13 617.

[166] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. Van Wyk, "Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics", *arXiv preprint arXiv:2407.02274*, 2024.

[167] Y. Wang, N. Dehio, A. Tanguy, and A. Kheddar, "Impact-aware task-space quadratic-programming control", *The International Journal of Robotics Research*, vol. 42, no. 14, pp. 1265–1282, 2023.

[168] H. Khurana and A. Billard, "Motion planning and inertia-based control for impact aware manipulation", *IEEE Transactions on Robotics*, 2023.

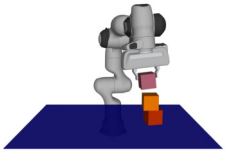
[169] L. Yan, T. Stouraitis, J. Moura, W. Xu, M. Gienger, and S. Vijayakumar, "Impact-aware bimanual catching of large-momentum objects", *IEEE Transactions on Robotics*, 2024.

[170] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[171] H. Klein, N. Jaquier, A. Meixner, and T. Asfour, "A riemannian take on human motion analysis and retargeting", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 5210–5217.

[172] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning", *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[173] A. Majumdar, Z. Mei, and V. Pacelli, "Fundamental limits for sensor-based robot control", *The International Journal of Robotics Research*, vol. 42, no. 12, pp. 1051–1069, 2023.



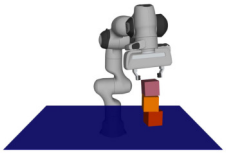
ACKNOWLEDGMENTS

Pursuing a Ph.D. requires being very privileged in a privileged society and being surrounded by amazing people. I am grateful for having been in such a position for the past four years. This time has taught me how engineering sciences work and has shaped me as a person in ways I could not have imagined.

First and foremost, I would like to thank my supervisors, Javier Alonso-Mora and Martijn Wisse, for their guidance, support, and patience with me. You, Javier, introduced me to the world of science and guided me through it. You were able to endure my stubbornness and my complaints, explained to me the importance of rigor, and showed me how to be a good scientist. Towards the end of my journey, and still today, I am grateful for all the exposure to other research groups you gave me. Without you, Martijn, my journey might have stopped much earlier, because the academic world is competitive and often an unhealthy environment. Your interpretation of the role as a professor kept me going and often times reminded me why I started the Ph.D. in the first place. It is refreshing and inspiring how you combine low-level issue-solving, such as robot repairs, with high-level political communication.

I would like to thank the members of my committee for their time and effort in reviewing my work. I am grateful for the feedback and the discussions we had.

Being part of the Cognitive Robotics department was a great opportunity, because I was surrounded by inspiring people and was allowed to shape this environment. I would like to especially thank Corrado, because he was always one step ahead of me to show me around without the slightest trace of arrogance. Making these demos possible would not have been possible without you. Chadi, the most stable pole in this hectic world, thank you for always staying calm around me and sharing the interest in robotics software and vim. Most of the work in this thesis would not have been possible without you. Being a very opinionated person myself, I could not be more thankful for having met you, Giovanni. You slowly convinced me of the concept of learning-from-demonstration and never hesitated to start a heated discussion on the topic of robotic manipulation, or anything else for that matter. The most thrilling and exciting moments in this journey were because of you, #platonics. Mariano, thank you for keeping a calm mind when around Giovanni and me, both when trying to figure out rotations and when deciding on where to have lunch. I also want to thank Bruno, for hinting me towards the geometric approaches, when I was looking for a meaningful research direction. Spending time in the office or the lab would have been very a lonely experience without my fellows. I enjoyed the company of Saray, I expect you to maintain the fabrics package, Luzia for the good anecdotes from the Länd, Elia, Alvaro, Yuije, Rodrigo, Andreu, Tomas. I also want to thank the students I had the pleasure to supervise. Evelien and Paul, thank you for the hard work on path planning and affordances. It was fun working with you, especially at times, when we were all isolated at home. Thank you, Alex, for



accepting the challenge of exploring learning-based methods with me. Thank you, Caspar, for helping to shape the idea of implicit environment representations for fabrics. I would also like to thank all the secretaries and the technical staff at CoR for their support and help with us, the often stubborn Ph.D. students.

Maxi, thank you for teaming up in this journey and for all the healthy distractions from the often times, toxic academic environment. I always enjoyed meeting up for dinner and board games with you and Toni. Thank you, Patrick, for all the phone calls discussing natural language processing and startup-life and your persistent demanding for explanations on why I stayed away from machine learning approaches in my research. I am grateful to having met you, Emily, Max and Rose, you do not only were my companions in this exciting journey of parenthood, but also the reason for the most pleasant years in the Netherlands.

If I am here defending my Ph.D. thesis, it is only because of you, Andrea and Roland. Throughout my life, you offered my all the opportunities that allowed me to be here now. You gave me advice when I needed it and gave me the freedom to explore the world, what I now understand is the hardest for parents. I am so thankful for all the support you offered me without expecting much in return in times. Doing a Ph.D. was often mentally demanding, and I would have not managed it without you, Merle. You helped me through the tough moments, that hardly anyone was aware of. You organized my Ph.D. more than I did, you reminded me of deadlines and courses we had to take, but most importantly, you gave me joy in Delft, shared all experiences and teamed up with me for this journey. During my Ph.D., I also enjoyed your birth, Luna. The most joyful moment I could think of. Through the years, you showed me how unimportant my job is in comparison and how incapable the robots are we develop. It is truly heartwarming to have you in my life, waiting for me to come back in the evening and wanting to run around the apartment giggling. My girls, we went through this together and I do not want to have missed it or experienced it without you.

Curriculum Vitæ

Max Spahn

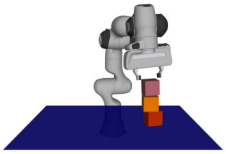
Born on 3 May 1994 in Bergisch Gladbach, Germany

Education

2013	Abitur, Norbert Gynmasium, Dormagen, Germany
2013–2019	B.Sc. & M.Sc. Mechanical Engineering, RWTH Aachen University, Germany
2015–2017	Diplôme d'ingénieur des Arts et Manufactures, École Centrale Paris, France
2020–2024	Ph.D. Robotics, Delft University of Technology, Netherlands

Robotic Competitions

2022	Winner ERF 2022 Franka Emika Hackathon Manipulation Challenge https://www.youtube.com/watch?v=eqpV09Kuc_o
2023	Robothon 2023, Platonics Delft www.platonics.nl
2024	Winner euROBIN Manipulation Skill Versatility Challenge at IROS, 2024, Platonics Delft www.platonics.nl



List of Publications

Journals

4. ☞ **M. Spahn** and J. Alonso-Mora. *"Overcoming Explicit Environment Representations with Geometric Fabrics"*. *Submitted to* IEEE Robotics and Automation Letters, 2024.

3. ☞ C. Pezzato*, C. Salmi*, E. Trevisan*, **M. Spahn**, J. Alonso-Mora and C.H. Corbato. *"Sampling-based model predictive control leveraging parallelizable physics simulations"*. *Submitted to* IEEE Robotics and Automation Letters, 2024.

2. ☞ **M. Spahn**, M. Wisse and J. Alonso-Mora. *"Dynamic Optimization Fabrics for Motion Generation"*. IEEE Transactions on Robotics, 2023.

1. ☞ C. Meo, G. Franzese, C. Pezzato, **M. Spahn** and P. Lanillos. *"Adaptation through prediction: multisensory active inference torque control"*. IEEE Transactions on Cognitive and Developmental Systems, 2022.

Conferences

4. ☞ **M. Spahn**, C. Pezzato, C. Salmi, R. Dekker, C. Wang, C. Pek, J. Kober, J. Alonso-Mora, C. Hernandez Corbato and M. Wisse. *"Demonstrating Adaptive Mobile Manipulation in Retail Environments"*. Robotics: Science and Systems, 2024.

3. ☞ S. Bakker, L. Knödler, **M. Spahn**, W. Böhmer and J. Alonso-Mora. *"Multi-Robot Local Motion Planning Using Dynamic Optimization Fabrics"*. IEEE International Symposium on Multi-Robot & Multi-Agent Systems, 2023.

2. ☞ **M. Spahn** and J. Alonso-Mora. *"Autotuning Symbolic Optimization Fabrics for Trajectory Generation"*. IEEE International Conference on Robotics and Automation, 2023.

1. ☞ **M. Spahn**, B. Brito and J. Alonso-Mora. *"Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition"*. IEEE International Conference on Robotics and Automation, 2021.

Workshops

1. ☞ **M. Spahn**, C. Salmi and J. Alonso-Mora. *"Local Planner Bench: Benchmarking for Local Motion Planning"*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2023.

* indicates equal contributions