EPFL

Thèse n° 10 373

Exact Obstacle Avoidance for Robots in Complex and Dynamic Environments Using Local Modulation

Présentée le 26 avril 2024

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'algorithmes et systèmes d'apprentissage
Programme doctoral en robotique, contrôle et systèmes intelligents

pour l'obtention du grade de Docteur ès Sciences

par

Lukas HUBER

Acceptée sur proposition du jury

Dr D. Gillet, président du jury
Prof. A. Billard, directrice de thèse
Prof. M. Saveriano, rapporteur
Prof. A. Ude, rapporteur
Prof. A. Karimi, rapporteur

École polytechnique fédérale de Lausanne

2024

Abstract

Robots excel in precise maneuvers in static environments guided by offline motion planning. However, the world is dynamic outside the factory floor: people in crowds move constantly, and operators want to work freely around the robot. The dynamical system using closed-form control function that can be evaluated in real-time enables robots to move without prior planning. This Ph.D. thesis uses this approach to modulate an initial dynamical system to enforce collision-free behavior. An analytical environment representation speeds up the computational time, making the algorithms feasible for the real-world. The contributions of this thesis are (1) ensuring convergence for complex obstacles, (2) introducing fast collision avoidance for sensor data, (3) obstacle avoidance based on nonlinear dynamics, and (4) introducing a collision-free torque controller.

In the first part of the thesis, linear dynamics are constrained using modulation of the dynamics. The obstacle avoidance description is inverted to enforce the flow within walls. We prove that such a flow never contacts the boundaries of the enclosing volume and converges to a single point. Additionally, we create smooth motion fields around obstacles with edges, such as tables. The technique avoids static and time-varying environments, which are moving or deforming. The effectiveness is demonstrated on an autonomous mobile robot in a static complex indoor environment and simulated and real crowds.

In the second part, we build on modulation-based obstacle avoidance and extend it to many obstacles and point-cloud descriptions of the environment. Computational cost is reduced by interpreting all obstacles and sensor readings as a single virtual obstacle. The proposed control scheme combines a high-level input with fast reactive obstacle avoidance. The approach is evaluated experimentally on a semi-autonomous wheelchair operator in shared-control mode crossing a busy market-place in the city center.

The third part focuses on augmenting the obstacle avoidance method to nonlinear dynamics and general concave obstacles. The proposed algorithm rotates the initial dynamics to the tangent plane of the obstacle's surface. The method guides nonlinear dynamics around obstacles while ensuring the absence of local minima. The composition of the vector rotational enables moving around graphs of star-shaped obstacles. Experimental validation used the method to ensure collision-free navigation of a 7DoF robot arm around dynamic, concave obstacles.

In the fourth part of the thesis, a force control is proposed, which enhances the resilience against noise and disturbances. Desired obstacle avoidance dynamics are coupled with a passive torque controller to ensure a safe response to external perturbations. The control policy exhibits compliant behavior in obstacle-free zones while transitioning to increased damping near obsta-

cles. Applied to a collaborative robot arm demonstrates superior collision rejection capabilities compared to the existing methods.
Finally, the obstacle method was extended to control holonomic multi-agent systems by introducing multiple control points for each volumetric agent. In simulated assistive living environments, the proposed framework was used to reposition furniture agents autonomously.
Overall, the contributions demonstrate the effectiveness and robustness of the obstacle avoidance algorithms in navigating robots in complex and dynamic environments.

Keywords: Robotics; Motion Planning; Obstacle Avoidance; Nonlinear Control; Dynamical Systems: Autonomous Agents; Mobile Robots; Robot Arms; Crowd Navigation; Human-Robot Interaction

Zusammenfassung

Roboter profilieren durch präzise Manöver in statischen Umgebungen gesteuert durch Offline-Bewegungsplanung. Doch außerhalb des Fabrikbodens ist die Welt dynamisch: Menschenmengen bewegen sich ständig, und Bediener möchten sich frei um den Roboter herum bewegen. Dynamische System mit einer geschlossenen Steuerfunktion welche in Echtzeit ausgewertet werden kann, ermöglicht es Robotern, ohne vorherige Planung zu agieren. In dieser Doktorarbeit wird dieser Ansatz verwendet, um ein dynamisches System zu modulieren und kollisionsfreies Verhalten zu erzwingen. Eine analytische Umgebungsrepräsentation verkürzt die Berechnungszeit und ermöglicht die Applikation der Algorithmen in der realen Welt. Die Neuheiten dieser Arbeit sind (1) die Gewährleistung der Konvergenz bei komplexen Hindernissen, (2) die Einführung schneller Kollisionsvermeidung für Sensordaten, (3) die Hindernisvermeidung auf der Grundlage nichtlinearer Dynamik und (4) die Einführung eines kollisionsfreien Drehmomentreglers.

Im ersten Teil der Arbeit werden lineare Dynamiken durch Modulation eingeschränkt. Die Beschreibung von Objekten wird umgekehrt, um den Fluss innerhalb von Wänden zu erzwingen. Wir beweisen, dass der Fluss niemals die Grenzen der Wände berührt und zu einem einzigen Punkt konvergiert. Darüber hinaus erstellen wir glatte Bewegungsfelder um Hindernisse mit Kanten. Die Technik funktioniert in statischen und zeitlich variierenden Umgebungen, die sich bewegen oder verformen. Die Wirksamkeit wird anhand eines autonomen mobilen Roboters in statischen Büroumgebung sowie in simulierten und realen Menschenmengen demonstriert.

Im zweiten Teil bauen wir auf der modulationsbasierten Hindernisvermeidung auf und erweitern sie auf viele Hindernisse und Punktewolkenbeschreibungen der Umgebung. Um die Berechnungskosten zu reduzieren, werden alle Hindernisse und Sensorwerte als ein einziges virtuelles Hindernis interpretiert. Das vorgeschlagene Steuerschema kombiniert eine hochrangige Eingabe mit schneller reaktiver Hindernisvermeidung. Der Ansatz wird experimentell an einem halbautonomen Rollstuhlfahrer im gemeinsamen Steuermodus getestet, der einen belebten Marktplatz im Stadtzentrum überquert.

Der dritte Teil konzentriert sich auf die Erweiterung der Kollisionsvermeidung auf nichtlineare Dynamik und allgemeine konkave Hindernisse. Der vorgeschlagene Algorithmus dreht die anfängliche Dynamik zur Tangentialebene der Hindernisoberfläche. Die Methode lenkt nichtlineare Dynamiken um Hindernisse herum und gewährleistet das Fehlen von lokalen Minima. Die Kombination der Vektorrotation ermöglicht das Bewegen um Graphen sternförmiger Hindernisse. Die experimentelle Validierung verwendet die Methode, um die kollisionsfreie Navigation eines Roboterarms um dynamische, konkave Hindernisse sicherzustellen.

Im vierten Teil der Arbeit wird eine Kraftsteuerung vorgeschlagen, die die Widerstandsfähigkeit

gegenüber Störungen und Geräuschen erhöht. Die gewünschten Kollisionsvermeidung wird mit einem passiven Drehmomentregler gekoppelt, um auf externe Störungen sicher zu reagieren. Die Steuerungsrichtlinie zeigt nachgiebiges Verhalten in hindernisfreien Zonen und wechselt in der Nähe von Hindernissen zu erhöhter Dämpfung. Die Anwendung auf einen kollaborativen Roboterarm zeigt überlegene Fähigkeiten zur Kollisionsvermeidung im Vergleich zu bestehenden Methoden.
Schließlich wurde die Hindernismethode erweitert, um holonome Mehragentensysteme zu steuern, indem für jeden volumetrischen Agenten mehrere Steuerpunkte eingeführt wurden. In simulierten assistiven Wohnumgebungen wurde das vorgeschlagene Framework verwendet, um Möbelagenten autonom umzupositionieren.
Zusammengefasst zeigt diese Arbeit die Wirksamkeit und Robustheit der Kollisionsvermeidung bei der Navigation von Robotern in komplexen und dynamischen Umgebungen.

Kennwörter: Robotik; Bewegungplanung; Kollisionsvermeidung; Objekte Umgehen; Nonlineare Reglung; Dynaische Systeme; Autonome Agenten; Mobile Roboter; Roboter Arm; Navigation in Menschenmengen; Mensch-Roboter Interaktion

Résumé

Les robots excellent dans des manœuvres précises dans des environnements statiques guidés par une planification de mouvement hors ligne. Cependant, le monde est dynamique en dehors du sol de l'usine : les gens dans les foules se déplacent constamment, et les opérateurs veulent travailler librement autour du robot. Le système dynamique utilisant une fonction de contrôle en forme fermée qui peut être évaluée en temps réel permet aux robots de se déplacer sans planification préalable. Cette thèse de doctorat utilise cette approche pour moduler un système dynamique initial afin d'imposer un comportement sans collision. Une représentation analytique de l'environnement accélère le temps de calcul, rendant les algorithmes réalisables pour le monde réel. Les contributions de cette thèse sont les suivantes : (1) garantir la convergence pour des obstacles complexes, (2) introduire une évitement rapide des collisions pour les données des capteurs, (3) éviter les obstacles basés sur la dynamique non linéaire et (4) introduire un régulateur de couple sans collision.

Dans la première partie de la thèse, les dynamiques linéaires sont contraintes en utilisant la modulation des dynamiques. La description de l'évitement des obstacles est inversée pour imposer le flux à l'intérieur des parois. Nous prouvons qu'un tel flux ne touche jamais les limites du volume environnant et converge vers un point unique. De plus, nous créons des champs de mouvement lisses autour des obstacles avec des bords, tels que des tables. La technique évite les environnements statiques et variables dans le temps, qui sont en mouvement ou en déformation. L'efficacité est démontrée sur un robot mobile autonome dans un environnement intérieur complexe statique ainsi que dans des foules simulées et réelles.

Dans la deuxième partie, nous nous appuyons sur l'évitement des obstacles basé sur la modulation et l'étendons à de nombreux obstacles et descriptions de l'environnement en nuage de points. Le coût de calcul est réduit en interprétant tous les obstacles et les lectures des capteurs comme un seul obstacle virtuel. Le schéma de contrôle proposé combine une entrée de haut niveau avec un évitement rapide des obstacles réactif. L'approche est évaluée expérimentalement sur un opérateur de fauteuil roulant semi-autonome en mode de contrôle partagé traversant un marché animé en centre-ville.

La troisième partie se concentre sur l'augmentation de la méthode d'évitement des obstacles pour les dynamiques non linéaires et les obstacles concaves généraux. L'algorithme proposé fait pivoter les dynamiques initiales sur le plan tangent à la surface de l'obstacle. La méthode guide les dynamiques non linéaires autour des obstacles tout en garantissant l'absence de minima locaux. La composition de la rotation vectorielle permet de se déplacer autour de graphes d'obstacles en forme d'étoile. La validation expérimentale a utilisé la méthode pour garantir une navigation sans

collision d'un bras robotique à 7 degrés de liberté autour d'obstacles dynamiques et concaves.
Dans la quatrième partie de la thèse, un contrôle de la force est proposé, ce qui améliore la résilience contre le bruit et les perturbations. Les dynamiques d'évitement des obstacles souhaitées sont couplées à un régulateur passif de couple pour garantir une réponse sûre aux perturbations externes. La politique de contrôle présente un comportement souple dans les zones sans obstacle tout en passant à un amortissement accru près des obstacles. Appliquée à un bras robotique collaboratif, elle démontre des capacités de rejet de collision supérieures par rapport aux méthodes existantes.
Enfin, la méthode des obstacles a été étendue pour contrôler des systèmes multi-agents holonomes en introduisant plusieurs points de contrôle pour chaque agent volumétrique. Dans des environnements de vie assistée simulés, le cadre proposé a été utilisé pour repositionner de manière autonome des agents de mobilier.
Dans l'ensemble, les contributions démontrent l'efficacité et la robustesse des algorithmes d'évitement des obstacles dans la navigation des robots dans des environnements complexes et dynamiques.

Mots Clés : Robotique ; Planification de mouvement ; Évitement d'obstacles ; Contrôle non linéaire ; Systèmes dynamiques ; Agents autonomes ; Robots mobiles ; Bras de robot ; Navigation dans la foule ; Interaction homme-robot.

Contents

Abstract (English/Français/Deutsch) **i**

1 Introduction **1**
1.1 Motivation . . . 3
1.1.1 Controlling Fast and Slow . . . 4
1.1.2 Position, Velocity and Force Control . . . 5
1.2 Robots in Human Environments . . . 6
1.2.1 Consistency . . . 7
1.2.2 Convergence . . . 7
1.2.3 Limitations . . . 8
1.2.4 Safety . . . 9
1.2.5 Stability . . . 9
1.3 Contributions . . . 10
1.3.1 Outline . . . 10
1.4 Thesis Assumptions . . . 11
1.4.1 Assumption 1: Known Robot State . . . 11
1.4.2 Assumption 2: Known Environment . . . 11
1.4.3 Assumption 3: Star World . . . 12
1.4.4 Assumption 4: Point Like Robot . . . 12
1.4.5 Assumption 5: Unicycle Model . . . 12

2 Related Work **13**
2.1 Motion Planning for Obstacle Avoidance . . . 13
2.1.1 Graph Based Path Planner . . . 16
2.1.2 Optimal Control . . . 20
2.1.3 Determinisic Reactive Obstacle Avoidance . . . 25
2.1.4 Data Driven Navigation . . . 31
2.1.5 Collision Control . . . 34
2.1.6 Contribution . . . 36
2.2 Compliant Robot Controller . . . 36
2.2.1 Passive Velocity Controller . . . 36
2.2.2 Stable Interaction Controller . . . 37
2.2.3 Compliance in Telemanipulation . . . 37

2.2.4 Energy Tanks . . . 38
2.2.5 Contribution . . . 38
2.3 Multi-Agent Navigation . . . 38
2.3.1 Contribution . . . 39

3 Preliminaries **41**
3.1 Notation . . . 41
3.2 Dynamical Systems . . . 41
3.2.1 Velocity Vector Field . . . 42
3.2.2 Rigid Body Dynamical Systems . . . 42
3.2.3 Straight Motion . . . 42
3.3 Obstacle Description . . . 43
3.3.1 Star Shaped Obstacles . . . 43
3.3.2 Distance Function . . . 45
3.4 Obstacle Avoidance through Modulation . . . 46
3.4.1 Dynamic Obstacles . . . 47
3.4.2 Multiple Obstacles . . . 48
3.5 Robot Kinematics . . . 49
3.5.1 Forward Kinematics . . . 49
3.5.2 Differential Kinematics . . . 50
3.6 Robot Dynamics . . . 51
3.6.1 Impedance Controller Design . . . 51
3.6.2 Passive Damping Controller . . . 51
3.7 Performance Guarantees . . . 52
3.7.1 Collision Avoidance . . . 52
3.7.2 Lyapunov Analysis . . . 53
3.7.3 Contraction Theory . . . 54
3.7.4 Passivity Analysis . . . 56

4 Developed Tools **57**
4.1 Directional Weighted Mean . . . 57
4.1.1 Pairwise Closest Distance in Direction Space . . . 60
4.1.2 Intersecting Obstacle Descent . . . 61
4.1.3 Closest Distance for Mixed Environments . . . 61
4.2 Vector Rotation . . . 62
4.2.1 Perpendicular Rotation Base . . . 62
4.2.2 Rotating a Vector . . . 63
4.2.3 Weighted Rotation Summing . . . 63
4.2.4 Weighted Rotation Sequence . . . 63
4.2.5 Direction Tree Evaluation . . . 64
4.3 Obstacle Avoidance with Robots Arms . . . 65
4.3.1 Goal Command Towards Attractor . . . 65
4.3.2 Link Avoidance . . . 66

4.3.3 Evaluation Weights for a Robot Arm . . . 67
4.3.4 Evaluation Along the Robot Arm . . . 69
4.3.5 Validation in Simulation . . . 70

5 Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds 73
5.1 Introduction . . . 74
5.2 Obstacle Avoidance . . . 75
5.2.1 Surface Friction Imitation . . . 75
5.2.2 Repulsive Eigenvalue . . . 76
5.3 Inverted Obstacle Avoidance . . . 77
5.3.1 Distance Inversion . . . 77
5.3.2 Modulation Matrix . . . 77
5.3.3 Guiding Reference Point to Pass Wall Gaps . . . 79
5.3.4 Concave Environments . . . 80
5.4 Nonsmooth Surfaces . . . 81
5.4.1 Pseudo Normal Vector . . . 82
5.4.2 Inverted Obstacles . . . 83
5.4.3 Implementation . . . 85
5.5 Dynamic Environments . . . 86
5.5.1 Moving Obstacles . . . 86
5.5.2 Deforming Obstacle . . . 86
5.5.3 Impenetrability with Respect to Maximum Velocity . . . 87
5.5.4 Reference Point Placement . . . 89
5.6 Comparison Algorithms . . . 90
5.6.1 Qualitative Comparison . . . 90
5.6.2 Quantitative Comparison . . . 91
5.6.3 Computational Complexity . . . 93
5.7 Empirical Validation . . . 94
5.7.1 Static Environment . . . 94
5.7.2 Dense Crowd (Simulation) . . . 95
5.7.3 Proof of concept: Outdoor Environment . . . 97
5.8 Discussion . . . 98
5.8.1 Scalability and Speed . . . 101
5.8.2 Contribution . . . 101
5.8.3 Future Work . . . 102

6 Fast Obstacle Avoidance Based on Real-Time Sensing 103
6.1 Introduction . . . 104
6.2 Fast Obstacle Avoidance . . . 104
6.2.1 Single-Modulation Obstacle Avoidance . . . 104
6.2.2 Eigenvalue Matrix . . . 106
6.3 Sample-Based Obstacle Avoidance . . . 108

6.3.1 Reference Summing for Sampled Data . . . 109
6.3.2 Decomposition Matrix . . . 109
6.3.3 Eigenvalue Matrix . . . 110
6.4 Disparate Obstacle Descriptions . . . 114
6.4.1 Fusion Weights of Sampled and Analytic Obstacles . . . 114
6.4.2 Asynchronous Evaluation . . . 115
6.4.3 Uniform Importance Scaling . . . 115
6.5 Experimental Validation . . . 116
6.5.1 Computational Speed . . . 116
6.5.2 Convergence Analysis Using Disparate Sensor Data . . . 116
6.5.3 Evaluation Using a Semi-Autonomous Wheelchair QOLO . . . 117
6.6 Discussion . . . 120
6.6.1 Future Work . . . 121

7 Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics 123
7.1 Introduction . . . 124
7.1.1 Problem Statement . . . 126
7.1.2 Contributions . . . 127
7.2 Obstacle Avoidance through Rotation . . . 127
7.2.1 Vector Rotation for Collision Avoidance . . . 128
7.2.2 Surface Repulsion . . . 133
7.2.3 Inverted Obstacles . . . 134
7.3 General Nonlinear Motion . . . 136
7.3.1 Nonlinear Motion without Stationary Point . . . 136
7.3.2 Nonlinear Motion in the Presence of a Stationary Point . . . 137
7.4 Multi-Obstacle Environments . . . 143
7.4.1 Dynamic Environments . . . 144
7.4.2 Motion within Multiple Enclosing Hulls . . . 144
7.5 General Concave Obstacles . . . 147
7.5.1 Velocity Propagation through Obstacle Tree . . . 147
7.5.2 Convergence Sequence . . . 151
7.5.3 Mapping Weight Normalization . . . 151
7.5.4 Practical Considerations . . . 152
7.6 Evaluation . . . 152
7.6.1 Computational Cost . . . 152
7.6.2 Obstacle Avoidance While Following a Stable Limit Cycle . . . 153
7.6.3 Nonlinear Path Following with Autonomous Wheelchair . . . 155
7.6.4 Obstacle Avoidance in Three Dimensions . . . 158
7.7 Discussion . . . 160
7.7.1 Stationary Points . . . 161
7.7.2 Trees of Stars . . . 161
7.7.3 Higher Dimensional Applications: Joint Limits . . . 162

7.7.4 Application to Robotic Systems . . . 162
7.7.5 Tangent Following on the Surface . . . 162
7.7.6 Region of Influence during Task Obstruction . . . 163

8 Obstacle Aware Passive Control for Dynamical Systems 165
8.1 Introduction . . . 166
8.1.1 Contribution . . . 167
8.1.2 Problem Statement . . . 167
8.2 Obstacle Aware Passivity . . . 168
8.2.1 Damping for Collision Repulsion . . . 170
8.2.2 Damping for Velocity Preservation . . . 172
8.2.3 Cluttered Environments . . . 172
8.2.4 Damping Parameter Design . . . 172
8.2.5 Passivity Analysis . . . 173
8.3 Discrete Time Behavior . . . 176
8.4 Evaluation . . . 179
8.4.1 Qualitative Repulse Rejection . . . 179
8.4.2 Noise Analysis . . . 181
8.4.3 Obstacle Aware Passivity Using a Robot Arm . . . 184
8.5 Discussion . . . 186
8.5.1 Discrete Control Convergence . . . 186
8.5.2 Applicability and Theoretical Analysis . . . 187
8.5.3 Point Mass Agents . . . 187
8.5.4 Caution in Obstacle's Proximity . . . 187

9 Conclusion 189
9.1 Contributions . . . 189
9.2 Limitations and Future Work . . . 190
9.2.1 System Simplification . . . 190
9.2.2 Extension Beyond Obstacle Avoidance . . . 191
9.2.3 Combination of Perception and Control . . . 191
9.2.4 From Local Avoidance to Global Planning . . . 192
9.3 Final Thoughts . . . 192
9.3.1 Extensive Field . . . 192
9.3.2 Problems Become Harder . . . 193
9.3.3 Co-Design of Systems . . . 194
9.3.4 Ethical Deployment of Robotics . . . 195

A Dynamical System Modulation for Decentralized Collision-Free Coordination of a Swarms 197
A.1 Introdcution . . . 199
A.1.1 Contributions . . . 201

A.1.2 Notation . . . 202
A.2 Agent Control Using Multiple Control Points . . . 202
A.2.1 Optimal Control Points . . . 202
A.2.2 Dynamic Weights . . . 204
A.3 Dynamic Attractor Motion . . . 205
A.3.1 Adaptive Repulsive Field . . . 205
A.3.2 Parking Zones . . . 206
A.4 Prioritization of the Agent . . . 207
A.5 Soft Linear/Angular Control Decoupling . . . 208
A.6 Minimization of The Virtual Drag . . . 209
A.6.1 Drag Factor . . . 209
A.6.2 Drag Angle . . . 209
A.6.3 Convergence to Attractor Orientation . . . 210
A.7 Safety Module . . . 210
A.8 Results . . . 212
A.8.1 Collision Avoidance with Human Disturbance . . . 212
A.8.2 Agent Priority . . . 213
A.8.3 Drag Minimization . . . 213
A.8.4 Quantitative Comparison . . . 215
A.9 Conclusion . . . 217
A.9.1 Future Work . . . 219

B Student Projects **221**

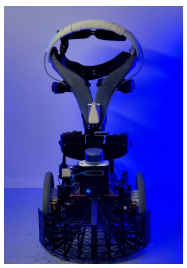
Curriculum Vitae **243**

1 Introduction

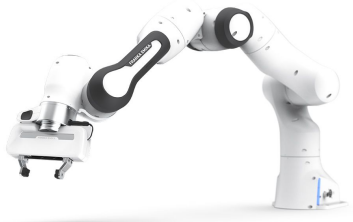
The Oxford Dictionary[1] defines the term *robot* as:

robot: *"a machine controlled by a computer that is used to perform jobs automatically"*

As general as in the definition, robots come with remarkable diversity in form, function, and operational environments. Each is tailored to specific applications, environments, and challenges, yet they remain flexible to perform various tasks. For instance, we encounter autonomous vehicles efficiently delivering products, robot arms tirelessly working in factories assembling products, and flying drones skillfully inspecting bridges and buildings. Furthermore, this definition of robots can be expanded to include semi-autonomous systems. These might involve robot drones guided by users while still maintaining control for flight stabilization or semi-autonomous wheelchairs enabling users to specify desired motions while the onboard control aids in obstacle avoidance (a selection of robots in Figure 1.1).



(a) Semi-autonomous wheelchair (mobile robot)



(b) Collaborative robot arm (Emika, 2023)

(c) Flying drone (Flyability, 2023)

Figure 1.1: Robots can have various forms, applications, and levels of autonomy. Common robots are mobile robots, robot arms, and flying robots.

What truly sets robots apart from general machines is their versatility and adaptability. Unlike traditional machines like conveyor belts, which can only transport items in predefined directions, robots, especially mobile ones, possess the ability to navigate and operate within dynamic and

[1] https://www.oxfordlearnersdictionaries.com/, 05.09.2023

unstructured environments flexibly. While a robot arm can autonomously perform many tasks, a forklift, for example, necessitates constant human supervision and control. This adaptability, however, brings forth substantial challenges in robot control. Due to their ability to function in human-like environments, robots must continuously adapt to changing conditions. On one hand, they must recognize and understand changes in their surroundings and how they affect their tasks. On the other hand, they must ensure their environment and their safety.

Since the first industrial robot was introduced in 1961 (Kurfess et al., 2005), robots were often deployed in segregated workspaces, frequently enclosed within safety cages. This segregation ensured complete control over all environmental parameters, from the precise position where a product was placed to the positioning of obstacles and the absence of humans during operation. Consequently, software development for these robots was relatively straightforward. The programs could be created before task execution, often involving point-to-point motion programmed by an operator. In this paradigm, the intelligence behind motion planning predominantly stemmed from human operators, and the robot operated more similar to a mechanical device than an autonomous agent.

However, advancements in computational power and the affordability of sensors have enabled a new era of smart robotics. Vision systems now actively detect the positions of incoming objects on conveyor belts, allowing for dynamic and adaptable packaging. Information sharing between robots and machines enables robots to react autonomously or feed manufacturing machines. Outside factory settings, we have witnessed the deployment of autonomous robots such as vacuum cleaners navigating cluttered living rooms and drones autonomously delivering packages to remote areas. Nevertheless, these developments remain incremental, with autonomous robots often requiring human intervention in complex environments or bad weather conditions.

Collaborative robots, commonly referred to as Cobots, represent a significant leap forward in the deployment and interaction of robots (Bischoff et al., 2010). Cobots are equipped with torque sensors in each joint, enabling them to sense and adapt to external forces directly. This capability enables collaborative robots to promptly detect and safely share their workspace with human operators (Standard, 2016). Furthermore, the potential for kinesthetic interaction with these robots opens up avenues for physical collaboration, such as kinesthetic teaching by demonstration and collaborative task sharing, for example, lifting heavy objects. However, in practice, most collaborative robot applications are not direct cooperation but coexistence or workspace sharing (Bauer et al., 2016). In such scenarios, humans and robots do not simultaneously work on the same task but proceed independently. In this context, it is paramount for the robot to stick to its intended motion while safely avoiding the human operator. This not only minimizes the risk of accidents but also reduces disruptions to workflow efficiency.

This thesis aims to advance the efficiency and safety of human-robot interaction. This is achieved by introducing obstacle avoidance algorithms with broad applicability. These algorithms enable robots to precisely execute their tasks while minimizing disruption to the environment. Foremost, the algorithms prioritize safety, ensuring that collisions are avoided. Simultaneously, the methods

must operate at a low computational cost, enabling them to adapt to changing environmental conditions. This capacity for flexibility and reactive behavior is paramount, as it equips robots with the tools needed to venture out of the cages into the real, human-shared world.

1.1 Motivation

Classical machines excel in high-volume production, where they prove remarkably cost-effective. They are designed and optimized for a specific task, which they execute tirelessly around the clock (Figure 1.2). However, their rigidity becomes apparent when faced with tasks outside their predefined capabilities. In contrast, humans exhibit exceptional flexibility. They can easily adapt to new tasks within their skill range. Therefore, manufacturing a product by hand is often the most efficient approach for small production volumes or prototyping scenarios. Robots, on the other hand, occupy an intermediary position between classical machines and human adaptability. They are often capable of performing a variety of tasks. However, the programming and adaptation of robots and their environments entail substantial costs, typically necessitating significant production volume to justify the investment. Human-robot interaction, particularly Cobots, offers greater flexibility than full robotization. In such scenarios, humans can assist robots, enhancing the system's adaptability. However, this approach also comes with a higher cost per unit due to the added human element.

Despite the advantages of robots in higher production volumes, humans still outperform them in terms of agility and adaptability. This inherent human capability challenges deploying robots in specific industries and everyday life. For instance, with its soft fabrics, the clothing industry brings subtle variability to each task execution. Even in car manufacturing, known for its extensive robotization, many intricate tasks necessitate human involvement, as full automation would be impractical with currently available technology.

This thesis aspires to make human-robot collaboration more widely applicable. This involves implementing safe and reactive collision avoidance strategies. As a result, human-robot collaboration becomes applicable in scenarios where human labor was previously the sole effective option. Additionally, we aspire to enhance the reliability of motion planning algorithms, showing their precise applicability and ensuring stability and convergence. By reducing the need for continuous human oversight and interaction while still allowing for shared workspaces, we aim to make human-robot interaction more efficient. This shift facilitates the deployment of robots in areas where robots were previously confined behind fences.

While manufacturing is the most prominent use case to illustrate the benefits of robotization, the implications extend far beyond this domain. More autonomous robots with enhanced obstacle avoidance capabilities become more viable for everyday applications as this simplifies the robot deployment and usability. One can envision a robot arm assisting a paralyzed individual with daily tasks, shared-control wheelchairs facilitating collision avoidance in crowded spaces, or robots performing extravehicular activities for maintenance in outer space. Of course, some tasks

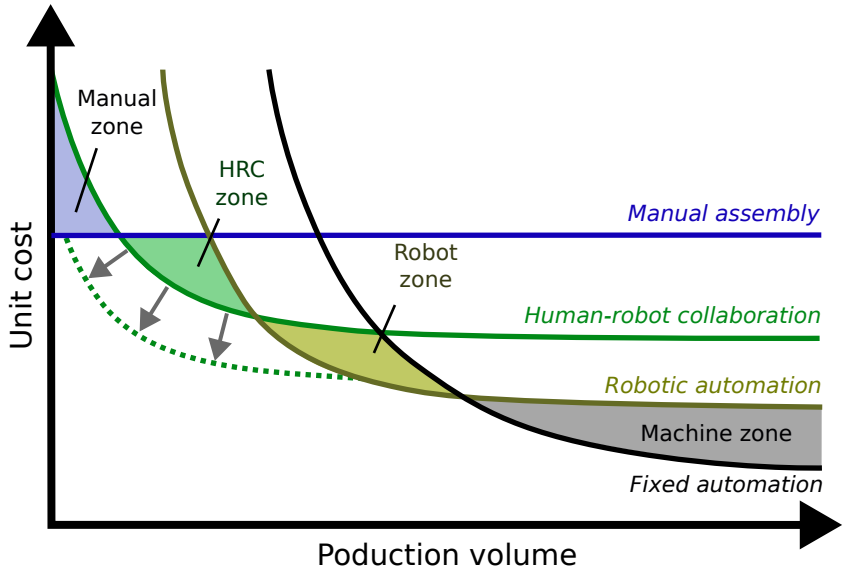


Figure 1.2: Unit economics versus flexibility gives information about the favorable production paradigm (image adapted from Matthias, Ding, and Miegel, 2013). This work contributes to making cobots cheaper to deploy and more flexible (gray arrows).

are inherently more relevant to robotization due to environmental factors, such as *dirty tasks* as cleaning nuclear waste or *dangerous tasks* as operating in near-vacuum conditions. Conversely, other *dull* tasks, such as high-volume manufacturing processes, like clothing manufacturing with soft materials or intricate mechanical watchmaking demanding high dexterity, will remain challenging to automate fully.

1.1.1 Controlling Fast and Slow

The ability to navigate and avoid obstacles is a fundamental skill human brains exhibit and a capability most animals share. Whether an octopus seamlessly coordinating its eight arms, a person reaching for the coffee jar hidden in a cluttered cupboard, or a flock of birds executing intricate maneuvers, the natural world showcases effortless collision-free motion. However, achieving such safe and efficient behavior for robots remains a difficult challenge. The problems of finding the shortest path and dynamically planning motion have been shown to be of NP-hard complexity (J. Canny, 1988). Consequently, all practical motion planning algorithms must rely on some form of heuristics, be it through simplifications of the environment or by tolerating longer computation times.

Like humans, robots require more time to compute complex motions than simpler tasks. Just as we take a few seconds to ponder a challenging problem but can instinctively address straightforward issues (Kahneman, 2017), robots face a similar trade-off. Planning the optimal route through an unfamiliar city or arranging multiple jars in a cluttered cupboard involves seconds

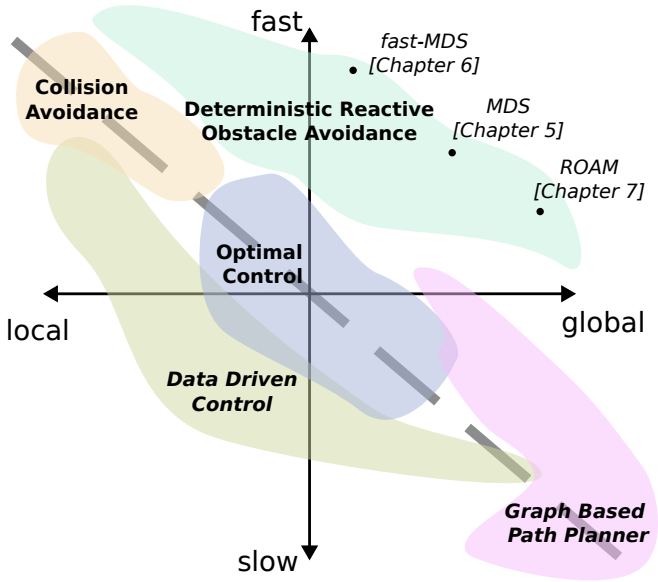


Figure 1.3: The first axis distinguishes between global convergence versus only local collision avoidance capabilities, while the second axis evaluates the algorithms regarding computational speed. Generally, the faster the algorithm, the more local it becomes, as further precise in Chapter 2, and hence, most algorithms are close to the inversely proportional dashed line. The new contributions are faster and can handle more global environments, extending obstacle avoidance to the top right corner.

of thinking. Potential path options must be explored, and potential improvements must be considered. Conversely, an initially planned path can be adjusted swiftly when encountering new obstacles, such as someone crossing our path or instinctively avoiding an incoming ball in a game of dodgeball.

In robotics, global path-planning algorithms designed for broad applications are often slower and cannot operate in real-time. Conversely, fast algorithms that allow rapid evaluation and replanning are typically limited to local convergence and cannot produce complex motion behaviors. As a result, most algorithms find themselves positioned close to the hypothetical line where path length and planning speed are inversely proportional (Figure 1.3). This work enhances obstacle avoidance by employing advanced heuristics and environmental simplifications. We leverage local avoidance algorithms and extend their applicability to more global scenarios, facilitating the navigation of complex environments.

1.1.2 Position, Velocity and Force Control

Our primary focus is on position when planning and executing motions. We think about reaching a specific restaurant in town, identifying it as a meeting point, or picking up the salt shaker at a certain spot on the table. However, reaching these positions involves complex control, including

controlling velocity and applying force effectively throughout the path. Estimating velocity is relatively straightforward, for example, when driving a car, as we are constantly exposed to the speedometer. But, in other situations, such as predicting the velocity of our hand, it becomes significantly more challenging to estimate the exact velocity. Furthermore, initiating motion requires the application of force, resulting in acceleration. Acceleration is essential for speeding up, slowing down, and changing the direction of motion. Describing this acceleration comprehensively for various positions and velocities can be quite unintuitive.

However, many Cobots are equipped with torque sensors, and safety is ensured by limiting the torque it can exert on the environment. Since the surrounding is often challenging to observe fully, accurately determining the precise position of all obstacles simultaneously is mostly infeasible. Yet, Cobots equipped with force and torque sensors provide real-time updates on any interaction torque, and by limiting maximum force, safe navigation is ensured. As a result, maximum safety and flexibility can be achieved by controlling Cobots with a desired control force.

Given the complexity of obtaining the desired acceleration (or force), a cascading evaluation approach is often employed, as visualized in Figure 1.4. The ideal velocity is calculated to reach a desired position (attractor). However, this velocity must be adapted to ensure collision-free motion to account for unexpected interactions. Finally, the torque command for the robot's motors is evaluated to achieve safe torque.

In this work, we contribute to improved collision-free motion generation (Chapter 5, 6, 7) and a novel torque controller for robot arms to execute collision-aware motions (Chapter 8).

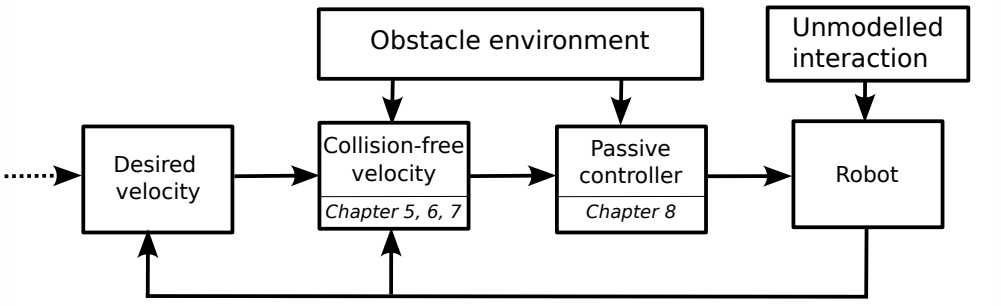


Figure 1.4: The desired velocity can be derived from a learned velocity field or oriented toward a desired attractor. This velocity is then used to calculate the obstacle avoidance velocity, which serves as input for the force controller to determine the control force.

1.2 Robots in Human Environments

The Oxford Dictionary offers a second definition for the term *robot*:

robot: *"(especially in stories) a machine that is made to look like a human and that can do some things that a human can do"*

This definition underscores the historical fascination with replicating humans through machines, a concept deeply rooted in our stories and imagination. It reflects a desire to play the role of creators, crafting machines in our image. However, this definition also encapsulates our ultimate expectations of robots. While we have already engineered machines and robots that outperform humans regarding reliability and efficiency, we often aspire for robots to coexist seamlessly in environments designed for humans. They should be versatile and adaptable, capable of performing tasks like humans do.

In robotics research, there is often an inclination to learn from humans (Ravichandar et al., 2020). Humans exhibit remarkable abilities, from intricate grasping to abstracting their environment and spatial path planning. Researchers frequently attempt to deconstruct these capabilities, seeking to replicate them in robotic systems. Compared to open-loop position-controlled machines and industrial robots, cobots are often designed with human-like closed-loop torque control, closely mimicking how humans move their limbs.

However, the full integration of robots into everyday life may still be decades away, primarily due to the significant differences between how humans live and how robots are programmed. Rather than solely drawing inspiration from human abilities, we want to take a step back and examine the fundamental distinctions between humans and robots. From this perspective, we can derive specific requirements that shape robotic control algorithms. It is worth noting that some of these requirements pertain specifically to the local obstacle avoidance algorithms developed in this thesis and may not directly apply to all areas of robotics.

1.2.1 Consistency

The human brain, while remarkably powerful, is not without its faults. We have all experienced moments when we forget items from our shopping lists once in the store or struggle to find our way back from a restaurant in a new city despite walking there hours earlier.

However, such forgetfulness and unpredictability are deemed unacceptable in robotics. Imagine the frustration, if a factory robot suddenly forgot how to move around a specific obstacle one morning, and collided with it instead. Or if a shared controller for a wheelchair behaved differently each time it was used, and the operator cannot be sure if it will avoid the other pedestrians in the crowd. Therefore, there is a strong desire for deterministic behavior regarding robots.

Requirement: *A*n algorithm must produce predictable and consistent outputs.

1.2.2 Convergence

Humans can engage in abstract thinking, allowing us to adapt to challenges by reevaluating situations and devising creative solutions. In the face of unknown situations, we can analyze our circumstances, refer to maps, or seek advice, often leading to successful problem-solving, even

in novel scenarios.

On the contrary, robots lack this abstract understanding of the world. Nevertheless, robots can switch between algorithms, such as following a globally planned path, but then deform it slightly if required. However, as environments become more complex, these methods can result in a stop-and-go behavior pattern. This behavior is particularly disadvantageous in highly dynamic environments, such as crowded spaces. The impressive output of large language models (LLMs) and similar intelligent systems can sometimes mislead us into thinking, that robots and machines possess comprehensive understanding and reasoning capabilities. However, these models lack the capacity for coherent abstractions and differentiated situation analysis. Their proficiency arises from extensive datasets, enabling them to generate a range of cohesive and articulate outputs. Yet, these models struggle with abstract thinking and often stumble in basic tasks or maintain logical coherence over extended time.

Given the difficulty of switching between behaviors in robots, it becomes crucial for a robot to ensure that it consistently achieves a goal, in obstacle avoidance, this is often specified as reaching an attractor or avoiding local minima. This requirement can be summarized as follows:

Requirement: *A*n algorithm must ensure the absence of local minima or provide guarantees for convergence to the goal state.

1.2.3 Limitations

The human body is equipped with remarkable redundancies and adaptability. For instance, when reaching for a pen on a table, you can stretch your arm to obtain it. If the path is obstructed, you may need to move your shoulder or stand up to reach the pen. Moreover, humans can adjust the number of fingers used for grasping, utilizing two or three fingers for one task and all five fingers or even both hands when necessary. Regarding collision sensing, the human skin hosts an estimated quarter of a million tactile sensors (Corniani and Saal, 2020). Furthermore, the human body exhibits the extraordinary capability to adapt and improve itself based on the demands of a specific task. Over time, the skin develops calluses in response to repeated friction, and muscles grow and enhance their internal coordination to better withstand repeated physical strain.

In contrast, robots lack these redundancies and adaptabilities. Once built and installed, a robot typically performs a specific task. Due to cost constraints, robots are often designed with a focus on their intended function and possess limited redundancies. Over time, mechanical systems, usually composed primarily of metallic components, may degrade and require active maintenance. Consequently, robots have clear limitations regarding the tasks they can perform. Moreover, hardware components have inherent reachability, velocity, and acceleration limitations. Combined with the deterministic behavior of computer chips, it allows us to determine which environments can be handled. The algorithm's scope can be explicitly defined based on these hardware limitations.

Requirement: *A*n algorithm's operational limits must be clearly defined.

1.2.4 Safety

While the human skeleton provides a relatively rigid structure, our bodies have a soft outer layer of biological tissues. This tissue serves as an effective safety measure when interacting with the environment. In the event of a collision, this soft tissue acts as a damping mechanism, dispersing the impact forces over a larger area to minimize localized damage. However, the soft outer layer of human bodies also makes us susceptible to injury. Most people have experienced wounds or tissue damage at some point. Yet, the remarkable self-repair capabilities of the human body allow it to maintain this flexible structure.

In contrast, robots cannot self-heal; many are constructed with rigid shells to reduce susceptibility to damage. While soft robotics research is actively exploring materials and designs that replicate human-like flexibility (C. Lee et al., 2017), such materials can be fragile may pose challenges for reliable robot control.

Consequently, robot motion must be carefully planned, and interactions with the environment must be flawlessly controlled. A collision between robots on the environment must be averted at all costs, as it can inflict major damage.

Requirement: *A*n algorithm must ensure safety by preventing collisions with the environment.

1.2.5 Stability

Like most biological systems, human bodies are inherently designed for energy efficiency. Throughout evolution and most of human history, energy was often the scarcest resource available (Smil, 2018). Consequently, the human body, in conjunction with the nervous system, has evolved to minimize energy consumption, resulting in the stable control of a healthy body.

During the Industrial Revolution, there was a significant shift as energy suddenly became relatively abundant due to the widespread use of fossil fuels. This era also witnessed the development of machines and, subsequently, in the following century, robots. In this context, the limiting factor for robots was primarily their cost, which was connected to the human labor expended during design and production, rather than concerns about energy consumption.

Today's robots typically have access to ample power resources, thanks to their connection to the electric grid or the availability of energy-dense batteries. Consequently, if a robot's controller becomes unstable, it can lead to excessive power draw, potentially causing damage to the hardware.

Requirement: *C*ontrol algorithms must ensure stability to prevent energy wastage and potential hardware damage.

1.3 Contributions

This thesis contributes to the closed-form obstacle avoidance algorithms in dynamic environments. The achievements can be summarized as follows:

- Development of a modulation-based avoidance algorithm tailored for indoor navigation.
- Introduction of a technique utilizing raw distance sensor input to enhance the collision-free avoidance velocity.
- Creation of a reactive navigation algorithm for general environments through vector rotation and space mappings.
- Proposal of a passive force control algorithm designed to reduce collision probabilities with obstacles.
- Extension and application of collision avoidance concepts to volumetric furniture swarms.

All these developments are accompanied by appropriate mathematical proofs and experimental evaluations using either a mobile robot or a collaborative robot arm.

1.3.1 Outline

- **Related Work** (Chapter 2) This chapter provides a comprehensive analysis of the field of motion planning, with a focus on closely related algorithms designed for obstacle avoidance in fast-moving environments.
- **Preliminaries** (Chapter 3) This chapter offers a brief introduction to the mathematical background that underpins the approaches developed in the thesis.
- **Developed Tools** (Chapter 4) This section presents mathematical and algorithmic tools and concepts developed throughout the thesis, with applications that extend beyond specific chapters. The tools encompass angular methods for faster rotation and gradient descent, as well as the usage of control points for obstacle avoidance with robotic arms.
- **Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds** (Chapter 5) This chapter explores obstacle avoidance in human environments, addressing how robots can navigate rooms, circumvent obstacles with sharp corners, and adapt to dynamically changing obstacle shapes.
- **Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics** (Chapter 6) Focusing on scenarios where robotic systems receive real-time sensor data in an unstructured format, this chapter introduces a solution for avoiding point clouds obtained from sensor inputs.

- **Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics** (Chapter 7) This chapter outlines a method for achieving obstacle rotation in initially nonlinear dynamics using local vector rotation and stereographic projection. The algorithm ensures the absence of minima on the obstacle surface and offers continuous behavior across space.

- **Obstacle Aware Passive Control for Dynamical Systems** (Chapter 8) This chapter introduces a torque controller that uses desired position, velocity, and obstacle distribution to apply suitable commands to the robot. This approach allows for disturbance rejection near obstacles while maintaining convergent dynamics at greater distances.

- **Conclusion** (Chapter 9) The concluding chapter summarizes the key findings and contributions of the thesis, highlighting unsolved problems and suggesting directions for future research.

- **Appendix - Dynamical System Modulation for Decentralized Collision-Free Coordination of a Swarms** (Chapter A) The appendix provides additional insights into applying obstacle avoidance algorithms to volumetric agents. It also delves into high-level state machines and agent prioritization in multi-agent scenarios. The algorithm's evaluation in simulation with multiple autonomous furniture interacting while being disturbed by an external agent.

In summary, this thesis explores advanced obstacle avoidance algorithms with practical applications in dynamic environments.

1.4 Thesis Assumptions

A set of assumptions was made for the successful and coherent development of the navigation algorithm presented in this thesis.

1.4.1 Assumption 1: Known Robot State

The algorithms assume that the robot's state and velocity are always known. The robot is expected to be either directly velocity-controlled or coupled with a force controller capable of accurately tracking the velocity, rendering any drift negligible. Achieving the latter assumes precise knowledge of the robot's dynamic model. Robots with articulated joints, such as robot arms, require precise joint position information and limb inertia estimates.

1.4.2 Assumption 2: Known Environment

The algorithms assume full knowledge of the environment, including the shapes and positions of obstacles. This work assumes that the environment's state is obtained from detection systems

such as cameras, LiDAR, infrared measurement systems, or a combination thereof, which provide accurate real-time data.

1.4.3 Assumption 3: Star World

This research focuses on environments composed of star-shaped obstacles and hulls. Star shapes are defined as shapes that have a center point from which every line extending to the shape's surface intersects the surface exactly once (detailed definition in Chapter 3). The development during this thesis extends to tree structures of stars, thus allowing for more complex obstacles that can be represented as finite unions of star shapes. For example, a table can be seen as a composition of its main surface and four legs, all representing star shapes.

1.4.4 Assumption 4: Point Like Robot

The planning process in this research operates in the state or configuration space, where each point represents a unique configuration of the robot denoted as $\vec{\xi}$. Any information about the robot's volume has to be encoded in the obstacle shape.

1.4.5 Assumption 5: Unicycle Model

The research assumes that the robot's motion constraints align with the unicycle model. This implies that the robot has a turning radius greater than zero. Consequently, the desired dynamics must be smooth and differentiable, specifically C1-smooth. This assumption holds for higher dimensions ($N > 2$), where both input and output dynamics must be C1-smooth.

These assumptions provide the foundation for the developed algorithms and approaches, ensuring their applicability to specific robotic scenarios.

2 Related Work

This thesis encompasses diverse developments in the field of robotics, broadly categorized into novel approaches for robot control for obstacle avoidance. The Chapters 5, 6, and 7, propose new strategies for motion planning, which outputs a collision-free velocity around obstacles. In addition, Chapter 8 introduces an obstacle-aware impedance controller designed for force-controlled robots. Furthermore, Chapter A adapts the obstacle avoidance algorithms for applications in volumetric swarms.

This section aims to provide an overview of existing work in the literature related to these topics to contextualize the contributions of this thesis.

2.1 Motion Planning for Obstacle Avoidance

The obstacle and collision avoidance challenge are important topics in robotics with many proposed solutions (Siciliano and Khatib, 2008; Lynch and F. C. Park, 2017). These methods can be categorized based on various factors, including computational complexity, local or global considerations, and determinism versus probabilistic.

This section aims to provide a comprehensive overview of the prevalent methods of obstacle avoidance summarized in Figure 2.1, encompassing established techniques and more recent advancements. We divide the related work of obstacle avoidance into five categories:

- **Graph Based Path Planner** (Sec. 2.1.1 rely on graph exploration to derive a collision-free motion. Sampling-based methods such as *Rapidly-Exploring Random Trees (RRT)* and *Probabilistic Roadmap (PRM)* in global path exploration structure the local collision-free path segments into trees that facilitate optimal path searching. *Space-Time Sampling* includes the time in addition to the special dimension in the path search to address the sequential nature of complex tasks. Alternatively, some graph-based path planners use space partitioning through structured *Grid Cell Decomposition (GCD)* or an *Unstructured Cell Decomposition*. Where the collision-free cells are used to create a motion graph.

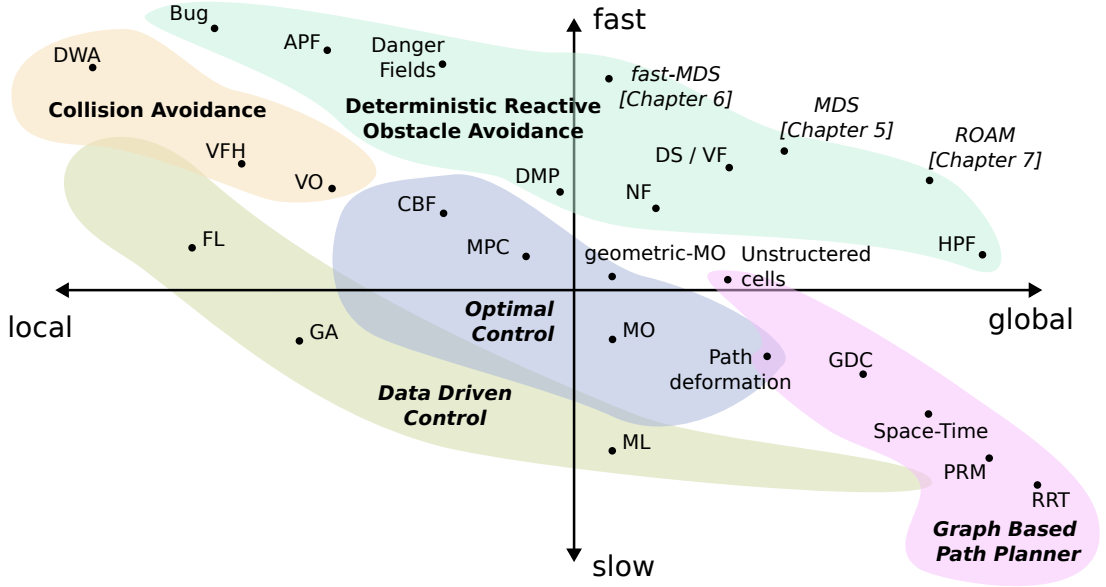


Figure 2.1: The first axis distinguishes between global convergence and local collision avoidance capabilities, while the second axis assesses the algorithms in terms of computational speed. The algorithms are detailed in Section 2. Collision-free motion generation is classified into five primary categories. Overall, they demonstrate an inverse relationship between global applicability and computational speed. It is important to note that this graphical representation simplifies the broad landscape of approaches to provide an overview. Note that while the presented algorithm comprises numerous variations and specific functionalities, they are simplified in this visualization to provide a general overview.

Notably, a common characteristic among these algorithms is their computationally intensive graph construction process, necessitating offline planning. Nevertheless, some of the modern variants leverage vector fields, enabling real-time adaptation.

- **Optimal Control** (Sec. 2.1.2 involves solving a minimization problem to find (local) optimal paths. This methodology finds utility enables *Local Path Deformation*, where it refines globally sampled solutions to avoid collisions in dynamic environments. Increased computational capabilities have enabled *Global Motion Optimization (MO)* to find an optimal path to a goal in cluttered environments, *Geometric Methods* have enabled reducing computational cost and finding better paths. The increased computational capabilities of onboard computers have facilitated real-time evaluation of limited horizon optimization problems such as *Model Predictive Control (MPC)*. This method has been further enhanced using *Control Barrier Functions (CBF)* avoiding boundaries such as obstacles.

- **Determnistic Reactive Obstacle Avoidance** (Sec. 2.1.3 provides a robotics control that evaluates the desired motion in real-time. This domain includes simple yet reliable control strategies, including the *Bug algorithm*, which selects a predefined avoidance direction, and *Artificial Potential Fields (APF)*, which actively exert forces to repel the robot from

obstacles. Similarly, *Danger Fields* invert the APF and calculate the force that pushes the obstacles away from the robot. While these fundamental control laws offer stability, they are susceptible to local minima. This challenge can be addressed through the *Navigation Functions (NF)* or AFP based on *Harmonic Potential Functions (HPF)*.

These control laws are frequently formulated as *Dynamical Systems (DS)* that describes the desired system evolution or as a *Vector Fields (VF)* which encodes the desired velocity field. Contrarily, *Dynamic Motion Primitives (DMP)* introduce a temporal transition between different motion patterns to ensure convergence to the goal.

This thesis contributes to the deterministic algorithm domain with novel developments, including the Dynamical System-based Modulation (DSM) for various environments discussed in Chapter 5. This approach is accelerated by applying the algorithm to unstructured sensor data in the fast-DSM, presented in Chapter 6. Additionally, enhanced convergence in complex environments is achieved in the Rotational Obstacle Avoidance Algorithm (ROAM), detailed in Chapter 7.

- **Collision Control** (Sec. 2.1.5) focuses on safeguarding the robot from collisions rather than converging to a goal. The *Dynamic Window Approach (DWA)* actively ensures the robot's path remains free of collisions within its immediate, dynamically adjusted window. Similarly, the *Vector Field Histogram (VFH)* observes the robot's surroundings in a radial direction, particularly suitable for mobile robots. *Velocity Obstacles (VO)* operate by analyzing collisions in the velocity space, ensuring collision-free navigation within complex dynamic environments. These techniques are particularly suited to ensuring collision checks online, in shared control setups, or in the presence of a global navigation strategy.

- **Data Driven Navigation** (Sec. 2.1.4) does often not explicitly encode obstacle avoidance knowledge but instead derives it implicitly from data. *Machine Learning (ML*) methods are the most commonly used. They find applications across a wide spectrum, ranging from estimating distance to obstacles to approximating globally learned trajectories. *Fuzzy logic (FL)* incorporated expert judgment data for binary decision-making to create an avoidance motion. Conversely, *Genetic Algorithms (GA)* iteratively create obstacle avoidance trajectories, based on choosing the most optimal among many sampled solutions. Data-driven approaches offer the advantage of adaptability to various scenarios, due to ease of implementation, and adaptability. However, it's important to note that their heavy reliance on data and generalizability can often result in increased computational demands compared to specialized algorithms, and a lack of avoidance guarantees.

The algorithms above are further detailed in the remainder of this section.

2.1.1 Graph Based Path Planner

Navigating within complex environments is complicated, as it is nearly impossible to anticipate all potential scenarios. Sampling methods are pivotal in identifying collision-free points and sub-paths within the space. These components are then integrated to form a graph, which is used to evaluate a cohesive global path. This approach effectively sidesteps the need to solve an intricate global control problem. Convergence of most graph-planning algorithms to a feasible path is assured over an infinite time. However, this results from an infinitely high resolution, which can be achieved. For a more comprehensive exploration of sampling-based methods, the readers can refer to (Elbanhawi and Simic, 2014; Kingston, Moll, and L. E. Kavraki, 2018).

Grid Cell Decomposition

Grid Cell Decomposition (GCD) offers a systematic way of partitioning space by dividing the environment into smaller regions called grid cells. Each cell is then categorized as obstacle-free or occupied (Elfes, 1989). This structural division forms the basis for generating viable trajectories, navigating through the obstacle-free cells. The grid is treated as a graph; the A* algorithm is then employed to efficiently determine the shortest path from the starting to the goal cell (Hart, Nilsson, and Raphael, 1968). GCD has a computational complexity of $\mathcal{O}(G^N)$, where N is the dimensionality of the space and G represents the grid resolution along each dimension. While reducing cell size improves path precision, it increases the computational demands, rendering the algorithm unsuitable for dynamic and cluttered environments (Acar et al., 2002). To address this, dynamic grid spacing techniques adjust the global resolution G while exploring space to retain local precision while having a low global resolution. GCD is often coupled with hierarchical tree-based graph search approaches to ensure efficient computation ("Multiresolution path planning for mobile robots" 1986). Nonetheless, due to the inherent inefficiencies in scaling with the number of dimensions N, GCD primarily finds its stronghold in guiding mobile robots through two-dimensional spaces (Galceran and Carreras, 2013). GCD can be directly populated for mobile robots through onboard sensory measurements such as LiDAR or laser scans. Beyond providing grid cell decomposition data, this sensory information aids in real-time environment map update (Durrant-Whyte and Bailey, 2006). The creation and refinement of environment maps are often facilitated through Simultaneous Localization and Mapping (SLAM) techniques (Bailey and Durrant-Whyte, 2006).

As GCD offers a structured representation of the collision-free environment, it is often integrated with various global planning strategies, optimization methodologies (Ajeil et al., 2020), and reinforcement learning paradigms (Kyaw et al., 2020). Furthermore, GCD serves as the foundational building blocks for more advanced techniques like the Dynamics Window Approach and Vectorfield Histograms, which will be elaborated upon in the subsequent sections.

Unstructured Cell Decomposition

Unstructured cell decomposition can be performed using Voronoi Diagrams. The two-dimensional space is divided into convex regions known as Voronoi Cells. Each cell houses a single point, with boundaries equidistant from their respective centers (Aurenhammer, 1991). Extending this concept, the Generalized Voronoi Diagram (VGD) introduces the notion of obstacle inclusion, where each cell encapsulates an obstacle, and the cell boundary is equidistant to the surface. As a result, the boundary of a VGD represents the optimal clearance path between two obstacles, paving the way for collision-free navigation along its edges (Ó'Dúnlaing and Yap, 1985). The VGD can be dynamically constructed through continuous space exploration (Rao, Stoltzfus, and Iyengar, 1991). Voronoi Graphs can be constructed in higher dimensional space, maintaining one-dimensional graph edges. This allows for fast optimal path searches despite the increasing complexity of the diagram construction (Choset and Burdick, 2000). The adaptability of Voronoi structures further comes to light through real-time updates that enable their application in dynamic environments (Bhattacharya and Gavrilova, 2008).

Voronoi Diagrams are used in multi-agent deployment and motion planning as they inherently offer insights into robot coverage. The Voronoi Cells are constructed to inhabit one agent each (Schwager, Rus, and J.-J. Slotine, 2011). Leveraging power diagrams and generalized Voronoi diagrams with different cell weights facilitates the control of fully actuated disk robot swarms. This enables simulated robots of varied sizes to undertake complex rearrangements. (Arslan and D. E. Koditschek, 2016b). An extra buffer is introduced between Voronoi cells, which allows deployment of unmanned aerial swarms (D. Zhou et al., 2017).

Power diagrams allow us to obtain the robot's collision-free, convex neighborhoods in cluttered environments. Where the robot inhabits one cell and each surrounding obstacle the remaining ones. The straightforward path optimization for the agent is performed in its local, convex cell. Resulting in a continuous vector field around circular obstacles without encountering local minima (Arslan and D. E. Koditschek, 2016a). Given sufficiently separated obstacles with a strongly convex hull, Voronoi diagrams show full convergence to the goal. Moreover, the resulting vector field is continuous and piece-wise smooth (Arslan and D. E. Koditschek, 2019). An alternative strategy involves shaping Voronoi cells to tangentially border circular obstacles, thereby maximizing the agent's cell without incorporating the obstacles (Pierson and Rus, 2017). This maintains cell convexity while empowering agents to optimize over an extended distance within their larger cells while ensuring the simplicity of the local optimization.

Alternative unstructured decomposition of the free (polygonial) space into convex regions can be obtained through a randomized, incremental algorithm (Seidel, 1991). A structured graph indicates the fastest path towards the goal from each cell, which is used to create a smooth vector field (Lindemann and LaValle, 2009).

Probabilistic Roadmaps

The Probabilistic Roadmaps (PRM) algorithm divides motion generation into two phases. The learning phase involves constructing the roadmap graph. During this phase, the algorithm samples collision-free configurations stored in the graph node. The connecting edges represent feasible paths between these configurations, determined using a local planner (L. E. Kavraki et al., 1996). The subsequent query phase takes the robot's desired start and end configurations and connects them to the roadmap graph to find the shortest path (Amato and Wu, 1996; L. Kavraki and Latombe, 1994). The local roadmap creation can be accelerated using a local, rapid obstacle avoidance algorithm (J. F. Canny and M. C. Lin, 1993).

PRM encompasses multiple steps, including collision checking, graph search, configuration sampling, and local planning. These steps have enabled numerous PRM variants (Geraerts and Overmars, 2004). Creating edges between configurations involves checking all nodes within a constant radius of each other. An improved version known as PRM* adapts the connection radius as a function of the number of nodes n and dimensions N (Karaman and Frazzoli, 2011), which leads to shorter trajectories. However, PRM* focuses solely on pure path planning and does not allow for accounting for kinematic constraints. A *lazy* variant of PRM defers collision checking until it is certain that a specific edge is part of the optimal path (Bohlin and L. E. Kavraki, 2000). A lazy evaluation of PRM also allows to adapt the trajectory concerning a dynamically changing environment (Jaillet and Siméon, 2004). In the context of PRM, the single-query approach employs the start and end configurations as seed points to build the roadmap. To expedite the process, the step size is increased in definitively collision-free environments (G. Sánchez and Latombe, 2003). However, narrow passages are a common challenge for PRM, and Arslan, Pacelli, and D. E. Koditschek, 2017 introduces a sensor-based, greedy steering law that enhances the discovery of challenging regions like narrow passages. Ravankar et al., 2020 utilize a hybrid method that combines artificial potential fields and PRM to improve convergence and mitigate dynamic obstacles.

However, a study by LaValle, Branicky, and Lindemann, 2004 indicates that the randomness inherent in the configuration sampling of PRM does not necessarily explicitly have advantages over a deterministic approach. While it is evident that a deterministic grid becomes impractical in higher dimensions due to the curse of dimensionality, a fully randomized approach can lead to uneven density across regions. An improved approach could involve adding nodes based on the current distribution of configurations.

Rapidly-Exploring Random Tree

The Rapidly-Exploring Random Trees (RRT) algorithm is a dynamic approach that effectively populates the search space by branching out a space-filling tree to find feasible paths (LaValle et al., 1998; LaValle and Kuffner Jr, 2000). Unlike simple graph-based structures, the tree consumes less memory and holds the added benefit of accommodating motion constraints. One of

its distinctive traits is the adaptability to address non-holonomic constraints, dynamic scenarios, and high degrees of freedom. RRT's iterative expansion involves applying randomized control inputs, guiding the virtual system towards randomly chosen points (LaValle and Kuffner Jr, 2001). RRT* takes this concept further by considering all neighboring nodes within the connection radius when integrating a new node into the tree, a departure from the approach of the original RRT. The incorporation connects to the node with the lowest cost from the node to its potential parent and from the parent to the root. Despite these advancements, RRT* reduces to a pure path planning problem in N-dimensional space, devoid of kinematic constraints.

RRT's modular architecture has led to numerous extensions driven by specific challenges and applications (Noreen, Khan, and Habib, 2016). The primary challenge that RRT encounters is navigating through narrow passages. Rodriguez et al., 2006 proposed enhancing the exploration process by elevating obstacles and workspaces' geometric and physical properties. Vonásek et al., 2009 introduced local Voronoi diagrams to guide exploration, thereby facilitating local convergence. Shi, Denny, and Amato, 2014 introduced a technique wherein RRTs are seeded in narrow passages to map them comprehensively. These localized maps are then integrated into a PRM-graph, accelerating workspace exploration. Other extensions include incrementally building two RRTs at the start and end positions, guided by a greedy heuristic that balances movement toward each other and yields a single-query planning solution (J. J. Kuffner and LaValle, 2000). The *anytime* RRT, prioritizes finding a feasible path initially and incrementally refining the trajectory thereafter. This strategy is advantageous in time-constrained scenarios, enabling a viable solution at any time (Karaman, Walter, et al., 2011). By parallelizing collision evaluation using artificial neural networks, RRT has successfully circumvented collisions in seven-dimensional configuration spaces (Koptev, Figueroa, and Billard, 2021).

RRT and PRM share many similarities in their randomized exploration of space, followed by subsequent graph-based path planning. As a result, they share similar extensions to their base algorithms.

Space-Time Sampling

Incorporating time as an additional dimension within the state representation introduces an interesting approach to employing sampling methods in the context of dynamic obstacles. Unlike conventional sampling methods, however, the trajectory must evolve exclusively in the positive time direction while adhering to prescribed velocity constraints.

The challenge of trajectory planning can be divided into discrete path and velocity planning problems (Kant and S. W. Zucker, 1986). The *path-time* representation of the task encompasses the temporal evolution of obstacles. This is used for graph-like path planning. Subsequently, velocity profiles are evaluated based on physically feasible paths. An alternative technique involves elevating the robot's configuration space, crafting a configuration *space-time* model. This approach is instrumental in describing plausible motion trajectories in the presence of

volumetric and dynamic obstacles and interactions between multiple robots (Erdmann and Lozano-Perez, 1987). Similarly, the *state-time* model augments the task-space constraints of the robot during path planning (Fraichard, 1998).

Trajectories can be created through the integration of Probabilistic Roadmaps (PRM) with the *state-time* representation (J. P. Van Den Berg and Overmars, 2005). Employing anytime algorithms and a partially pre-planned PRM scheme, collision-free paths within the *state-time* domain can be assessed in real-time to accommodate unforeseen obstacle trajectories (J. Van Den Berg, Ferguson, and J. Kuffner, 2006). Capitalizing on the *state-time* model, a paradigm encompassing simultaneous planning, optimization, and execution has enabled adaptive motion planning for a mobile manipulator (Vannoy and Xiao, 2008).

Leveraging time as an additional dimension can come with distinct advantages, particularly in the centralized motion planning of multiple robots. Nonetheless, the efficacy of such methods is inherently reliant on accurate predictions of future positions and velocities of obstacles, even for dynamic algorithms designed to handle these scenarios.

General Remarks

The computational complexity inherent in sampling-based algorithms, such as RRT and PRM, exhibits a linear relationship to the number of nodes n within the sampled tree, which in turn is exponentially linked to the dimensionality N: $\mathcal{O}(n) = \mathcal{O}(\exp(N))$ (Noreen, Khan, and Habib, 2016). While these algorithms possess probabilistic completeness, signifying their theoretical capability to discover solutions given infinite time, their reliance on a sampling-oriented approach results in a significant computational burden. This limitation curtails their practicality for real-time recalculations, a crucial demand in dynamic environments. To tackle this challenge, specialized on-chip circuitry has been devised to facilitate swift re-planning of RRT for robotic arm movements (S. Murray et al., 2016). Nonetheless, this entails chip designs fine-tuned to the specific robot and environment configurations. Moreover, sampling-based methods (structured and unstructured) often have a hard time exploring narrow passages.

In graph-based planners, the bulk of time is often consumed during the initial space exploration. Nevertheless, the subsequent tree search can be executed relatively efficiently. However, if subject to large disturbances or the environment undergoes substantial changes, a complete re-computation of the system becomes necessary. This can result in the *stop-compute-go* motion pattern often associated with robots.

2.1.2 Optimal Control

Optimal control is a powerful mathematical minimization problem, seeking to diminish the cost function while adhering to the imposed constraints. In robotics motion control, a straightforward problem statement considers the path's length as the cost function while concurrently imposing

constraints to ensure the avoidance of obstacles. The optimization process is executed iteratively through a series of steps following the trajectory along the negative gradient of the cost function. This approach incrementally fine-tunes the trajectory, driving it closer to an optimal solution while considering the balance between the cost function's reduction and adherence to constraints.

Local Path Deformation

While global path planning techniques present robust trajectory solutions, their drawback lies in their inability to swiftly respond to dynamic obstacles due to the prolonged computation time. However, trajectory optimization solves this limitation by adapting in real-time.

Adaptation of trajectories for robotics has been inspired by the concept of *snakes* in computer vision (Kass, Witkin, and Terzopoulos, 1988), which contours visual features to track edges within images. The proposed *elastic band* method uses the discretized trajectory within the robot's configuration space obtained through global motion planning. Subsequently, each trajectory point undergoes the influence of repulsive forces that push it away from the obstacles and internal elastic forces that counteract to retain the trajectory's original shape. To ensure collision-free paths in the presence of multiple obstacles, the trajectory is enveloped in overlapping protective bubbles (Quinlan and Khatib, 1993).

While this methodology guarantees safety, the approach becomes increasingly conservative and bubble-intensive for applications in high-dimensional configuration space. In response, *elastic strips* emerged, representing a task space trajectory describing the motion of all robotic joints, including the end-effector (Brock and Khatib, 1998). The elastic strips extend the concept of external repulsive and internal elastic forces into the task space. Employing inverse kinematics, joint torques are evaluated from the cumulative forces, allowing dynamic adaptation while controlling in task space (Brock and Khatib, 2002).

The *elastic strips* concept has been effectively combined with roadmap approaches, unifying task constraints, kinematic considerations, and dynamic limitations for mobile manipulators (Y. Yang and Brock, 2010). This strategy extends further into contact interaction tasks for humanoid robots, enhancing adaptability while accounting for specific requirements (Chung and Khatib, 2015).

While the elastic deformation techniques offer real-time adaptability, their application can result in sub-optimal paths, especially when dynamic obstacles disrupt the initial trajectory. In such scenarios, global re-planning becomes necessary, potentially leading to temporary halts in the robot's motion to ensure optimal and safe pathfinding.

Global Motion Optimization

The paradigm of optimal control involves iteratively improving trajectories through optimization. However, this approach often relies on a well-performing initial trajectory, which may not always

be accessible or have a significant computational cost. This is where motion optimization (MO) approaches scenarios where the initial trajectory could be infeasible.

Inspired by elastic band methods, Warren, 1989 proposed an MO that takes an initially colliding path and globally repulses it from the colliding region through iterative steps. Through gradient descent iterations, MO refines this trajectory, ultimately finding an optimal collision-free path. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) leverages this notion, by framing the cost function in terms of distance fields of obstacles and utilizing covariant gradient descent (Ratliff, M. Zucker, et al., 2009; M. Zucker et al., 2013). Trajectory optimization (Trajopt) introduces a sequential convex optimization process and simplified collision constraints to improve trajectories (Schulman, J. Ho, et al., 2013; Schulman, Y. Duan, et al., 2014). In comparison to CHOMP, Trajopt yields a higher proportion of converging solutions. Incremental Trajectory Optimization Motion Planning (iTOMP) offers real-time replanning in dynamic surroundings through parallelized, iterative execution and planning (C. Park, Pan, and Manocha, 2012). Meanwhile, Stochastic Trajectory Optimization for Motion Planning (STOMP) generates noisy trajectories, which are combined to derive an updated path. This eliminates the need for computationally expensive or inaccessible gradients (Kalakrishnan et al., 2011). An encompassing framework for motion planning and the implementation of various algorithms has been put forth by Sucan, Moll, and L. E. Kavraki, 2012.

MO can potentially accommodate numerous constraints, making it highly adaptable to complex scenarios, including full-body collision constraints. However, the complexity of these constraints can render the MO problem non-convex, necessitating multiple seeds for the solver to converge to a suitable path. This additionally contributes to an extended computational runtime for MO.

Geometrical Methods

Traditional optimization methods often demand linearization or simplification techniques to ensure fast optimization. However, a distinct category of geometric optimization controllers enables precise control over nonlinear mechanical systems (Udwadia, 2003). The method simplifies mechanical control problems to well-known MO formulation, where an optimal solution is found. And hence, streamlining the derivation of standard control challenges in robotics (Peters et al., 2008).

Riemannian Motion Optimization uses the Jacobian as a Riemannian metric to perform a *pullback* of the task space into the configuration space. Inspired by the diffeomorphic sphere to starshape projection (Rimon and D. E. Koditschek, 1991), a metric stretches the space toward task-space obstacles before minimizing the length of discretized trajectories. Consequently, the motion to approach obstacles becomes less desirable, resulting in the generation of collision-free trajectories (Ratliff, Toussaint, and Schaal, 2015). Leveraging insights from Riemannian geometry has led to the introduction of objective constraints that enhance the success rate of interior-point motion optimization. This was further guided by a numerical harmonic function, obtained

through diffusion-like heat propagation (Mainprice et al., 2020). Strides in position-dependent Riemannian metrics have facilitated the enhancement of task design through Riemannian Policies, enabling reactive force control under constraints (Bylard, Bonalli, and Pavone, 2021). Combining multiple Riemannian Motion Policies into a graph structure with geometrically consistent transformations has enabled tackling individual tasks collectively (C.-A. Cheng et al., 2021). Integrating geometric fabric methods into classical mechanical systems has to design complex behaviors, as demonstrated in multi-obstacle avoidance for a 7DoF robot arm (Van Wyk et al., 2022).

Geometric methods are applicable across a broad range of robotic applications, leading to improved constraint fulfillment and convergence of the MO. Nonetheless, the intricacies of their formulation and subsequent adaptation to novel tasks are key limitations hindering their widespread adoption. Poor parameterization can potentially introduce unintended motion artifacts, underscoring the need for careful tuning and design.

Model Predictive Control

Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), operates by utilizing a system model to predict future behavior and compute an optimal control response (Mayne et al., 2000). The optimization problem in MPC operates over a finite time horizon, thus mitigating complexity and improving convergence time. In essence, MPC is to Motion Optimization within a limited time horizon. The improvement in computational power, coupled with refined algorithms and pertinent simplifications, has enabled real-time control of (semi-)autonomous agents such as ground vehicles (Ji et al., 2016) or quadrotors (Nan et al., 2022).

While linear MPC has many applications, the presence of obstacles often necessitates using nonlinear MPC (NMPC) due to intricate system boundaries. However, the computational demands of nonlinear optimization methods are often much larger. Hence, appropriate algorithmic strategies are required to ensure convergence in finding a solution in real-time, such as individual wheel torques (Frasch et al., 2013) or whole-body motion generation for a humanoid robot (Koenemann et al., 2015). Leaps in computational power have made real-time collision avoidance possible using MPC (Williams, Aldrich, and Theodorou, 2017; Ji et al., 2017). Nonlinear MPC allows the incorporation of non-holonomic constraints, which has enabled the control of dynamic quadrotor flight (I. Sánchez et al., 2021) Nonlinear MPC has additionally been instrumental in controlling non-holonomic robots navigating among multiple convex obstacles. However, exact collision checking is computationally expensive for volumetric agents and general environments. Hence, simplifications are applied, such as using repulsive forces inspired by artificial potential fields (Ji et al., 2016) or simplifying the obstacle and agent to convex shapes (X. Zhang, Liniger, and Borrelli, 2020). In scenarios lacking an analytical cost function, sampling-based MPC has proven effective for dynamic configuration-space collision avoidance (Bhardwaj et al., 2022). Advancements in real-time optimization algorithms have brought MPC within reach for onboard robot control, capitalizing on hardware's improved computational capacity. This has facilitated

shared control of wheelchairs using MPC (Bardaro et al., 2018).

The limited horizon of MPC decreased the computational cost. However, it also restricts solutions to local optimality, thereby hindering the guarantee of global convergence. Like MO, complex constraints can be integrated, which amplifies computational time. Consequently, the design of MPC warrants careful consideration to achieve desirable system behavior. Nevertheless, MPC-based techniques sometimes struggle to ensure feasible solutions.

Control Barrier Functions

Barrier functions partition the state space of a control system into safe and unsafe regions, facilitating trajectory feasibility assessments without necessitating explicit knowledge of reachable workspace boundaries (Prajna and Jadbabaie, 2004). While these functions can arise from analytical system constraints, they can be obtained through learning from demonstrations (Saveriano and D. Lee, 2019). Barrier functions can represent task space constraints, such as obstacles, or configuration space constraints, such as joint limits. Control Barrier Functions (CBFs) extend this concept, enabling the direct derivation of feasible feedback control from the system's state, which ensures that the system remains within the safe region (Wieland and Allgöwer, 2007). This mirrors the Control Lyapunov Functions (CLFs) approach that elevates Lyapunov functions to directly design feedback controllers to achieve global stability (Artstein, 1983).

Combining CBFs and CLFs using quadratic programming (QP) has led to collision-free path planning methodologies (Ames, Grizzle, and Tabuada, 2014). However, the CLF constraint was relaxed to guarantee a feasible control, leading to the absence of asymptotic convergence assurances. An alternative approach unifies CBFs and CLFs into a cohesive framework, termed Control Lyapunov Barrier Functions (CLBFs) (Romdlony and Jayawardhana, 2014). Although CLBFs offer direct control extraction without optimization, this convenience comes at the cost of conservative motion around obstacles (Romdlony and Jayawardhana, 2014).

Introducing a virtual orientation state steers the feedback system towards local minima while ensuring the collision constraints (Reis, Aguiar, and Tabuada, 2020). However, this strategy occasionally necessitates temporary violation of CLF-based constraints. To ensure convergence, Notomista and Saveriano, 2021 employ a diffeomorphic transformation from a spherical to a starshaped-world, inspired by (Rimon and D. E. Koditschek, 1991). This transformation simplifies the control problem but requires complete knowledge of all obstacles, rendering it challenging in dynamic environments.

While numerous state-space methods assume point-like agents limited by the CBF, Notomista and Saveriano, 2021 ensures collision avoidance for multiple rigid ellipsoidal bodies. The approach extracts the separating hyperplanes to ensure a safe distance from them.

As the deployment of autonomous systems increases, the significance of safe controller design becomes apparent, especially in unstructured environments for which deterministic control

systems are hard to design. CBFs enable the development of safety-critical systems, as discussed in Ames, Coogan, et al., 2019. However, like Lyapunov (control) functions, designing CBFs requires a substantial understanding of the underlying system. While CBFs assure safety, there is no universally applicable solution for constructing such functions for any control system. Moreover, when combined with optimization, CBFs sometimes struggle to guarantee convergence or necessitate the relaxation of constraints.

2.1.3 Determinisic Reactive Obstacle Avoidance

Regarding collision avoidance, the ability to provide performance guarantees takes precedence. A distinct advantage is the controller's capacity to discern whether a problem can be resolved in real-time to evade an obstacle. This foresight enables dependable predictions about the algorithm's behavior in specific scenarios and establishes a foundation for reliable decision-making.

Bug Algorithm

The bug algorithm (BA) is a navigation algorithm tailored for mobile robots, earning its name due to its trajectories resembling the movements of a crawling bug. In its basic form, it functions as follows. The agent progresses straightforwardly toward the target until it encounters an obstacle. It then turns, either to the left or right (with a consistent rotation direction) and follows the obstacle's surface. Finally, the algorithm switches back to moving towards the goal once it determines that the path toward the goal no longer leads to a collision (Lumelsky and Stepanov, 1986).

The simplicity of the BA results in low computational cost and convergence in numerous scenarios. However, the bug algorithm does exhibit limitations, particularly in complex environments with highly non-convex obstacles it can get trapped in infinite loops. Enhancements to convergence were proposed, including circling the obstacle before exiting at the point nearest to the goal (Sankaranarayanan and Vidyasagar, 1990), albeit this often results in longer trajectory lengths. Alternatively, global convergence can be guaranteed by sticking to a straight-line trajectory from the start to the goal position (Kamon and Rivlin, 1997). Exiting the surface, the following mode is done only when this line is reached. BA can be extended to three-dimensional space, where point clouds are used as input (Kamon, Rimon, and Rivlin, 1999).

The BA's efficacy lies in its simplicity and independence from global knowledge, making it suitable for robots with limited computational capabilities. However, the BA's convergence often depends on precise localization (McGuire, Croon, and Tuyls, 2019), a challenge in real-world scenarios where perception and navigation tasks demand significant computational resources.

The BA is similar to the wall-following algorithm, often used as a solution to maze solving. However, wall following requires a fully connected space rather than individual obstacles. It can be seen as a simplified BA, where the agent is always attached to the surface.

Artificial Potential Fields

Artificial Potential Fields (APF) represent each obstacle as a repulsive field superposed with an attractive field toward the global goal. This creates a global vector field that controls the robot (Khatib, 1986). However, APF is prone to local minima in free space, and an iterated potential value allows the construction of APF with almost global convergence (D. Koditschek, 1987). Alternatively, using superquadrics as potential functions results in enhanced convergence behavior. Superquadratic functions generate isolines that mimic the obstacle's shape on the surface while transitioning into circular contours as the distance increases (P. Khosla and Volpe, 1988).

The reactive nature of APF makes it ideal for dynamic environments, demonstrating efficacy in scenarios like tracking moving targets (L. Huang, 2009). Notably, APF has also been successfully integrated into closed-feedback loops to enable real-time collision avoidance for robotic manipulators (Tulbure and Khatib, 2020). While APF's efficiency and ease of implementation are commendable, its susceptibility to local minima in multi-obstacle environments remains a challenge (Koren and Borenstein, 1991).

Leveraging sensory input of event cameras, the lightweight and fast nature of APF can guide quadrotors to effectively evade moving objects, exemplified by successfully avoiding an incoming soccer ball (Falanga, Kleber, and Scaramuzza, 2020).

Navigation Functions

Navigation functions (NFs) were developed to address the issue of multiple local minima by constructing APFs (D. E. Koditschek and Rimon, 1990). While initially designed for sphere worlds, NFs utilize diffeomorphic mappings between *star-worlds* and *sphere-worlds* to extend their applicability to more general scenarios (Rimon and D. E. Koditschek, 1991). NF can also be employed to navigate around inverted obstacles that represent space boundaries, and their scope has been further expanded to include navigation around *trees-of-stars* (Rimon and D. Koditschek, 1992). However, constructing valid NFs requires comprehensive knowledge of the obstacle distribution, including the distances between obstacles, making tuning critical parameters challenging. Efforts have been made to simplify the creation of NFs by simplifying the tuning of critical parameters in sphere worlds (S. G. Loizou, 2011; Paternain, D. E. Koditschek, and Ribeiro, 2017), and alternative approaches have been proposed to automate the tuning process (S. G. Loizou, 2017). Full convergence around ellipse obstacles can be achieved using quadratic potential functions (Kumar, Paternain, and Ribeiro, 2019). Machine learning methods can tune the hyper-parameters of potential fields to obtain human-inspired behavior for obstacle avoidance (A. Duan et al., 2020).

NFs offer the advantage of quick re-evaluation, making them robust to disturbances. NF capabilities can be extended to include discovering obstacles' positions at runtime, but the approach relies on full knowledge of obstacles' shapes (S. Loizou and Rimon, 2022). Nonetheless, due to

the inherent difficulty in tuning NFs for general obstacle configurations, they are challenging to apply to dynamic environments.

Danger Fields

Danger-fields (Lacevic and Rocco, 2010) estimate the danger levels in the vicinity of a robot. This field is evaluated based on the robot's state and velocity. Unlike similar methods such as APF, danger fields are defined with respect to the robot (rather than the obstacle) and incorporate speed into the analysis. Finally, the danger field is evaluated at the positions of the obstacles, such as the limbs of the human operator.

Danger-fields can be used to generate collision-free controllers around the robot. At each obstacle's position, the gradient of the danger-field is evaluated to derive a repulsion force. The cumulative impact of these forces is then exerted upon the robot arm (Lacevic, Rocco, and Zanchettin, 2013). This control scheme can be combined with APF (Zanchettin, Lacevic, and Rocco, 2015). However, as previously discussed, designing such APF remains challenging, especially within higher-dimensional configuration spaces.

Safety-fields introduce a more comprehensive perspective by incorporating not only the danger source's relative position and velocity vector but also its shape and size (Polverini, Zanchettin, and Rocco, 2014). Subsequently, the concept of danger zones emerges, describing regions within space where safety constraints are violated. This concept can optimize collaborative robots' speeds while meticulously preventing overlap between danger zones and human operators, leading to a collision-free, reactive motion (Lacevic, Zanchettin, and Rocco, 2022).

While danger field has demonstrated their effectiveness in collision avoidance, they cannot inherently guarantee goal convergence without supplementary trajectory planning. Nevertheless, collision avoidance is the priority in scenarios involving human-robot collaboration, and the absence of convergence is accepted since the operator's presence allows for manual intervention to navigate the robot out of a local minimum.

Harmonic Potential Functions

Harmonic Potential Functions (HPF) are widely used in fluid dynamics to describe the potential laminar flow of incompressible fluids (Milne-Thomson, 1996). HFPs present closed-form solutions for generating smooth two-dimensional flow patterns around circles and ellipses. HPF ensures the absence of topologically critical points in free space. This means that the desired velocity field obtained from HPFs never vanishes in free space, conversely to methods such as APF.

As analytical HPFs are difficult to define for a general environment, numerical approximations have proven helpful in practice (Connolly, Burns, and Weiss, 1990; Connolly, 1997). Linear panel

representations for obstacles represent closed-form HPFs (J.-O. Kim and P. K. Khosla, 1992). While this method is limited to a two-dimensional workspace, it can deal with concave obstacles (Feder and J.-J. Slotine, 1997). In more complex scenarios, interpolation techniques have been employed, where known local HPFs were interpolated to navigate complex environments (Guldner and Utkin, 1993). Numerical HPFS, inspired by Poisson's equation, have additionally enabled control of volumetric agents within two-dimensional spaces (Z. Li and Bui, 1998).

Dynamic obstacles can be considered by evaluating the HFP in the local frame of obstacles (Waydo and R. M. Murray, 2003; "Real-time obstacle avoidance using harmonic potential functions" 2011). Furthermore, complex obstacle shapes can be described by dividing them into multiple circular obstacles. The weighted sum of the corresponding HPF is combined with a stream function to create a smooth path (Daily and Bevly, 2008). HPF motion can be used for motion planning in three-dimensional, although limitations persist, primarily in dealing with elliptical and concave obstacles Lau, Eden, and Oetomo, 2015.

While HPFs offer considerable advantages, their application does not inherently guarantee asymptotic convergence toward the goal. Moreover, analytic HPFs are hard to find, and numerical approximations or interpolations are not guaranteed to maintain the topological properties of the original HPFs.

Dynamical System

A Dynamical System (DS) is a mathematical framework that characterizes a system's evolution. It comprises a state vector describing the system's state and a function that dictates how this state evolves through time (Birkhoff, 1927). In robotics, a *desired* DS can represent the intended motion, which is then approximated by a suitable controller (Billard, Mirrazavi, and Figueroa, 2022). DS-based control offers notable advantages, primarily the ability to evaluate control based solely on the robot's current state, independent of time. Moreover, DS controllers often employ computationally efficient functions to describe the desired motion, rendering them robust against disturbances as the controller does not require extensive pre-planning. Many motions are approximated by first-order DS, outputting the desired velocity based on the current position. When used for force control on collaborative robots, they are often coupled with passive force controllers to approximate the DS (Kronander and Billard, 2015). However, for complex force patterns, such as contact tasks, designing second-order DS is more appropriate (Salehian and Billard, 2018). DS can be manually designed for specific tasks (Khoramshahi and Billard, 2019) or generated through machine learning algorithms, like learning from demonstrations (S. M. Khansari-Zadeh and Billard, 2011). DS should exhibit stability and possess low computational costs for successful deployment regardless of their origin.

Modulation of Dynamical Systems (DSM) is an effective strategy to obtain advanced behavior using DS, such as collision avoidance in reactive environments. DSM involves multiplying the initial DS with a modulation matrix, improving the initial motion. By maintaining the

modulation matrix at full rank, DSM ensures the preservation of stable points (Kronander, Khansari, and Billard, 2015). By creating DSM inspired by HPF, reactive collision avoidance in the presence of multiple obstacles is achieved (S. M. Khansari-Zadeh and Billard, 2012). DSM can address dynamic obstacles by evaluating these obstacles within an averaged-moving frame (S. M. Khansari-Zadeh, 2012). DSM can be directly applied to the point cloud data from sensors. The algorithm extracts the closest distance to the surface and approximates the surface normal (Saveriano and D. Lee, 2013). Obstacles with small concavities can be avoided, too, as the DSM uses the approximation of the surface being locally flat (Saveriano and D. Lee, 2014). DSM has successfully been integrated into a unifying framework, which includes human action monitoring and action reshaping, enabling successful human-robot interaction (Saveriano, Hirt, and D. Lee, 2017). DSM can overcome local minima on the border of intersecting convex obstacles by applying a switching to surface following as soon as such a zone is reached (Zheng et al., 2020). Using a reference point for each obstacle, which gives DSM additional information about the center of the obstacle, and increases the convergence around convex and concave (star-shaped) obstacles (L. Huber, Billard, and J.-J. Slotine, 2019).

A smoother motion can be achieved by restricting the normal component of the MDS with respect to the obstacle while freely adapting the tangent part. Hence allowing the control of curvature while ensuring collision avoidance, as, for example, needed aerial vehicles (Marchidan and Bakolas, 2022). MDS relies on star-shaped obstacles to ensure convergence. To achieve this, the workspace can be segmented into disjoint star-shaped obstacles, excluding both the current robot position and the attractor position to path feasibility (Dahlin and Karayiannidis, 2023a). Alternatively, MDS paths can be used as reference trajectories to define constraining tunnels for an MPC algorithm that outputs the robot control (Dahlin and Karayiannidis, 2023b). Combining MDS and MPC, Dahlin and Karayiannidis, 2023c further demonstrated the ability to achieve setpoint stabilization and more complex path-following tasks, showcasing the potential of this integrated approach in enhancing two-dimensional motion planning.

MDS shows excellent potential in dynamic motion planning. However, current approaches come with limitations to the environment configuration, such as the requirement of distinct star-shaped obstacles. Furthermore, most algorithms require an analytical obstacle description, often requiring computationally expensive image recognition.

Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) employ a (virtual) canonical state that evolves by a dynamical system (Ijspeert, Nakanishi, and Schaal, 2002; Saveriano, Abu-Dakka, et al., 2021). The virtual system governs the activation of the attractor of the real system. This ensures the global convergence of the actual system. Even though the DMP is not explicitly dependent on time, the evolution of the canonical system makes it implicitly time-dependent. The system is reactive and stable against external disturbances, while the virtual system remains deterministic. As the virtual system activates consecutive local attractors along a path, this approach can be

seen as a global path planning method (Ijspeert, Nakanishi, Hoffmann, et al., 2013). While often used in learning from demonstration, the DMP can also be generalized and extended for new tasks using the sensorimotor knowledge of the system (Ude et al., 2010).

DMPs can be applied to learning from demonstrations, where they can be taught to replicate collision-free motion in static environments. However, DMP lacks awareness of obstacles at any given moment; instead, it merely follows the avoidance path (Matsubara, Hyon, and Morimoto, 2010). Obstacle awareness can be introduced to DMPs by incorporating nonlinear *coupling* terms. These coupling terms can be learned from demonstrations with respect to the relative position of obstacles and have been successfully employed to dynamically modify trajectories (Rai et al., 2014). By using a coupling term APF to the differential equation describing the dynamical system enables avoidance of point-like obstacles. The coupling term adds repulsive force based on the relative position and velocity of the obstacle, allowing for dynamic environments (D.-H. Park et al., 2008). An approach that can be extended to three-dimensional space (Hoffmann et al., 2009). In more complex scenarios, a super-quadratic potential function can be employed to model obstacles in two and three dimensions. This allows for navigation around convex, volumetric obstacles using the DMP framework (Ginesi et al., 2019).

DMPs have the advantage of being able to represent complex motion while ensuring global convergence. However, DMPs come with implicit time dependency, which might be undesirable in human-robot collaboration scenarios. Moreover, introducing active obstacle avoidance in the DMP framework results in an absence of global asymptotic stability.

Vector Fields

Vector fields (VFs) describe state-dependent motion, applicable when dynamics can be modeled as a first-order system of differential equations. VFs can be used to describe various time-invariant motions and have shown beneficial to design path-following dynamics in N-dimensions (Goncalves et al., 2010). In this approach, a (non-intersecting) path is embedded in the VF, enabling the system to autonomously re-approach the path whenever it deviates. This mechanism ensures stable dynamics when controlling volumetric, rigid bodies (Kapitanyuk, Chepinskiy, and Kapitonov, 2014). By transforming self-intersecting trajectories into non-intersecting ones, VFs permit tracking more general paths while guaranteeing the absence of singularities (Yao, Marina, et al., 2021).

Furthermore, VFs can decode behaviors that ensure collision avoidance. For instance, a locally rotating VF can be combined with a guiding VF toward a goal position, enabling goal approaching while also ensuring obstacle avoidance. However, although effective at avoiding highly concave shapes, this method may introduce undesired motion artifacts (Liddy et al., 2008). VF has been employed to evade multiple circular obstacles in two-dimensional space using local repulsive fields (Panagou, 2014). The smoothness of VF techniques has also found application in controlling fixed-wing aircraft (Wilhelm and Clem, 2019). Combining path-following with reactive obstacle

avoidance has allowed the execution of initially nonlinear dynamics, such as following a limit cycle, in the presence of obstacles. However, this method requires discontinuous switching when the initial controller reaches a local minima (Yao, B. Lin, et al., 2022).

In general, VFs are designed to exhibit Lipschitz smoothness across space and bounded in magnitude. Using VF to model the desired dynamics of a system is equivalent to using a first-order DS. However, it is to be noted that many proposed obstacle avoidance methods using VF are limited to considering one obstacle at a time.

Aguiar, Hespanha, and Kokotović, 2008 shows that global path-following dynamics, as defined by DS or VF, are less restrictive than tracking a reference signal, as is used in the DMP-framework. This is a result, of the path-following having an additional state related to the time, $\Theta(t)$, which constrains the evolution of the reference signal along the desired motion.

2.1.4 Data Driven Navigation

The availability of digital data coupled, and cost-effective computational power has enabled the use of data-driven navigation methods. While Machine Learning techniques are among the most prominent, data can also be sourced from a few expert-derived rules or discovered by imitating natural selection processes.

Fuzzy Logic

Fuzzy Logic (FL) effectively maps a continuous spectrum of environments and states to logical control using a set of IF-THEN rules (Zadeh, 1965; Zadeh, 1996). Inspired by human decision-making processes, FL excels at seamlessly integrating both numeric (*fuzzy*) and symbolic (*logic*) elements of reasoning.

FL has garnered recognition for its robustness in addressing real-time decision-making and automatic control problems in uncertain environments (Saffiotti, 1997). The underlying logic allows us to combine different robotics behaviors, such as speed control, goal searching, and obstacle avoidance (Nasrinahar and Chuah, 2018). Combined with other methods, fuzzy logic can take over the higher level control and has been used to steer robots out of local minima generated by APF (Tsourveloudis, Valavanis, and Hebert, 2001; Teli and Wani, 2021). The intuitive nature of FL allows for the design of robust controllers based on qualitative models, leading to a wide array of applications and algorithmic developments, as reviewed by Hentout, Maoudj, and Aouache, 2023.

However, the logical nature of FL can limit the applicability of conventional control analysis tools. Often, only experimental assessment of the algorithm is feasible. Consequently, in cases where classical controllers are available, they are often preferred due to their well-established analytical frameworks. Moreover, when FL incorporates various control behaviors, it can give

rise to conflicting commands, potentially leading to indecisive or jerky robot motion. Careful consideration of such challenges is essential in the design and implementation of fuzzy logic-based control systems.

Learning the Desired Motion Behavior

Machine Learning (ML) enables approximating complex concepts without the need for explicit definition or analytical system knowledge, making it interesting for various aspects of robotic control and navigation.

The most holistic approach in ML-based robotics is end-to-end learning, where a robot's behavior is learned directly from control inputs. This method enables the acquisition of complex robot behaviors with minimal human intervention. It finds applications in diverse areas, including collision avoidance for off-road vehicles based on camera input (Muller et al., 2005), collision-free multi-robot navigation (Riviere et al., 2020), and high-speed control using camera feedback (Michels, Saxena, and Ng, 2005). However, while end-to-end learning can produce impressive results in situations similar to the training data, it often struggles to generalize to more diverse or unfamiliar settings due to the multitude of potential control behaviors for the same sensory input.

Computer vision, one of the most impactful fields of ML in robotics, has greatly enhanced obstacle detection capabilities (Voulodimos et al., 2018). Large datasets and standardized visual decoding techniques have allowed the creation of large and powerful models (Redmon et al., 2016). However, onboard inference times for real-time obstacle detection can be a limitation.

Learning specific (collision-free) motion can result in more versatile dynamics following a globally sampled path. Conversely, algorithms like Rapidly-exploring Random Trees (RRT) have been the foundation for training reinforcement learning models around obstacles (Cai et al., 2023). When expert data is available, learning from demonstration can reproduce complex motions (Ravichandar et al., 2020). For learning a motion controller, stability is ensured at the design stage to ensure safe execution (Hersch et al., 2008; S. M. Khansari-Zadeh and Billard, 2011). These approaches create general dynamics but lack obstacle awareness, limiting collision avoidance capabilities and adaptability to changing environments.

Reinforcement learning allows robots to explore diverse scenarios, topologies, and control schemes to improve their models (Cai et al., 2023). However, exploration can be time-consuming due to the vast exploration space. Heuristics, such as leveraging previously explored Probabilistic Roadmap (PRM) paths, can expedite creating paths in cluttered obstacle environments (Y. Zhang, Fattahi, and W. Li, 2013).

ML can approximate specific control scheme components, particularly those challenging to model analytically or computationally expensive to evaluate. This includes tasks like learning robot gaits to enhance limb control (Tsounis et al., 2020), estimating the rigid-body motion of incoming objects (S. Kim, Shukla, and Billard, 2014), or accelerating collision checks through

neural network parallelization (Koptev, Figueroa, and Billard, 2022).

The broad applicability of ML methods has enabled many offline and online robotics applications, as reviewed in (Kroemer, Niekum, and Konidaris, 2021). However, applying ML in robotics comes with many challenges. Deep learning methods, which come with a large number of parameters, entail substantial computational costs that hinder real-time applicability (LeCun, Bengio, and Hinton, 2015). Additionally, ML models do not have a straightforward method for improving the model. Adding data for certain situations might decrease the performance in others (Brunke et al., 2022). Furthermore, it is challenging to predict where and under what conditions ML algorithms may fail, as seemingly similar scenarios can yield vastly different outcomes and collision avoidance can mostly not be guaranteed (Tsymbal, 2004). Further research is needed to address these limitations and ensure the safe and effective integration of ML in robotics applications.

Genetic Algorithms

Genetic Algorithms (GAs) explore the search space to discover optimal solutions for controller design (Box, 1957; Bremermann et al., 1962). This iterative process is inspired by genetics and can be divided into two key steps (Holland, 1992). (1) Selection identifies the best-performing individuals for advancement to the next iteration. (2) Reproduction and mutation combine two parameter sets, and random changes are introduced (mutation) in specific values.

GAs can be applied to robotics in various scenarios, including global path planning. For instance, GA can create global paths by comparing different paths to the goal in terms of length, which are recombined and obtained by combining segments of the individual paths. The path segments are obtained by dividing the two paths with respect to the intersection point. Hence, this does not easily translate to higher dimensions (Xiao et al., 1997). Efficient collision avoidance checks, coupled with crossover, have extended GA-based path planning to swarms operating in dynamic environments (S. X. Yang, Hu, and Meng, 2006). Improved mutation methods, such as seeking locally collision-free nodes before random selection, have been proposed, allowing application in simple yet dynamic environments (Tuncer and Yildirim, 2012). In multi-robot systems, GAs have facilitated collision avoidance during the repositioning of the agents by comparing sequences of discrete control sequences that undergo mutation and recombination (López-González et al., 2020).

The simple algorithm of GA can be applied to a wide range of complex environments, with little domain knowledge. However, like learning algorithms, GAs are susceptible to getting trapped in local optimal solutions, and an ideal path might never be found due to inadequate initial trajectories. Similar to graph-based methods, the search space expands significantly with the number of dimensions. Hence GAs have been mainly applied to mobile navigation applications.

2.1.5 Collision Control

Among collision control methods, the least intrusive approach involves real-time collision checking, where the system continuously monitors for potential collisions during operation. While these methods offer advantages in terms of computational efficiency and minimal disruption to the desired path, they typically lack guarantees of convergence and are, in general, complemented with global planning methods for more comprehensive collision avoidance.

Dynamic Window Approach

The Dynamic Window Approach (DWA) (Fox, Burgard, and Thrun, 1997) is a dynamic motion planning method that operates in velocity space, the *dynamic window*. It is constructed to consider both the robot's dynamic constraints, surrounding obstacles and the velocities attainable within a short time interval. The DWA algorithm determines the robot's velocity by considering local attractors derived from a global path planner Fox, Burgard, Thrun, and Cremers, 1998.

Integrating DWA with NFs allows the robot to make informed decisions while navigating around obstacles. The NFs are tailored to the local environment and give a global avoidance velocity, while the DWA does the feasibility evaluation (Brock and Khatib, 1999). Interpreting the DWA as a constraint MPC and integrating it with Control Lyapunov Functions (CLF) can ensure global convergence toward the goal position. This approach provides a robust framework for guiding the robot toward its destination with stability guarantees (Ogren and Leonard, 2005). By considering the anticipated motion of surrounding obstacles, the DWA algorithm can further improve its behavior. Accounting for predicted obstacle movements enables the robot to proactively avoid potential collisions and navigate more efficiently in dynamic environments. (Missura and Bennewitz, 2019).

The Dynamic Window Approach is a versatile and effective tool for robot motion planning, enabling real-time adjustments of velocity commands within a dynamically defined window. The DWA's simple feasibility check of potential velocity has allowed a combination with various global avoidance algorithms.

Vector Field Histograms

The Vector Force Fields (VFF) (Borenstein and Koren, 1989), is a reactive motion planning method that combines two concepts to enable mobile robots to navigate their environments. Firstly, VFF generates a certainty grid based on sensor readings, creating a representation of the surrounding terrain. Secondly, VFF computes repulsive forces, similar to the principles of Artificial Potential Fields (APF), by considering the certainty of each grid field and its distance from the robot. These repulsive forces help the robot avoid obstacles, while an attractive force is applied to guide it toward its goal. However, like APF-based methods, VFF is prone to end up in local minima. Hence, switching to a wall-following strategy similar to the bug algorithm enables

the robot to exit (Koren and Borenstein, 1991).

To improve navigation through narrow passages, Vector Field Histogram (VFH) introduces a histogram that characterizes the forces exerted by the grid cells at various discretized angles (Borenstein, Koren, et al., 1991). The robot selects its desired motion direction based on where the histogram forces fall below a predefined threshold. This approach facilitates the successful traversal of constrained spaces. An enhanced variant of VFH, known as VFH+, introduces streamlined parameter tuning, reduces complexity, and enhances trajectory approximation, ultimately improving the algorithm's reliability (Ulrich and Borenstein, 1998). VFH's short evaluation time makes it suitable for applications such as shared control in wheelchair navigation (Levine et al., 1999; Q. Li, W. Chen, and J. Wang, 2011).

VFH often struggles to navigate through narrow passages, when the reduction of the sensor reading to the directional histogram fails to detect narrow passages. This happens when a passage is approached from unfavorable angles or if the passage's width is narrow relative to the sensor's sampling angle. Consequently, VFH can encounter difficulties when navigating through (narrow) doorways.

Velocity Obstacles

Velocity Obstacles (VO) represent obstacle avoidance in the velocity space, by considering potential future collisions based on potential velocities of the robot (Fiorini and Shiller, 1998; Kufoalor, Brekke, and Johansen, 2018). VO assumes that the robot and the surrounding obstacles maintain their velocity within a specified time horizon.

VO has demonstrated its effectiveness in navigating in multi-agent scenarios, ensuring safe and reciprocal collision avoidance among multiple agents (Van den Berg, M. Lin, and Manocha, 2008). It has also been extended to encompass acceleration dynamics and account for non-holonomic constraints of robotic agents (Wilkie, J. Van Den Berg, and Manocha, 2009). The fast computation time enables the application of VO to shared control systems for wheelchairs, where VO enhances safety (Prassler, Scholz, and Fiorini, 2001). To reduce the appearance of local minima, VO can be extended to non-holonomic robots and consider the robot's geometry in the algorithm. Combining VO with a simple optimization problem allows finding a suitable control to navigate in dynamic crowds (Gonon, Paez-Granados, and Billard, 2021). Moreover, incorporating acceleration dynamics into VO enables robots to better anticipate the potential behavior of other agents by assuming decaying acceleration (J. Van Den Berg, Snape, et al., 2011) or constant acceleration (Gonon, Paez-Granados, and Billard, 2022).

One of the key advantages of VO is its non-invasive nature as it performs collision checking with the environment, minimizing its impact on the initial velocity. Consequently, it is often integrated with a global planner or used to adapt control commands in shared-control scenarios, ensuring both safety and efficiency in robot navigation. However, such combinations can be prone to local minima, particularly in static and cluttered environments.

2.1.6 Contribution

This thesis leverages the DS framework (Section 2.1.3) to steer robotic motion to a collision-free motion. In Chapter 5, a control scheme inspired by HPF (Section 2.1.3) is developed. This control scheme is adept at navigating within human-inhabited environments, making the DS framework for obstacle avoidance, originally developed by (S. M. Khansari-Zadeh and Billard, 2012; L. Huber, Billard, and J.-J. Slotine, 2019), applicable to a broader range of environments.

Similarly, the fast-MDS approach introduced in Chapter 6 is designed to handle unstructured sensor data. As a result, the fast-MDS efficiently computes feasible paths, but this comes at the cost of losing some convergence properties. The final algorithm shares similarities with Collision Control methods (Section 2.1.5), in particular, the reactive VFH (Sec. 2.1.5). Nevertheless, fast-MDS still shows improved behavior in a cluttered environment, where it avoids local minima, a property inherited from the HPF-inspired algorithm design.

In Chapter 7, the ROAM approach is introduced as a reactive method capable of handling complex spaces. Although it is based on the DS framework, the intricate nature of motion in complex spaces requires the use of graph-like descriptions for general obstacles. Additionally, the algorithm employs graphs of rotation to guide motion around obstacles, similar to graph-based avoidance methods (Sec. 2.1.1). However, the graph structures in ROAM are relatively small and do not require pre-computation of paths. This enables rapid replanning in dynamic environments and around complex obstacles.

2.2 Compliant Robot Controller

Impedance control, a powerful feedback algorithm, effectively applies Cartesian impedance to nonlinear manipulators' end-points (Takegaki and Arimoto, 1981; Hogan, 1985). The controller replaces the computationally intensive *inverse kinematics* with the more straightforward *forward kinematics*. Impedance control establishes a dynamic relationship between desired position, velocity, and force, offering a holistic control framework. Initially, impedance controllers employed constant stiffness, but researchers have explored various dynamic control parameter approaches to enhance adaptability in complex environments (Vanderborght et al., 2013; Abu-Dakka and Saveriano, 2020). However, introducing dynamic parameters into the control framework requires taking special care of the system's stability. Furthermore, admittance control is designed to adapt to external forces, while remaining stable contact (Glosser and Newman, 1994). Admittance control can be interpreted as a special type of impedance control.

2.2.1 Passive Velocity Controller

Passive velocity controllers use a state-dependent velocity, which is converted into a control force through a damping control law. Since the controllers have variable damping parameters, stability can be guaranteed using storage tanks inspired by a virtual flywheel (P. Y. Li and Horowitz,

1999). Furthermore, complex control parameters can be learned from human demonstrations. By continuously adapting the parameters, the controller can be observed to improve its tracking performance in direct human-robot collaboration tasks (Gribovskaya, Kheddar, and Billard, 2011). However, high compliance often results in decreased motion following. Yet, carefully designing the damping matrix, which increases stiffness in the direction of motion, but remains compliant otherwise, results in improved convergence (Kronander and Billard, 2015). Combining impedance controllers with admittance controllers can be used to increase accuracy in cooperative control (Fujiki and Tahara, 2022). The damping controller can be combined with multiple sampled via points to construct a dynamical system controller with variable stiffness profile (X. Chen, Michel, and D. Lee, 2021). Adding a feedback term based on an energy tank or quadratic programming ensures stability and convergence to the attractor, respectively (Michel, Saveriano, and D. Lee, 2023). However, these controllers' adaptations focus on improving movement accuracy rather than actively rejecting disturbances.

2.2.2 Stable Interaction Controller

Impedance controllers play an important role in interaction tasks, as in these situations, the robot is subjected to forces that are hard to predict precisely. However, they must be actively managed to ensure stable contact without damage occurring. In these situations, passive controllers using storage tanks based on the system's energy allow to limit contact force (Kishi et al., 2003). A general framework can be used to ensure that position-, torque-, and impedance controllers exhibit passivity during interaction tasks. This is achieved by interpreting the torque feedback as the shaping of the motor inertia. It allows the use of flexible robot arms for complex interaction tasks, such as insertion or wiping (Albu-Schäffer, Ott, and Hirzinger, 2007). By sensing the interaction force and actively adapting the trajectory based on the physical interaction force and the virtual repulsive force from the obstacle. This can be used for obstacle-aware motion generation (Haddadin et al., 2010).

2.2.3 Compliance in Telemanipulation

Teleoperated systems of robot arms come with control delays and require a closed-loop controller of the robot that can adapt autonomously, ensuring stable and reliable performance. Passive controllers present themselves perfectly for such a task (Stramigioli et al., 2005). This method addresses the intricate dynamics of interactive systems, ensuring stable and reliable performance. A similar approach involves slowly updating the desired position while incorporating a spring-damper model through an impedance controller, enabling seamless interaction with the teleoperated robotic system (D. Lee and K. Huang, 2010). However, these models require human input for active collision avoidance.

2.2.4 Energy Tanks

Many impedance controllers with time-varying control rely on energy tanks to ensure stability. This is a virtual state of the system, which is filled or emptied depending on the controller command. Limiting the energy tank to a maximum value ensures the system's stability (Ferraguti, Secchi, and Fantuzzi, 2013). However, when this limit is reached, the controller is constrained and interferes with the controller's optimal functioning. This can result in the controller not achieving a main functionality, such as collision avoidance. Alternatively, the impedance controller variables can be constrained by adapting the damping and stiffness, as well as their rate of change based on a Lyapunov function (Kronander and Billard, 2016). Energy tank further used that general tasks represented through Dynamic Motion Primitives remain stable, such as adaption on contact tasks (Shahriari et al., 2017).

2.2.5 Contribution

In the context of implementing obstacle avoidance on a torque-controlled robot arm, previous research by S. M. Khansari-Zadeh and Billard, 2012; L. Huber, Billard, and J.-J. Slotine, 2019 made use of a passive controller (Kronander and Billard, 2015) closely following the desired velocity. While DS passive-controlled robots have demonstrated effectiveness in interactive scenarios, their inherent drawback is the inability to guarantee collision-free navigation around obstacles. As such, the controller does not reject external forces, and the robot might be pushed into an obstacle. This thesis addresses this limitation in Chapter 8 by proposing a novel controller design. The method involves modifying the passive control law design by introducing environmental awareness into the controller's capabilities, thus enhancing its obstacle avoidance capabilities.

2.3 Multi-Agent Navigation

Multi-agent navigation is a fundamental aspect of robotics, especially in scenarios involving multiple robots or agents that require coordination to achieve a common goal. The field of Swarm Intelligence has found diverse applications in robotics (Chakraborty and Kar, 2017). However, many algorithms focus on optimizing task execution, and collision avoidance often emerges due to probabilistic swarm aggregation (Bayındır, 2016).

Many of the above-described reactive collision avoidance can inherently handle moving adversary agents and can be used for decentralized multi-agent navigation. One approach to multi-agent collision avoidance draws inspiration from Artificial Potential Fields (APF) and employs principles described as *artificial physics* (Mogilner and Edelstein-Keshet, 1999; Fetecau, 2011). This method combines a repulsive field for collision avoidance with attractive dynamics for fostering flocking behavior. While this approach effectively prevents collisions in obstacle-free environments with numerous active agents, it may not guarantee local convergence. Nonetheless,

the advantage of having all agents react to each other is that it can enhance convergence. APF-like algorithms have safely guided swarms of drones through environments like forests (X. Zhou et al., 2022). However, these methods are typically designed for circular collision robot shapes, limiting their versatility. These localized algorithms offer ease of implementation and onboard evaluation. However, they often require agents to share perception information about the environment to facilitate coordination.

Optimal Reciprocal Collision Avoidance (ORCA) is a notable approach inspired by Velocity Obstacles (VO) to enable collision-free coordination among multiple, circular agents. Where each agent takes care of half the avoidance responsibility, all agents adapt the speed with respect to the others (J. Van Den Berg, Guy, et al., 2011). Efforts have been made to extend ORCA to accommodate elliptic obstacles (Best, Narang, and Manocha, 2016) and general shapes constructed from multiple circles (Ma, Manocha, and W. Wang, 2018). However, these extensions often decouple velocity and rotation calculations, restricting agents from moving with a desired orientation.

2.3.1 Contribution

Chapter A uses the MDS (Sec 2.1.3) for the control of volumetric swarms. MDS has been developed for dynamic environments, which guarantee convergences, conversely to approaches such as Artificial Potential Fields (AFP) or Velocity Obstacles (VO), often used for decentralized swarm navigation. The chapter exploits the low computational cost of the MDS, which allows real-time evaluation of obstacle avoidance algorithm at multiple control points sequentially for each agent.

3 Preliminaries

In this chapter, we introduce fundamental concepts from mathematics, control theory, and robotics that are essential for a comprehensive understanding of the work presented in subsequent chapters.

3.1 Notation

Before diving into the core content, let us establish a consistent notation framework for our discussions.

Throughout this thesis, vectors will be denoted using bold symbols. The state variable is represented as $\boldsymbol{\xi} \in \mathbb{R}^N$, where N is the dimensional of the space. To describe the evolution of this state, we use the time derivative notation, where $\dot{\boldsymbol{\xi}} = \frac{d}{dt}\boldsymbol{\xi}$ represents the velocity of the system, and $\ddot{\boldsymbol{\xi}} = \frac{d^2}{dt^2}\boldsymbol{\xi}$ denotes the acceleration. Generally, superscripts denote variable names, while subscripts are used for enumeration. Square brackets indicate specific elements of a vector or a matrix. For example, $\boldsymbol{\xi}[1]$ refers to the first element of vector $\boldsymbol{\xi}$, and $\boldsymbol{\xi}[1:2]$ represents a sub-vector comprising the first two elements of $\boldsymbol{\xi}$.

Unless explicitly stated otherwise, the norm used is the Euclidean norm, denoted as $|\cdot| = |\cdot|_2$. The circular constant is represented by π. The identity matrix of appropriate dimensions is denoted as $\boldsymbol{I} \in \mathbb{R}^{N\times N}$. The symbol $\circ$ signifies the iterative evaluation of a function. For instance, when considering two functions, $a(\cdot)$ and $b(\cdot)$, we get that $a \circ b(\boldsymbol{\xi}) = a(b(\boldsymbol{\xi}))$. Angle-brackets denote the dot product of two vectors, i.e., $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b}$. The $\times$-operator indicates the cross product.

3.2 Dynamical Systems

Dynamical systems (DS) are useful for describing the behavior and motion of continuous-time systems, from electric circuits to mechanical systems. During this thesis, the desired dynamics are described by a closed-form dynamical system, of which an appropriate controller then approximates the motion. We focus on two possible DS: first-order dynamics, which describes

the desired velocity at the current position, and second-order dynamics, which incorporate the inertia effects on the control system.

3.2.1 Velocity Vector Field

Consider a first-order dynamical system $\boldsymbol{f} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ that describes the motion at position $\boldsymbol{\xi} \in \mathbb{R}^N$ of a continuous-time system, indicated by the state derivative $\dot{\boldsymbol{\xi}} \in \mathbb{R}^N$:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{3.1}$$

A (smooth) velocity vector field (VF) can be used to describe the evolution of this dynamics, where $\boldsymbol{\xi}$ is the state of the system and $\dot{\boldsymbol{\xi}}$ are the vector. VF and DS are used interchangeably to describe the desired motion of such first-order systems.

3.2.2 Rigid Body Dynamical Systems

Beyond describing ideal motion, DS can characterize the behavior of torque-controlled robotic systems. In this context, the DS defines the second-order dynamics of the robotic system. The following equation encapsulates the relationship between state, velocity, and acceleration:

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \tag{3.2}$$

Note that any second-order system can be transferred into a first-order system, by introducing a new variable, $\boldsymbol{\gamma} = \dot{\boldsymbol{\xi}}$. However, this system has a state that combines position $\boldsymbol{\xi}$ and velocity $\boldsymbol{\gamma}$; hence, it is of dimensionality $2N$. Furthermore, the state space has constraint velocities that can be achieved.

3.2.3 Straight Motion

In many scenarios, the goal of a robotic system is to reach a specified goal state $\boldsymbol{\xi}^a$. Following a straight motion is the shortest path to reach such a state. For this, let us define *straight dynamics*, that do maintain the direction along a trajectory:

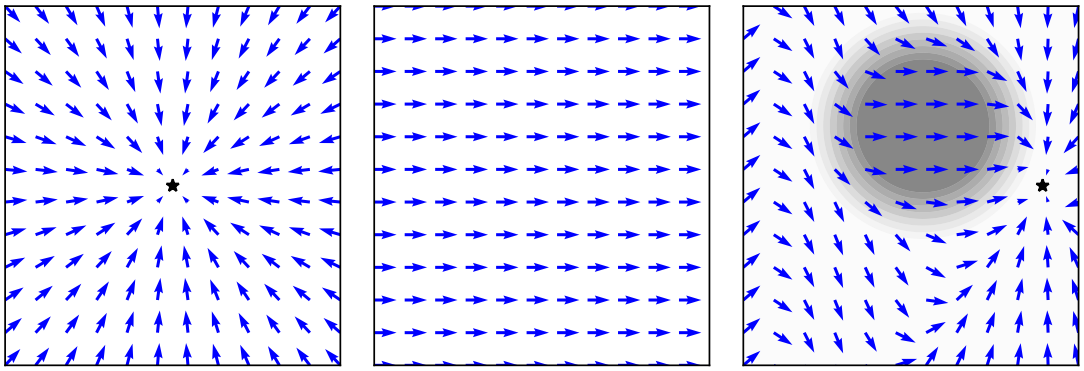
Definition 3.2.1 (Straight Dynamics)**.** A dynamical system of the form (3.1) is referred to as *straight dynamics* if, for all initial conditions $\boldsymbol{\xi}_0$ and $\dot{\boldsymbol{\xi}}_0$, the flow remains collinear with the initial velocity, satisfying $\dot{\boldsymbol{\xi}}_0^T \dot{\boldsymbol{\xi}}_t = \|\dot{\boldsymbol{\xi}}_0\| \|\dot{\boldsymbol{\xi}}_t\|$, for all $t \geq 0$. The flow is said to be *locally straight* if it is straight within a subdomain $\mathcal{X}^s$.

Multiple straight dynamics are visualized in Figure 3.1. In the subsequent sections, we adopt the following parameterization for globally straight dynamics:

$$\boldsymbol{f}^s(\boldsymbol{\xi}) = q(\boldsymbol{\xi}) \left(\boldsymbol{\xi}^a - \boldsymbol{\xi} \right) \tag{3.3}$$

where $q : \mathbb{R}^N \to \mathbb{R} \setminus 0$ is a continuous state-dependent scaling that modulates the speed of the linear system $(\boldsymbol{\xi}^a - \boldsymbol{\xi})$ towards a fixed point $\boldsymbol{\xi}^a$. This scaling allows for the adjustment of the speed of the linear dynamics as the system approaches or moves away from the attractor $\boldsymbol{\xi}^a \in \mathbb{R}^N$, which serves as the unique stable fixed point or *attractor* of the system (Fig. 3.1a). The condition $q(\boldsymbol{\xi}) \neq 0$, combined with the continuity of q, preserves the directionality of the flow. Consequently, the vector field consistently points towards or away from the attractor across the state space.

Definition 3.2.2 (Collinear Dynamics)**.** Straight dynamics which have the attractor positioned infinitely far away, i.e., $\lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^a\|\to\infty} \boldsymbol{f}^s(\boldsymbol{\xi}) = q(\boldsymbol{\xi})\boldsymbol{v}^a$, is called *(locally) collinear*.



(a) Globally straight (b) Globally collinear (c) Locally straight

Figure 3.1: Dynamical systems can exhibit different straightness characteristics. Such as globally straight with a single, stable attractor $\boldsymbol{\xi}^a$ visualized as a star (a). They can be globally collinear when the attractor is infinitely far away (b). Conversely, dynamics might be globally defined but only locally straight in a subdomain, visualized as the dark gray region (c).

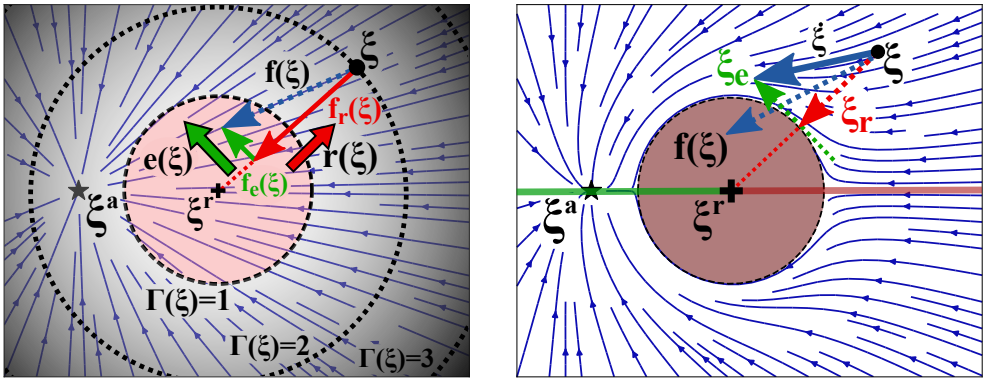
3.3 Obstacle Description

Environment representations of robotic systems can take various forms, ranging from sampled point clouds to occupancy grids. However, such sample-based representations often have an increased computational cost as the environmental resolution becomes finer. Yet, high resolutions are required in cluttered spaces. We adopt an implicit description to mitigate this challenge, employing an analytical function $\Gamma(\boldsymbol{\xi})$.

3.3.1 Star Shaped Obstacles

In this work, the obstacles are limited to star shapes (or trees of stars). These are shapes for which a reference point exists, $\boldsymbol{\xi}^r$, such that any line starting from this point intersects the surface exactly once. The formal definition has been proposed as (Brunn, 1913; Hansen et al., 2020).

Definition 3.3.1 (Starshape)**.** An obstacle is considered star-shaped if there exists a point $\boldsymbol{\xi}^r$ with $\Gamma(\boldsymbol{\xi}^r) < 1$ such that every point on the boundary or in the interior set of the obstacle, i.e., $\boldsymbol{\xi} \in \mathcal{X}^i$



(a) Initial dynamics (b) Avoidance through modulation

Figure 3.2: The initial system $\boldsymbol{f}(\xi)$ is decomposed in tangent $\boldsymbol{e}(\xi)$ and reference direction $\boldsymbol{r}(\xi)$ (a). The increased value of the distance function is visualized with increased shading. The individual stretching along tangents and compression in reference direction allows to safely avoid the obstacle with $\dot{\xi}$ (b). (Images are taken from L. Huber, Billard, and J.-J. Slotine, 2019.)

and $\mathcal{X}^i : \boldsymbol{\xi} \in \mathbb{R}^N, \Gamma(\boldsymbol{\xi}) \leq 1$, is connected to $\boldsymbol{\xi}^r$ by a line segment $l(\boldsymbol{\xi}^r, \boldsymbol{\xi})$ contained within $\mathcal{X}^i$.

The set of all such points $\boldsymbol{\xi}^r$ is referred to as the kernel of $\mathcal{X}^i$ and denoted as:

$$\ker(\mathcal{X}^i) = \left\{ \boldsymbol{\xi}^r \in \mathcal{X}^i : l(\boldsymbol{\xi}^r, \boldsymbol{\xi}) \subset \mathcal{X}^i, \ \forall \boldsymbol{\xi} \in \mathcal{X}^i \right\} \tag{3.4}$$

The point $\boldsymbol{\xi}^r$ refers to the *reference point*, as proposed in (L. Huber, Billard, and J.-J. Slotine, 2019), and represented by a cross symbol +. [1]

Using, the reference point $\boldsymbol{\xi}^r$ we can define the reference direction $\boldsymbol{r} : \mathbb{R}^N \setminus \boldsymbol{\xi}^r \rightarrow \{\boldsymbol{r} \in \mathbb{R}^N : \|\boldsymbol{r}\| = 1\}$:

$$\boldsymbol{r}(\boldsymbol{\xi}) = \left(\boldsymbol{\xi} - \boldsymbol{\xi}^r\right) / \|\boldsymbol{\xi} - \boldsymbol{\xi}^r\|. \quad \forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^r \tag{3.5}$$

Additionally, we introduce the local radius $R(\boldsymbol{\xi})$, specifying the distance from the center point to the surface in each direction. It is given as a function of the local boundary point $\boldsymbol{\xi}^b$ as:

$$R(\boldsymbol{\xi}) = \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \qquad \text{with} \quad \boldsymbol{\xi}^b = b\boldsymbol{r}(\boldsymbol{\xi}) + \boldsymbol{\xi}^r \quad \text{such that} \quad b > 0, \ \boldsymbol{\xi}^b \in \mathcal{X}^b \tag{3.6}$$

[1] The reference point is in literature sometimes denoted *kernel point*.

3.3.2 Distance Function

As proposed by S. M. Khansari-Zadeh and Billard, 2012, a continuous distance function $\Gamma(\boldsymbol{\xi}) : \mathbb{R}^d \setminus \mathcal{X}^i \mapsto \mathbb{R}_{\geq 1}$ is defined around each obstacle and divides the space into three regions:

$$\begin{aligned} &\text{Exterior points:} && \mathcal{X}^e = \{\boldsymbol{\xi} \in \mathbb{R}^d : \Gamma(\boldsymbol{\xi}) > 1\} \\ &\text{Boundary points:} && \mathcal{X}^b = \{\boldsymbol{\xi} \in \mathbb{R}^d : \Gamma(\boldsymbol{\xi}) = 1\} \\ &\text{Interior points:} && \mathcal{X}^i = \{\boldsymbol{\xi} \in \mathbb{R}^d \setminus (\mathcal{X}^e \cup \mathcal{X}^b)\} \end{aligned} \tag{3.7}$$

By construction, $\Gamma(\cdot)$ increases monotonically (C^1-smoothness) with increasing distance from the center along positive reference direction $\boldsymbol{r}(\boldsymbol{\xi})$.

In situations of uncertainties of the obstacles' position and shape, as might be the result of discrete sensor reading, an additional margin region can be useful:

$$\text{Margin exterior points:} \quad \mathcal{X}^g = \{\boldsymbol{\xi} \in \mathbb{R} : \boldsymbol{\xi} - (R^{\text{robo}} + D^{\text{gap}})\boldsymbol{r}(\boldsymbol{\xi}) \in \mathcal{X}^e\} \tag{3.8}$$

where $D^{\text{gap}} \in \mathbb{R}_{>0}$ is the margin distance, and $R^{\text{robo}} \in \mathbb{R}_{>0}$ the robot radius.

The distance function divides the space into colliding and non-colliding regions. This description of save regions resembles the concept of Control Barrier Functions as often used in MPC. (see Sec.) Conversely to that approach, this body of work focuses on the convergence of the avoidance dynamics and does not consider dynamic constraints.

Multiple possible choices of distance functions are presented in the remainder of the subsection, which are then used in the obstacle avoidance algorithm described in Section 3.4.

Proportional Distance

The proportional distance function is designed to scale with the shape of the obstacle. A larger obstacle has a higher importance than a smaller obstacle. It is proportional to the local radius $R(\boldsymbol{\xi})$, given in (3.6), given by:

$$\Gamma(\boldsymbol{\xi}) = \left(\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| / R(\boldsymbol{\xi})\right)^{2p} \quad \forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^r \tag{3.9}$$

The isolines created by the distance function have the same shape as the base obstacle. However, it has the effect that positions are perceived to be closer along the longest axes, while along the shortest axes, the points are perceived further away.

Uniform Distance

A uniform evolution of the distance function can be achieved by utilizing the distance to the surface:

$$\Gamma(\boldsymbol{\xi}) = \|\boldsymbol{\xi} - \boldsymbol{\xi}^b\| / d_0 + 1 \quad \forall \boldsymbol{\xi} \in \mathscr{X}^e \tag{3.10}$$

where $d_0 \in \mathbb{R}_{>0}$ is a scaling factor, which modulates the obstacle's influence, and the boundary point $\boldsymbol{\xi}^b(\boldsymbol{\xi})$, is evaluated as proposed in (3.6). This distance function has the advantage of having a distance function that increases uniformly in all directions. Moreover, the scaling factor d_0 can be set to give more or less importance to certain obstacles.

Limit Distance

Since the avoidance method considers all obstacles that have a finite distance value, it can be desirable to distance threshold $d^{\max} \in \mathbb{R}$, such that:

$$\lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^b\| \to d^{\max}} \Gamma(\boldsymbol{\xi}) \to \infty \quad \text{and} \quad \lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^b\| \to 0} \Gamma(\boldsymbol{\xi}) = 1 \tag{3.11}$$

For example, the distance function can be set as follows:

$$\Gamma(\boldsymbol{\xi}) = \begin{cases} d^{\max} / (d^{\max} - \|\boldsymbol{\xi} - \boldsymbol{\xi}^b\|) & \text{if } \|\boldsymbol{\xi} - \boldsymbol{\xi}^b\| < d^{\max} \\ \infty & \text{otherwise} \end{cases} \tag{3.12}$$

Navigation functions for obstacle avoidance (D. E. Koditschek and Rimon, 1990; Paternain, D. E. Koditschek, and Ribeiro, 2017) and vector field avoidance algorithms (Panagou, 2014; Yao, B. Lin, et al., 2022) often require parameter tuning based on the obstacle distribution to ensure the absence of local minima. This is required because overlapping regions of influence can lead to local minima. While dynamical system-based obstacle avoidance (as introduced in the next section) is not prone to this problem, such a distance field is advantageous to limit the computational cost in dense regions.

3.4 Obstacle Avoidance through Modulation

Dynamic obstacle avoidance is a fundamental aspect of robotic navigation, and it can be achieved through a dynamic modulation matrix $\boldsymbol{M}(\boldsymbol{\xi})$ applied to the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$, as described in (L. Huber, Billard, and J.-J. Slotine, 2019):

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi}) \boldsymbol{f}(\boldsymbol{\xi}) \qquad \text{with} \quad \boldsymbol{M}(\boldsymbol{\xi}) = \boldsymbol{E}(\boldsymbol{\xi}) \boldsymbol{D}(\boldsymbol{\xi}) \boldsymbol{E}(\boldsymbol{\xi})^{-1} \tag{3.13}$$

The modulation matrix $\boldsymbol{M}(\boldsymbol{\xi})$ is composed of two main components: the basis matrix $\boldsymbol{E}(\boldsymbol{\xi})$ and

the diagonal matrix $\boldsymbol{D}(\boldsymbol{\xi})$. The basis matrix decomposes the initial velocity $\boldsymbol{f}(\boldsymbol{\xi})$ into the local frame, where it is stretched using the diagonal matrix.

The basis matrix is composed as follows:

$$\boldsymbol{E}(\boldsymbol{\xi}) = [\boldsymbol{r}(\boldsymbol{\xi}) \;\; \boldsymbol{e}_1(\boldsymbol{\xi}) \;\; .. \;\; \boldsymbol{e}_{N-1}(\boldsymbol{\xi})] \tag{3.14}$$

It consists of the reference direction as defined in (3.5) and a set of tangent vectors $e_i(\boldsymbol{\xi})$, with $i \in 1, .., N-1$, forming an orthonormal basis orthogonal to the normal direction $\boldsymbol{n}(\boldsymbol{\xi})$. The normal direction $\boldsymbol{n}(\boldsymbol{\xi})$ is computed as the derivative of the distance across space,

$$\boldsymbol{n}(\boldsymbol{\xi}) = d\Gamma(\boldsymbol{\xi}) / d\boldsymbol{\xi} \tag{3.15}$$

Importantly, because the obstacle is required to be star-shaped, as defined in Definition 3.3.1, the following property holds:

$$\exists \boldsymbol{\xi}^r \in \mathscr{X}^i \quad : \; \langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0 \quad \forall \boldsymbol{\xi} \in \mathscr{X}^e \tag{3.16}$$

This property guarantees that the basis matrix $\boldsymbol{E}(\boldsymbol{\xi})$ has full rank and is invertible.

The diagonal matrix $\boldsymbol{D}(\boldsymbol{\xi})$ is defined as follows:

$$\boldsymbol{D}(\boldsymbol{\xi}) = \mathbf{diag}\left(\lambda^r(\boldsymbol{\xi}), \lambda^e(\boldsymbol{\xi}), \ldots, \lambda^e(\boldsymbol{\xi})\right) \tag{3.17}$$

The values λ_r and λ_e stretch and shrink the velocity along the reference and tangent directions, respectively. $\lambda_{(\cdot)}$ are often referred to as eigenvalues, as they represent the Eigen-decomposition of the modulation $\boldsymbol{M}(\boldsymbol{\xi})$ (S. M. Khansari-Zadeh and Billard, 2012). These eigenvalues need to ensure that the velocity towards the obstacle is reduced when approaching the surface and that the velocity is influenced little when far away from the obstacle.

The eigenvalues are defined as follows (L. Huber, Billard, and J.-J. Slotine, 2019):

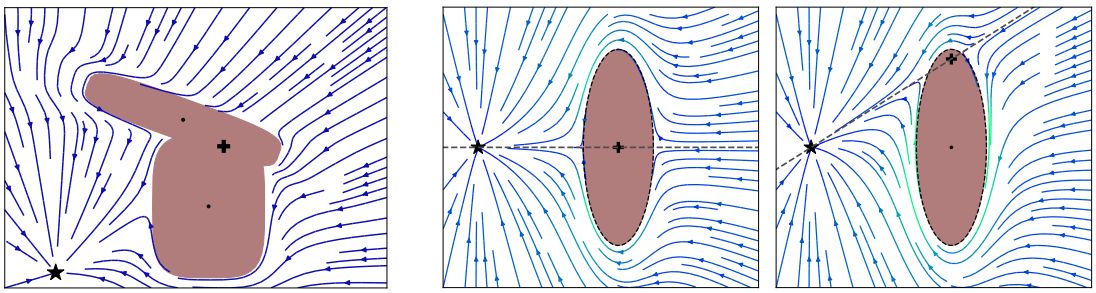
$$\lambda^r(\boldsymbol{\xi}) = 1 - 1/\Gamma(\boldsymbol{\xi})^{1/\rho} \qquad \lambda^e(\boldsymbol{\xi}) = 1 + 1/\Gamma(\boldsymbol{\xi})^{1/\rho} \tag{3.18}$$

with the reactivity factor $\rho \in \mathbb{R}_{>0}$ and the distance function $\Gamma(\boldsymbol{\xi})$ from (3.9). In this work, we use $\rho = 1$.

3.4.1 Dynamic Obstacles

In scenarios involving moving obstacles, the modulation is performed in the relative reference frame, which is then transformed back to the inertial frame, as expressed in equation (S. Khansari-Zadeh, 2012):

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi}) \left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\tilde{\boldsymbol{\xi}}} \right) + \dot{\tilde{\boldsymbol{\xi}}} \quad \text{with } \dot{\tilde{\boldsymbol{\xi}}} = \dot{\boldsymbol{\xi}}^{L,o} + \dot{\boldsymbol{\xi}}^{R,o} \times \tilde{\boldsymbol{\xi}} \tag{3.19}$$



(a) Starshaped Obstacle (b) Reference point at center (c) Reference point at the top

Figure 3.3: Placing the reference point (black cross) in the kernel space of the obstacles ensures avoidance of an obstacle of all but one trajectory (a). The reference point indicates how the vector field is split, for example, at the center of the obstacle (b), or the top of the obstacle (c). Green shading of the trajectory lines in (b) and (c) indicate higher velocities. (Images are taken from L. Huber, Billard, and J.-J. Slotine, 2019.)

where the linear velocity $\dot{\boldsymbol{\xi}}^{L,o}$ and the angular velocity $\dot{\boldsymbol{\xi}}^{R,o}$ are with respect to the obstacle's reference point $\boldsymbol{\xi}^r$, and $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}^r$ is the relative position. It's important to note that avoiding moving obstacles involves more than just matrix multiplication. Hence, stable points such as local minima or saddle points, can occur at different positions even with full-rank of the modulation matrix $\boldsymbol{M}(\boldsymbol{\xi})$.

3.4.2 Multiple Obstacles

When dealing with multiple obstacles, the avoidance velocity is calculated as a weighted mean of the modulated velocities $\dot{\boldsymbol{\xi}}^o$ obtained by applying (3.13) to each obstacle $o = 1..N^{\text{obs}}$. Both the magnitude $|\dot{\boldsymbol{\xi}}^o|$ and direction $\boldsymbol{n}^{\dot{\boldsymbol{\xi}}^o}$ are separately considered (Fig. 3.4).

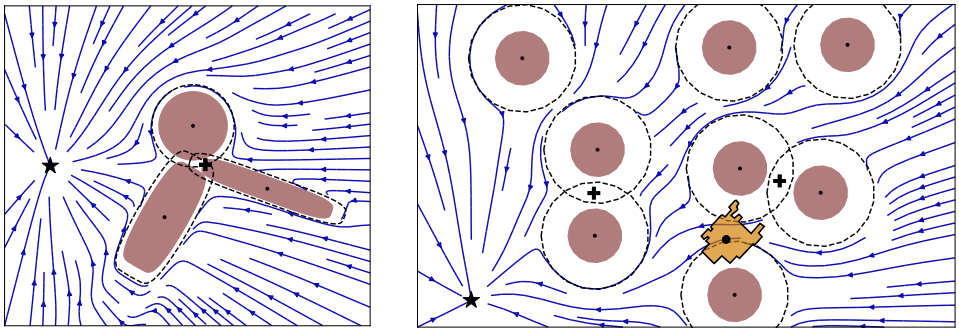
The importance weight is designed to balance the effect of each obstacle and ensures that the influence of other obstacles vanishes at the boundary of each obstacle. This weight, denoted as w^o, is inversely proportional to the distance $\Gamma^o(\boldsymbol{\xi}) - 1$:

$$w^o(\boldsymbol{\xi}) = \frac{\prod_{i \neq o}^{N^o} \left(\Gamma^i(\boldsymbol{\xi}) - 1\right)}{\sum_{k=1}^{N^o} \prod_{i \neq k}^{N^o} \left(\Gamma^i(\boldsymbol{\xi}) - 1\right)} \quad o = 1..N^o \tag{3.20}$$

The magnitude is obtained as a weighted sum of the individual velocities:

$$\|\dot{\tilde{\boldsymbol{\xi}}}\| = \sum_{o=1}^{N^o} w^o \|\dot{\boldsymbol{\xi}}^o\| \tag{3.21}$$

Directional aggregation is achieved by considering the directional weighted mean, as described in Section 4.1. These techniques allow for effective obstacle avoidance in scenarios with multiple



(a) Multiple intersecting obstacles (b) Multiple-distinct obstacles representing a crowd

Figure 3.4: The obstacle avoidance method can avoid multiple touching obstacles (a) as long as they share a common reference point (black cross). In cluttered dense environments, the methods navigate safely between the obstacles as long as they represent mutually distinct starshapes (b). (The images are taken from L. Huber, Billard, and J.-J. Slotine, 2019.)

obstacles, even when they intersect or are densely scattered (Figure 3.4).

3.5 Robot Kinematics

Robot kinematics describes the movement of a rigid-body robot in space. Where for this work, a robot consists of interconnected rigid links and joints, which can be either linear or rotational.

3.5.1 Forward Kinematics

Forward kinematics involves determining the position of the robot's end-effector, denoted as $\boldsymbol{\xi}$, based on the joint coordinates $\boldsymbol{q}$. This computation is achieved through a series of homogeneous transformation matrices $T \in \mathbb{R}^{4\times4}$:

$$\boldsymbol{T}_{0,N^{\text{links}}} = \prod_{i=0}^{N^{\text{links}}-1} \boldsymbol{T}_{i,i+1} \tag{3.22}$$

Here, the transformation matrices are represented as:

$$\boldsymbol{T}_{i,i+1} = \begin{bmatrix} & & & l_i \\ & \boldsymbol{R}(\boldsymbol{q}_i) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

The link lengths $l_i = \|\boldsymbol{j}_i - \boldsymbol{j}_{i+1}\|$ determine the distance between two joints $\boldsymbol{j}_i$ and $\boldsymbol{j}_{i+1}$. Furthermore, $\boldsymbol{R}(\boldsymbol{q}_i) \in SE(3)$ denotes the local rotation, describing the rotation from the joint i, to the

joint $i+1$:

$$\frac{\boldsymbol{j}_{i+1} - \boldsymbol{j}_i}{\|\boldsymbol{j}_{i+1} - \boldsymbol{j}_i\|} = \boldsymbol{R}(\boldsymbol{q}_i) \frac{\boldsymbol{j}_i - \boldsymbol{j}_{i-1}}{\|\boldsymbol{j}_i - \boldsymbol{j}_{i-1}\|} \tag{3.24}$$

3.5.2 Differential Kinematics

Differential kinematics allows us to compute the time derivative of the end-effector position $\dot{\boldsymbol{\xi}}$ with respect to the joint velocities $\dot{\boldsymbol{q}}$, assuming that the effector position is given by the forward kinematics function $\boldsymbol{\xi} = k(\boldsymbol{q})$. This is done using the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{N^j \times N^j}$:

$$\dot{\boldsymbol{\xi}} = \frac{\partial k(\boldsymbol{q})}{\partial t} = \frac{\partial k(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{t}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.25}$$

The Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{N^j \times N^j}$ is dependent on the robot's joint configuration $\boldsymbol{q}$.

In many robotic applications, it is desirable to control the robot in Cartesian space, specifying the desired end-effector velocity $\dot{\boldsymbol{\xi}}$. Consequently, the joint velocities $\dot{\boldsymbol{q}}$ must be computed using the inverse Jacobian, minimizing the joint velocities:

$$\begin{aligned} \underset{\boldsymbol{J}(\boldsymbol{q})}{\text{minimize}} \quad & \frac{1}{2}\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} \\ \text{subject to} \quad & \dot{\boldsymbol{q}} = \boldsymbol{J}^\dagger(\boldsymbol{q})\dot{\boldsymbol{\xi}} \end{aligned} \tag{3.26}$$

In cases where the number of joints N^j is equal to the number of end-effector states N^{ee}, the Jacobian is square, and assuming a suitable joint configuration, it is invertible. However, in overactuated systems where $N^{\text{ee}} < N^j$, there are more joints than necessary, and many possible joint velocities can achieve the desired motion. In such cases, the joint velocities are computed using the pseudoinverse of the Jacobian:

$$\dot{\boldsymbol{q}}^* = \boldsymbol{J}^\dagger\dot{\boldsymbol{\xi}} \qquad \text{with} \qquad \boldsymbol{J}^\dagger = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} \tag{3.27}$$

To handle singularities and ensure invertibility of $\boldsymbol{J}\boldsymbol{J}^T$, a damping factor $\lambda \in \mathbb{R}_{>0}$ can be introduced, resulting in the damped least squares inverse:

$$\boldsymbol{J}^\dagger = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I}\right)^{-1} \tag{3.28}$$

where $\boldsymbol{I} \in \mathbb{R}^{N^{\text{ee}} \times N^{\text{ee}}}$ is the identity matrix, and $\lambda \in \mathbb{R}_{>0}$ is a damping factor. The damping factor ensures invertibility and a higher value ensures smoother motion close to singularities. However, high damping values come at the cost of lower accuracy of following the desired velocity $\dot{\boldsymbol{\xi}}$. Hence, the damping factor is often set to a small value, such as $\lambda = 0.01$.

In cases where there are more degrees of freedom at the end effector than joint states, $N^{\text{ee}} > N^{\text{j}}$, the robot is underactuated, and the desired velocity can only be approximated. However, this

work does not cover the nuances of underactuated robots.

3.6 Robot Dynamics

In human-robot collaboration, it is important to understand the dynamics of a robot system, particularly when it is controlled using torque commands. This section describes robot dynamics and presents important control strategies.

Rigid Body Dynamics

A force-controlled system is subject to acceleration effects, inertia, and external disturbances. The general rigid-body dynamics with respect to the end-effector position $\boldsymbol{\xi}$, can be expressed as follows:

$$\boldsymbol{\tau_c} = \mathscr{M}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathscr{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}})\dot{\boldsymbol{\xi}} + \boldsymbol{g}(\boldsymbol{\xi}) = \boldsymbol{\tau_e} + \boldsymbol{\tau_c} \tag{3.29}$$

where we have the mass matrix of the system $\mathscr{M}(\boldsymbol{\xi}) \in \mathbb{R}^{N\times N}$, the Coriolis matrix $\mathscr{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \in \mathbb{R}^{N}$, the gravity vector $\boldsymbol{g}(\boldsymbol{\xi}) \in \mathbb{R}^{N}$), the control torque $\boldsymbol{\tau_c} \in \mathbb{R}^{N}$, and the external torque, also referred as disturbance, $\boldsymbol{\tau_e} \in \mathbb{R}^{N}$.

3.6.1 Impedance Controller Design

The Impedance Controller is a widely employed strategy for guiding torque-controlled robot arms. In this control scheme, the robot continuously monitors the motion of its end effector $\boldsymbol{\xi}$ and computes the necessary joint torques and forces to generate the control torque $\boldsymbol{\tau}_c$. The controller establishes a linear relationship between the control torque $\boldsymbol{\tau_c}$ and the desired deviations in end-effector position $\boldsymbol{\xi}$, velocity $\dot{\boldsymbol{\xi}}$, and acceleration $\ddot{\boldsymbol{\xi}}$:

$$\boldsymbol{\tau}_c = \boldsymbol{J}^{\dagger}\left(\eta\left(\boldsymbol{q}, \dot{\boldsymbol{\xi}}\right) - \left(\mathscr{M}\left(\ddot{\boldsymbol{\xi}} - \ddot{\boldsymbol{\xi}}_a\right) + \mathscr{D}\left(\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_a\right) + \mathscr{K}\left(\boldsymbol{\xi} - \boldsymbol{\xi}_a\right)\right)\right) \tag{3.30}$$

where $\boldsymbol{\xi}_a$ is the target position, which leads to target velocity $\dot{\boldsymbol{\xi}}_a$ and target acceleration $\ddot{\boldsymbol{\xi}}_a$. The external forces $\eta\left(\boldsymbol{q}, \dot{\boldsymbol{\xi}}\right)$ are predicted using the robot's model and actively compensated for. Often it includes terms for Coriolis acceleration $\mathscr{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}})$ and gravity force $\boldsymbol{g}(\boldsymbol{\xi})$. It's worth noting that Coriolis forces can be neglected when assuming relatively low robot velocities.

3.6.2 Passive Damping Controller

The passive damping controller, proposed by Kronander and Billard, 2015, offers a method for generating control forces based on a velocity field. This controller excels at selective disturbance rejection by deploying adaptive damping depending on the direction of the desired motion.

Typically, the controller is configured with high damping along the direction of motion to ensure

rapid convergence of the robot's velocity towards the desired value, resulting in excellent tracking performance. In contrast, it exhibits high compliance in the direction perpendicular to the motion, allowing for more flexible behavior and higher resistance to external forces.

The passive control force is calculated as follows:

$$\boldsymbol{\tau_c} = \boldsymbol{g}(\boldsymbol{\xi}) + \mathscr{D}(\boldsymbol{\xi})\left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}\right) \tag{3.31}$$

This control law includes a gravity compensation term $\boldsymbol{g}(\boldsymbol{\xi}) \in \mathbb{R}^N$ and damps the difference between the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$ and the actual velocity $\dot{\boldsymbol{\xi}}$.

The positive definite damping matrix $\mathscr{D}(\boldsymbol{\xi}) \in \mathbb{R}^{N\times N}$ is defined as:

$$\mathscr{D}(\boldsymbol{\xi}) = \mathscr{Q}(\boldsymbol{\xi})\mathscr{S}(\boldsymbol{\xi})\mathscr{Q}(\boldsymbol{\xi})^{-1} \tag{3.32}$$

where $\mathscr{Q}(\boldsymbol{\xi}) = \left[\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_N\right]$ is an orthonormal basis matrix, of which the first vector is pointing in the desired direction of motion

$$\boldsymbol{q}_1(\boldsymbol{\xi}) = \boldsymbol{q}_1^f(\boldsymbol{\xi}) = \dot{\boldsymbol{\xi}} / \|\dot{\boldsymbol{\xi}}\| \tag{3.33}$$

The diagonal matrix $\mathscr{S}(\boldsymbol{\xi}) \in \mathbb{R}^{N\times N}$ consists of damping factors $s_i \in \mathbb{R}_{>0}$ in the corresponding directions $i = 1..N$. This design allows for separate control of damping along the direction of motion and perpendicular to it. Typically, a relatively high value is assigned to the first damping factor to enhance consistency with the desired velocity, while the damping in other directions is set lower to enable compliance perpendicular to the motion, i.e., $s_i / s_1 \ll 1,\ i = 2..N$.

Damping control is a simplified version of the impedance controller with time-varying control parameters.

3.7 Performance Guarantees

In robot control, the assurance of reliable performance is of great importance. This section presents multiple performance guarantees, which are fundamental to designing robotic systems.

3.7.1 Collision Avoidance

The primary goal of the algorithms presented in this thesis is collision avoidance. As ensuring that a robot navigates without colliding with obstacles is essential for safe and effective operation. To provide performance guarantees in this regard, we turn to the Bony-Bezis theorem (Bony, 1969; Brezis, 1970), also known as the Nagumo theorem (Nagumo, 1942). The theorem establishes a crucial link between maintaining motion within a specific region and collision avoidance.

Theorem 3.7.1 (Bony-Bezis-Theorem)**.** *Let $\mathscr{X}^e$ be a closed subset of a manifold $\mathscr{X}$ and $\boldsymbol{f}(\boldsymbol{\xi})$ be a Lipschitz-continuous vector field on $\mathscr{X}$. The following conditions are equivalent:*

- *every integral curve of $\boldsymbol{f}(\boldsymbol{\xi})$ starting in $\S^e$ remains in this region, i.e., $\boldsymbol{\xi} \in \S^e,\ t = 0..\infty$.*
- *$\left\langle \boldsymbol{f}(\hat{\boldsymbol{\xi}}), \boldsymbol{n}(\hat{\boldsymbol{\xi}}) \right\rangle \geq 0$ at every exterior point $\hat{\boldsymbol{\xi}}$ of $\S^e$, and the normal vector $\boldsymbol{n}(\hat{\boldsymbol{\xi}})$ to the boundary of $\S^e$, pointing into the set.*

In the context of obstacle avoidance, $\mathscr{X}^e$ represents the obstacle-free space:

$$\S^e = \left\{ \boldsymbol{\xi} : \gamma_o(\boldsymbol{\xi}) \geq 1,\ \forall\, o = 1..n^{\text{obs}} \right\} \tag{3.34}$$

3.7.2 Lyapunov Analysis

Lyapunov theory allows analyzing the stability and convergence of dynamical systems (J.-J. E. Slotine et al., n.d.). It provides insights into how systems behave over time.

Consider an autonomous (nonlinear) dynamical system $\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi})$ where $\boldsymbol{\xi} \in \mathscr{X} \subset \mathbb{R}^N$ denotes the state vector, and a continuous vector field $\boldsymbol{f} : \mathscr{D} \to \mathbb{R}^N$. The system has an equilibrium point $\boldsymbol{\xi}^a \in \mathscr{X}$ such that $\boldsymbol{f}(\boldsymbol{\xi}^a) = \boldsymbol{0}$.

- The equilibrium $\boldsymbol{\xi}^a$ is *Lyapunov stable*, if for every $\epsilon > 0$ there exists a $\delta > 0$ such that $\|\boldsymbol{\xi}(0) - \boldsymbol{\xi}^a\| < \delta$ then $\|\boldsymbol{\xi}(t) - \boldsymbol{\xi}^a\| < \epsilon$ for all $t \geq 0$.
- The equilibrium $\boldsymbol{\xi}^a$ is *asymptotically stable*, if it is Lyapunov stable and there exists a $\delta > 0$ such that $\|\boldsymbol{\xi}(0) - \boldsymbol{\xi}^a\| < \delta$ then $\lim_{t\to\infty} \|\boldsymbol{\xi}(t) - \boldsymbol{\xi}^a\| = 0$. (In such a case the equilibrium point $\boldsymbol{\xi}^a$ is also called *attractor* of the dynamical system.)
- The equilibrium $\boldsymbol{\xi}^a$ is *exponentially stable* if it there exists $\alpha > 0$, $\beta > 0$, $\delta > 0$ such that if $\|\boldsymbol{\xi}_0 - \boldsymbol{\xi}^a\| < \delta$ then $\|\boldsymbol{\xi} - \boldsymbol{\xi}^a\| \leq \alpha \|\boldsymbol{\xi} - \boldsymbol{\xi}^a\| e^{-\beta t}$ for all $t \geq 0$.

Lyapunov stability can be analyzed using a Lyapunov function, denoted as $V(\boldsymbol{\xi})$:

Theorem 3.7.2 (Lyapunov Stability)**.** *The system $\boldsymbol{f} : \mathscr{D} \to \mathbb{R}^N$ is called Lyapunov stable, if there exist a Lyapunov function $V : D \to \mathbb{R}$ with respect to the stable point $\boldsymbol{\xi}^a = \boldsymbol{0}$ such that:*

- *The function is zero only at the stable point: $V(\boldsymbol{\xi}) = 0$ if and only if $\boldsymbol{\xi} = 0$*
- *The function is positive definite everywhere else: $V(\boldsymbol{\xi}) > 0\ \forall \boldsymbol{\xi} \neq 0$*
- *The time derivative has a negative derivative away from the stable point: $\dot{V}(\boldsymbol{\xi}) = \frac{d}{dt} V(\boldsymbol{\xi}) = \nabla V(\boldsymbol{\xi}) \boldsymbol{f}(\boldsymbol{\xi}) \leq 0\ \forall \boldsymbol{\xi} \neq 0$*

The requirement of the stable point being at the origin does not constrain the system, as it can be fulfilled through coordinate transformations. Furthermore, the system is asymptotically stable if $\dot{V}(\boldsymbol{\xi}) < 0 \ \forall \boldsymbol{\xi} \neq 0$.

This fundamental theorem of Lyapunov Stability asserts that a system is Lyapunov stable if and only if a suitable Lyapunov function exists. Despite the existence of Lyapunov functions for any stable systems, their discovery can pose challenges, as there is no systematic approach to finding them.

3.7.3 Contraction Theory

Contraction theory is a powerful framework for analyzing the stability of dynamical systems, which relies on the concept of a contraction metric (Lohmiller and J.-J. E. Slotine, 1998). In essence, it investigates the local convergence between neighboring trajectories, as depicted in Figure 3.5. When all neighboring trajectories tend to approach one another within an invariant region $\mathscr{X}$, that region is considered contracting, and convergence towards one trajectory is guaranteed.

Contraction theory uses the notion of distance between two neighboring trajectories, denoted as $\delta\boldsymbol{\xi}$:

$$\frac{d}{dt}\left(\delta\boldsymbol{\xi}^T\delta\boldsymbol{\xi}\right) = 2\delta\boldsymbol{\xi}^T\delta\dot{\boldsymbol{\xi}} = 2\delta\boldsymbol{\xi}^T\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\xi}}\delta\boldsymbol{\xi} \leq \lambda^{\max}\boldsymbol{\xi}^T\delta\boldsymbol{\xi} \leq -\beta\delta\boldsymbol{\xi}^T\delta\boldsymbol{\xi} \quad \Rightarrow \quad \|\delta\boldsymbol{\xi}\| \leq \|\delta\boldsymbol{\xi}_0\|e^{-\beta t} \tag{3.35}$$

$\beta \in \mathbb{R}_{>0}$ represents the contraction rate, and $\lambda^{\max}$ is the maximum eigenvalue of the Jacobian matrix $1/2(\partial \boldsymbol{f}/\partial \boldsymbol{\xi} + \partial \boldsymbol{f}/\partial \boldsymbol{\xi}^T)$. A system is considered contracting if its Jacobian is negative definite. It is called semi-contracting if the Jacobian is negative semi-definite, and indifferent when the Jacobian is skew-symmetric.

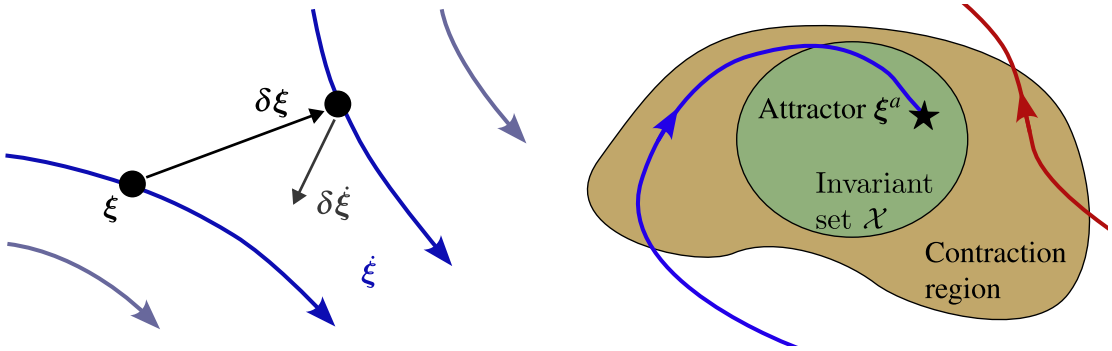
Contraction Metric

For a more general analysis, Lohmiller and J.-J. E. Slotine, 1998 introduced a coordinate transformation:

$$\delta\boldsymbol{\gamma} = \boldsymbol{\Theta}\delta\boldsymbol{\xi} \quad \Rightarrow \quad \delta\boldsymbol{\gamma}^T\delta\boldsymbol{\gamma} = \delta\boldsymbol{\xi}^T\boldsymbol{M}\delta\boldsymbol{\xi} \tag{3.36}$$

This transformation defines a Riemann space, with $\boldsymbol{\Theta}(\boldsymbol{\xi}, t) \in \mathbb{R}^{N\times N}$ being a smoothly differentiable square matrix and $\boldsymbol{M}(\boldsymbol{\xi}, t) = \Theta^T\Theta$ the contraction metric. The metric is assumed to be uniformly positive definite, where exponential convergence of $\delta\boldsymbol{\gamma} \to \boldsymbol{0}$ implies exponential convergence of $\delta\boldsymbol{\xi} \to \boldsymbol{0}$.

The coordinate transformation from $\boldsymbol{\xi}$ to $\boldsymbol{\gamma}$ may not always be explicitly integrable. However, the transformed distance $\delta\boldsymbol{\gamma}$ is defined, sufficient for the contraction analysis.



(a) Two neighbouring trajectories (b) Contracting spaces

Figure 3.5: Contraction analysis involves monitoring the evolution of the distance between two neighboring trajectories, denoted as $\delta\boldsymbol{\xi}$ (a). If the distance decreases within a given region, that region is characterized as *contracting*. Moreover, when an invariant set $\mathcal{X}$ is identified as contracting and includes a stable point $\boldsymbol{\xi}^a$ the dynamical system exhibits asymptotic stability within that region (b). It follows that all trajectories converge towards the attractor, such as the blue trajectory. However, trajectories that merely enter the contracting region (see the example in red) may still exit it at a later stage.

The contraction in the transformed space $\delta\boldsymbol{\gamma}$ is studied as follows:

$$\frac{d}{dt}\left(\partial\boldsymbol{\gamma}^T\delta\boldsymbol{\gamma}\right) = 2\delta\boldsymbol{\gamma}^T\frac{d}{dt}\partial\boldsymbol{\gamma} = 2\delta\boldsymbol{\gamma}^T\left(\dot{\Theta}\delta\boldsymbol{\xi} + \Theta\delta\dot{\boldsymbol{\xi}}\right) = 2\delta\boldsymbol{\gamma}^T\left(\dot{\Theta} + \Theta\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}}\right)\Theta^{-1}\delta\boldsymbol{\gamma} \tag{3.37}$$

Equivalently, in the original coordinates $\delta\boldsymbol{\xi}$, this yields:

$$\frac{d}{dt}\left(\partial\boldsymbol{\xi}^T\boldsymbol{M}\delta\boldsymbol{\xi}\right) = \delta\boldsymbol{\xi}^T\left(\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}}^T\boldsymbol{M} + \dot{\boldsymbol{M}} + \boldsymbol{M}\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}}\right)\delta\boldsymbol{\xi} \tag{3.38}$$

Therefore, contraction can be proven by demonstrating one of the following equivalent conditions:

$$\exists\beta_\Theta \in \mathbb{R}_{>0} : \left(\dot{\Theta} + \Theta\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}}\right)\Theta^{-1} \leq -\beta_\Theta \quad \Leftrightarrow \quad \exists\beta_M \in \mathbb{R}_{>0} : \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}}^T\boldsymbol{M} + \dot{\boldsymbol{M}} + \boldsymbol{M}\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{\xi}} \leq -\beta_M\boldsymbol{M} \tag{3.39}$$

A contracting system ensures that all trajectories converge to a single trajectory exponentially. However, contraction does not give any guarantees concerning the system's stability.

Asymptotic Stability

Assuming that a dynamical system exhibits contraction within an invariant set $\mathcal{X}$ and possesses a stable point $\boldsymbol{\xi}^a \in \mathcal{X}$ (as illustrated in Figure 3.5), it follows that all trajectories passing this set converge to this attractor. Consequently, the system is deemed asymptotically stable within the region $\mathcal{X}$.

Contraction theory offers an alternative to Lyapunov theory for proving convergence properties (and vice versa for other systems). Moreover, it can be applied to systems lacking stationary points, ensuring convergence to a specific limit cycle or other attractors.

3.7.4 Passivity Analysis

Considering varying control parameters carefully is crucial, as such a design can inject energy into a system, potentially leading to unstable behavior and damaging the system (Ferraguti, Secchi, and Fantuzzi, 2013). In human-robot interaction, the robot faces external disturbances of unknown nature. To achieve stable and bounded behavior in the face of such disturbances, *passivity* analysis is a valuable tool. By employing passivity analysis, the control system can be designed to maintain stable responses (bounded system) in the presence of any external force (bounded input).

This has been summarized by following theorem (Willems, 1972; Sepulchre, Jankovic, and Kokotovic, 2012):

Definition 3.7.1 (Passivity)**.** A dynamical system with input $u \in \mathscr{U}$ and output $y \in \mathscr{Y}$ is passive with respect to the supply rate $s : \mathscr{U} \times \mathscr{Y} \to R$ if, for any $u : \mathbb{R}_{>0} \to \mathscr{U}$ and any time $t^* \geq 0$ the following is satisfied

$$\int_0^{t^*} s\big(u(t), y(t)\big)\, d\tau \geq S(t^*) - S(0) \tag{3.40}$$

where $S(t) \in \mathbb{R}_{\geq 0}$ is the storage function.

An important property of passivity is, that *a negative feedback loop consisting of two passive systems is passive* Sepulchre, Jankovic, and Kokotovic, 2012. Hence, if the controller is passive, its application to a (passive) environment-hardware system results in a passive system.

Note, that every passive system is bounded-input, bounded-output (BIBO) stable. However, not every BIBO stable system is passive.

4 Developed Tools

Throughout this thesis, multiple mathematical tools have been developed to help compute complex nonlinear vector fields and apply vector fields to robot arms with multiple degrees of freedom. Since they represent multi-purpose tools that can be combined with multiple algorithms developed in this thesis, they have been collected in this chapter.

4.1 Directional Weighted Mean

Publication Note

The material in this section was originally published together with Chapter 5 in:
L. Huber, J.-J. Slotine, and A. Billard (2022a). "Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds". In: *IEEE Transactions on Robotics*

The weighted summation of vectors may yield a zero-sum result, such as when two opposing vectors bear equal weight, potentially leading to undesirable local minima within the free space. To mitigate this issue and enable the combination of multiple vectors without generating spurious attractors, we propose a generalization of the directional weighted summation method originally introduced in (L. Huber, Billard, and J.-J. Slotine, 2019).

This extended approach proves valuable when combining the dynamics generated by multiple obstacles. Furthermore, this section expands upon this method's application to determining the closest distance between convex obstacles and identifying a point within their intersection region. Leveraging the directional approach allows us to devise an algorithm with low computational complexity, which is crucial in environments with dynamic obstacles.

Let us first define a general direction $\boldsymbol{v}$ given as:

$$\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\| = 1\} \tag{4.1}$$

The transformation into direction space $\mathscr{K}$ is given by:

$$\mathscr{K} = \{\kappa \in \mathbb{R}^{N-1} \,:\, \|\kappa\| < \pi\} \tag{4.2}$$

The direction space is with respect to a base vector $\boldsymbol{b}$, which is the first column of the orthonormal transformation matrix $\boldsymbol{B}$. This allows the transformation into a new basis:

$$\hat{\boldsymbol{v}}_i = \boldsymbol{B}^T \boldsymbol{v}_i \tag{4.3}$$

The magnitude of the transformed vector in direction space is equal to the angle between the original vector and the reference vector. The transformation of the initial vector $\boldsymbol{v}_i$ in the direction-space is:

$$\kappa_i(\boldsymbol{b}) = \boldsymbol{k}(\boldsymbol{v}_i, \boldsymbol{b}) = \begin{cases} \arccos\left(\hat{\boldsymbol{v}}_{i[1]}\right) \frac{\hat{\boldsymbol{v}}_{i[2:]}}{\|\hat{\boldsymbol{v}}_{i[2:]}\|} & \text{if } \hat{\boldsymbol{v}}_{i[1]} \neq 1 \\ \boldsymbol{0} & \text{if } \hat{\boldsymbol{v}}_{i[1]} = 1 \end{cases} \tag{4.4}$$

The mean is evaluated as a function of the weight w_i of all N^v vectors:

$$\bar{\kappa} = \sum_{i=1}^{N^v} w_i \kappa_i \tag{4.5}$$

The mapping into original space is evaluated as:

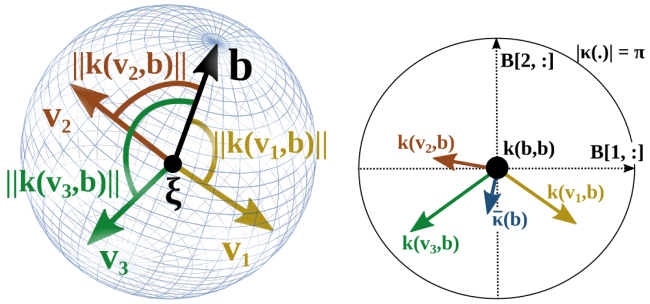
$$\bar{\boldsymbol{v}}(\bar{\kappa}) = \begin{cases} \boldsymbol{B} \begin{bmatrix} 1 & 0 & .. & 0 \end{bmatrix}^T & \text{if } \|\bar{\kappa}\| = 0 \\ \boldsymbol{B} \begin{bmatrix} \cos(\|\bar{\kappa}\|) & \sin(\|\bar{\kappa}\|) \frac{\bar{\kappa}}{\|\bar{\kappa}\|} \end{bmatrix}^T & \text{otherwise} \end{cases} \tag{4.6}$$

In the two-dimensional case, this hyper-sphere is a line that represents the angle between the initial DS $\boldsymbol{f}(\xi)$ and the modulated DS $\dot{\xi}$. It has a magnitude strictly smaller than π, and the directional space is a vector space, where the weighted mean is taken (see for the three-dimensional case in Fig. 4.1).

Note, this vector extraction captures the rotation of $\boldsymbol{\xi}$ with respect to $\boldsymbol{b}$ and has proven to be useful for minima-free vector-summing. The relative rotation additionally satisfies the following properties:

$$\cos(\|\boldsymbol{k}(\boldsymbol{b}, \boldsymbol{\xi})\|) = \langle \boldsymbol{b}, \boldsymbol{\xi} \rangle \quad \text{and} \quad \|\boldsymbol{k}(\boldsymbol{b}, \boldsymbol{\xi})\| < \pi \tag{4.7}$$

Theorem 4.1.1. *Consider a unit vector $\boldsymbol{b}$ as the basis for the projection given in (4.4) and the corresponding reconstruction function defined in (4.6). The resulting transformation of unit vector $\boldsymbol{k}(\boldsymbol{v}, \boldsymbol{b}) : \{\boldsymbol{v} \in \mathbb{R}^N \setminus -\boldsymbol{b} \,:\, \|\boldsymbol{v}\| = 1\} \rightarrow \mathscr{K}$ defined in (4.2) is a bijection and the basis vector projects to the origin, i.e., $\boldsymbol{b} \rightarrow \boldsymbol{0}$.*



(a) Three initial vectors. (b) Direction-space

Figure 4.1: Various directions (here three) are described with respect to a basis direction $\boldsymbol{r}^0$ (a). The directions are transformed to the direction-space $\mathscr{K}$ (b) where the weighted mean, $\bar{\kappa}$, is obtained.

Proof. The proof is divided into three parts: *(I)* showing that transformation and reconstruction are the inverse functions of each other, *(II)* any unit vector is transformed to the direction space $\mathscr{K}$, and *(III)* is reconstructed to a unit vector.

(I) Inverse Functions: To be the inverse functions, applying one after the other onto a vector, must results in the original vector. (Vectors $\boldsymbol{b}$ will be treated separately below).
Let us apply the forward transformation based on (4.3) and (4.4) to a unit vector $\boldsymbol{v}_1$. It can be summarized to:

$$\kappa(\boldsymbol{b}) = \arccos\left(\langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle\right) \frac{\hat{\boldsymbol{B}}^T \boldsymbol{v}_1}{\|\hat{\boldsymbol{B}} \boldsymbol{v}_1\|} \tag{4.8}$$

with $\hat{\boldsymbol{B}}$ the matrix $\boldsymbol{B}$ without the first row.
We apply it to a single vector, hence the corresponding weight is $w_1 = 1$. From (4.5), we get $\bar{\kappa} = \kappa_1$. The reconstruction follows with (4.6):

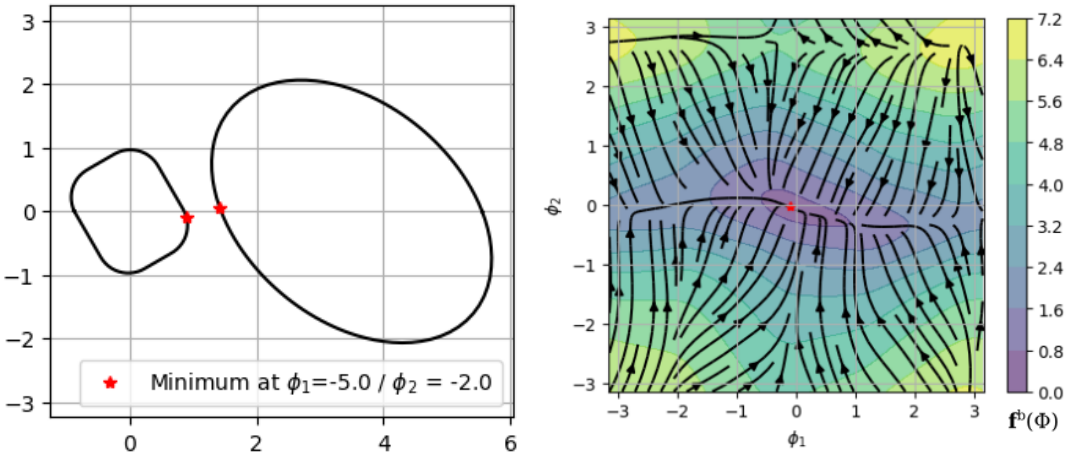
$$\bar{\boldsymbol{v}} = \boldsymbol{B} \begin{bmatrix} \cos\left(\|\bar{\kappa}\|\right) \\ \sin\left(\|\bar{\kappa}\|\right) \frac{\bar{\kappa}}{\|\bar{\kappa}\|} \end{bmatrix} = \boldsymbol{B} \begin{bmatrix} \cos\left(\arccos\left(\langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle\right)\right) \\ \frac{\sin(\arccos(\langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle))(\hat{\boldsymbol{B}})^T \boldsymbol{v}_1}{\|(\hat{\boldsymbol{B}})^T \boldsymbol{v}_1\|} \end{bmatrix} = \boldsymbol{B} \begin{bmatrix} \langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle \\ (\hat{\boldsymbol{B}})^T \boldsymbol{v}_1 \end{bmatrix} = \boldsymbol{B} (\boldsymbol{B})^T \boldsymbol{v}_1 = \boldsymbol{v}_1 \tag{4.9}$$

by using

$$\sin(\arccos\left(\langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle\right)) = \sqrt{1 - \langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle^2} = \sqrt{\|\boldsymbol{B}\boldsymbol{v}_1\|^2 - \langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle^2} = \|\left(\hat{\boldsymbol{B}}\right)^T \boldsymbol{v}_1\| \tag{4.10}$$

For the case that $\boldsymbol{v}_1 = \boldsymbol{b}$, we get from (4.4) that $\bar{\kappa} = \kappa_1 = \boldsymbol{0}$. And with (4.6, we get $\bar{\boldsymbol{v}}(\bar{\kappa}) = \boldsymbol{b}$, i.e., the original vector.
Hence for all cases, the transformation is bijective.

(II) Transformation Domain From definition of the transform in (4.4), the maximum magnitude is the arccos, which is $\|\kappa_i\| < \pi$. Hence it lies in the domain of (4.2). (The magnitude π is only reached for a vector $-\boldsymbol{b}$, which is excluded from the transform).



(a) Minimum Distance

(b) Value Function

Figure 4.2: The minimum distance problem for a rectangular object (with margin) and an ellipsoid. The boundary-reference-point which corresponds to the closes point is marked in red (a) and the corresponding gradient descent problem in direction space (b).

(III) Reconstruction Domain From the inverse transform (4.6), we have that norm of any transformed vector $\|\bar{\boldsymbol{v}}(\bar{\kappa})\| = 1$, this follows from the fact that $\boldsymbol{B}$ is orthonormal. As a result they all lie in the domain (4.1). □

4.1.1 Pairwise Closest Distance in Direction Space

The directional space can be used for gradient descent to find the closest distance between two (convex) obstacles. This is done by moving along the surface of the obstacle in direction space. Since the direction space is of dimension $d-1$, finding the closest point of each obstacle has only $2(d-1)$ degrees of freedom (instead of $2d$ in the Cartesian space). The problem converges to the global minimum when the points start on the line which connects the two center points.
The optimization problem is given as:

$$\min_{\Phi} \boldsymbol{f}^b(\Phi) \qquad \text{with } \boldsymbol{f}^b(\Phi) = \|\boldsymbol{\xi}_1^b(\phi_1) - \boldsymbol{\xi}_2^b(\phi_2)\|\,,\ \Phi = \begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix} \tag{4.11}$$

where the $\xi_i^b(\phi_i)$ denotes the boundary point in the direction ϕ_i for the obstacle i with respect to its reference point ξ_i^r. The direction space of each obstacle is created such that the null-direction points towards the other obstacles center. An example is visualized in Fig. 4.2.

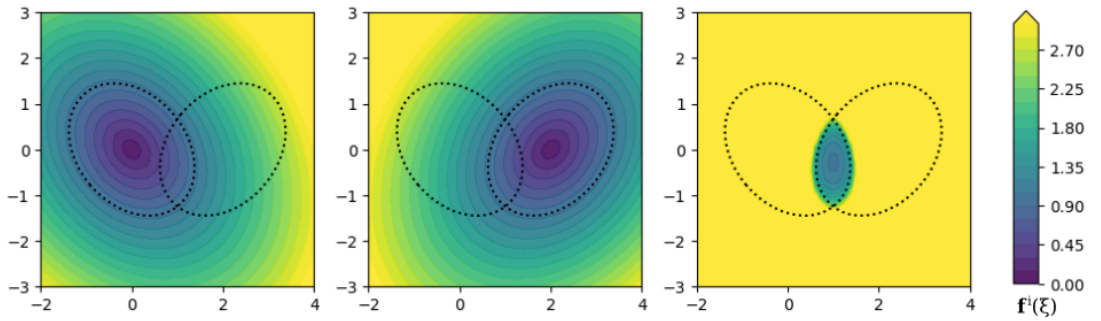


Figure 4.3: The individual Γ-function from the obstacle in (a) and (b) are summed as described in (4.12). The resulting value function (c) is used for the angle based gradient-descent.

4.1.2 Intersecting Obstacle Descent

Two intersecting obstacles need to share one reference point, which lies within both obstacles. The simplification of the problem to surface points only is not of use anymore. Hence, the optimization problem is evaluated in Cartesian space to find a common point that lies inside of the two obstacles' boundaries:

$$\min_{\xi \in \mathcal{X}_1^b \cap \mathcal{X}_2^b} \boldsymbol{f}^i(\boldsymbol{\xi}) \qquad \text{with } \boldsymbol{f}^i(\boldsymbol{\xi}) = \frac{\Gamma^b}{\Gamma^b - \Gamma_1(\xi)} + \frac{\Gamma^b}{\Gamma^b - \Gamma_2(\xi)} \tag{4.12}$$

with Γ^b the base distance value, i.e. where the value function reaches infinity. It is chosen slightly larger than the Γ-value on the surface, we simply choose $\Gamma^b = 1.1$. The step size can be optimized based on the gradient. The optimization problem is convex, and points starting within the intersection region will stay inside due to the infinite repulsion at the boundary as $\Gamma \to 1$. The value function of two ellipses can be found in Fig. 4.3.

4.1.3 Closest Distance for Mixed Environments

The above method for gradient descent in directional space to find the closest distance between two objects, can be applied to an object-boundary pair, if obstacle's curvature c^o is larger than boundary's curvature c^b at any position:

$$c^o(\xi_1) < c^b(\xi_2) \qquad \forall\, \xi_1, \xi_2 \tag{4.13}$$

with the local curvature being defined as:

$$c^{(\cdot)}(\boldsymbol{\xi}) = \lim_{\Delta\boldsymbol{\xi} \to \mathbf{0}} \frac{R(\boldsymbol{\xi}) - R(\boldsymbol{\xi} + \Delta\boldsymbol{\xi})}{\Delta\boldsymbol{\xi}} \qquad \forall\, \boldsymbol{\xi} \in \mathcal{X}^b : \langle \Delta\boldsymbol{\xi}, \boldsymbol{n}(\boldsymbol{\xi}) \rangle = 0 \tag{4.14}$$

Note that for non-circular obstacles, the condition might locally not hold mainly if the space

contains polygon obstacles with local flat regions ($c = 0$). This can lead to a locally non-optimal solution when placing the reference point based on the closest distance.

4.2 Vector Rotation

Publication Note

The material in this section was originally published together with Chapter 7 in:
L. Huber, J.-J. Slotine, and A. Billard (2023). "Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics". In: *IEEE Transactions on Robotics*

In this section, we introduce a novel approach to describe vector fields by extracting the local rotations between vectors. Additionally, we demonstrate how complex sequences of vector rotations can be effectively represented through a tree structure. Finally, we propose an algorithm for constructing the rotation tree and a method for reducing it into a single rotation. The method has the advantage that it ensures the absence of local minima by design.

This handling of vector rotations has proven essential for the Rotational Obstacle Avoidance Method (ROAM), as presented in Chapter 7. Moreover, the concept could be further applied to motion planning methods when the absence of local minima is required, but also more general vector addition tasks.

4.2.1 Perpendicular Rotation Base

Let us consider two initial directions $\boldsymbol{v}_i, \boldsymbol{v}_o \in \mathbb{R}^N \setminus \mathbf{0}$. They are used to construct the base vectors $\boldsymbol{b}_i, \boldsymbol{b}_o \in \{\boldsymbol{b} \in \mathbb{R}^N : \|\boldsymbol{b}\| = 1\}$:

$$\boldsymbol{b}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|}\,,\ \boldsymbol{b}_o = \frac{\hat{\boldsymbol{b}}_o}{\|\hat{\boldsymbol{b}}_o\|}\,, \quad \beta = \arccos\left(\frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle}{\|\boldsymbol{v}_i\|\,\|\boldsymbol{v}_o\|}\right) \tag{4.15}$$
$$\text{with} \quad \hat{\boldsymbol{b}}_o = \boldsymbol{v}_o - \boldsymbol{v}_i \langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle \ , \quad \forall\, \boldsymbol{v}_i, \boldsymbol{v}_o : \frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle}{\|\boldsymbol{v}_i\|\,\|\boldsymbol{v}_o\|} \neq -1$$

The angle β and two vectors $\boldsymbol{b}_i$, $\boldsymbol{b}_o$ represent the rotation of a vector in N dimensions, this is equivalent to $2N+1$ parameters.[1]

[1] This compares to $N(N-1)/2$ parameters required to describe rigid body rotation in N dimensions. Hence, already at $N = 4$ the rigid body orientation uses more parameters to describe vector rotation, and stays more efficient for higher dimensions.

4.2.2 Rotating a Vector

Rotating a vector $\boldsymbol{v}$ entails rotating the component which lies in the plane spanned by $\boldsymbol{b}_i$ and $\boldsymbol{b}_o$, while conserving the part orthogonal to the plane.

$$\begin{aligned} \boldsymbol{v}_r\left(\boldsymbol{v}, \beta, \boldsymbol{b}_{\{i,o\}}\right) &= p_0 \boldsymbol{b}_i \cos(\phi) + \boldsymbol{b}_o \sin(\phi) + \boldsymbol{v} - \sum_{j \in \{i,o\}} \boldsymbol{b}_j \left\langle \boldsymbol{b}_j, \boldsymbol{v} \right\rangle \\ \text{with} \quad \phi = \phi_0 + \beta\,,\ \tan(\phi_0) &= \frac{\langle \boldsymbol{b}_i, \boldsymbol{v} \rangle}{\langle \boldsymbol{b}_o, \boldsymbol{v} \rangle}\,, \\ p_0 &= \sqrt{\langle \boldsymbol{b}_i, \boldsymbol{v} \rangle^2 + \langle \boldsymbol{b}_o, \boldsymbol{v} \rangle^2} \end{aligned} \tag{4.16}$$

Note that by changing the rotation angle β, a partial rotation or over-rotation can be applied to a vector, too. In this work, the basis is only explicitly stated if not clear from the context.

Moreover, as each rotation basis is defined by the two vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_o$, a rotation can be rotated, too. For example, a rotation expressed in a relative reference frame can be rotated to the global frame. This allows for the evaluation of weighted rotation sequences, as will be discussed later in this section.

4.2.3 Weighted Rotation Summing

Let us assume that we have $N^{\text{vec}} \in \mathbb{N}_{>0}$ vector rotations which are defined with respect to a shared basis vector $\boldsymbol{b}_i$. The rotations have a respective second basis vector $\boldsymbol{b}_{o,v}$ and rotation angle β_v with $v \in 1..N^{\text{vec}}$. The basis and angle of the weighted average rotation is defined as:

$$\hat{\boldsymbol{b}}_o = \boldsymbol{B}(\boldsymbol{b}_i)\left(\sum_{v=1}^{N^{\text{vec}}} \boldsymbol{B}^T(\boldsymbol{b}_i)\, w_v \beta_v \boldsymbol{b}_{o,v}\right) \quad \text{and} \quad \hat{\beta} = \|\hat{\boldsymbol{b}}_o\| \tag{4.17}$$

where $w_v \in [0,1]$ are the vector weights, and $\boldsymbol{B}(\boldsymbol{b}_i)$ is an orthonormal basis with respect to the vector $\boldsymbol{b}_i$.

We further define the rotational sum using the symbol $\hat{+}$ as:

$$\hat{\boldsymbol{v}} = \boldsymbol{v}_0 \hat{+} \sum_{v=1}^{N^{\text{vec}}} w_v \boldsymbol{v}_v \tag{4.18}$$

where the summation is performed according to (4.17), with normalization of the vector pairs $[\boldsymbol{v}_0, \boldsymbol{v}_v]$ to obtain the rotations as described in (4.15).

4.2.4 Weighted Rotation Sequence

Let us assume that we have N^{rot} sequential rotations such that the output vector $\boldsymbol{b}_{o,n}$ of the element n is equal to the input vector of the next in the sequence $n+1$, i.e., $\boldsymbol{b}_{o,n} = \boldsymbol{b}_{i,n+1}$ with $n = 1..N^{\text{rot}} - 1$. To evaluate the weighted sequence each basis is adapted with respect to all the

parent rotations as:

$$\boldsymbol{b}_{\{i,o\},c} \leftarrow \boldsymbol{v}_r((w_n - 1)\beta_n, \boldsymbol{b}_{\{i,o\},c}) \quad n = 1..N^{\text{rot}} - 1,\ c > n \tag{4.19}$$

The full weighted rotation of the sequence is obtained using the vector rotation as defined in (4.15) with input vector $\boldsymbol{b}_{i,1}$ and output $\boldsymbol{v}_{o,n} = \boldsymbol{v}_r((1 - w_n)\beta_n, \boldsymbol{b}_{o,n})$ with $n = N^{\text{rot}}$.

4.2.5 Direction Tree Evaluation

Let us define a direction tree with nodes $n = 1..N^{\text{node}}$. There exists a single (unique) root node r, which has no parent. All other nodes n have single parent $p(n)$, but a node can have multile children, given by the set $\mathscr{C}_n$. The set of all offspring, i.e., including the children of children etc., is referred to as $\mathscr{C}_n^{\text{tot}}$. Each node has a level l, which indicates the discrete distance to the root. The value of each node corresponds to a unit vector $\boldsymbol{v}_i$, and hence the vector rotation between the parent and the node can be evaluated using (4.15). For given weights w_n, the weighted direction is evaluated as described in Algorithm 1. The rotation tree is interpreted as a combination of weighted summations and sequences, as described earlier in this section.

Algorithm 1 Rotation Tree Summing

Input: w_n, $\boldsymbol{v}_n$ for $n = 1..N^{\text{node}}$
Output: Averaged vector-rotation $\boldsymbol{v}^*$

for $l = L^{\max}..1$ **do** {Reverse iteration over levels}
  for $n : l_n = l$ **do** {For all nodes of the same level}
    $w_{p(n)} \leftarrow w_{p(n)} + w_n$ {Update cumulative weight}
  end for
end for
$\boldsymbol{v}_1^* = \boldsymbol{v}_r$ {Initialize base of root r}
for $l = 2..L^{\max}$ **do** {Iteration over levels}
  $\boldsymbol{v}_l^* = \boldsymbol{v}_{l-1}^* \hat{+} \sum_c^{l_c==l} \boldsymbol{v}_c$ {Weighted average from (4.18)}
  $\boldsymbol{v}_r(\cdot), \beta \leftarrow \boldsymbol{v}_{l-1}^*, \boldsymbol{v}_l^*$ {Rotation from vectors (4.15)}
  for $c \in \mathscr{C}_l^{\text{tot}}$ **do** {For all offsprings of l}
    $\boldsymbol{v}_{r,c}(\cdot), \beta_c \leftarrow \boldsymbol{v}_l^*, \boldsymbol{v}_c$ {Rotation from vectors (4.15)}
    $\boldsymbol{b}_{j,c} \leftarrow \boldsymbol{v}_{r,c}(-\beta_c, \boldsymbol{b}_{j,c})$ {Sequence inversion (4.19)}
    $\boldsymbol{b}_{j,c} \leftarrow \boldsymbol{v}_r(\beta, \boldsymbol{b}_{j,c})$ {Apply rotation to $\boldsymbol{v}_l^*$ as (4.16)}
  end for
end for
$\boldsymbol{v}^* = \boldsymbol{v}_{L^{\max}}^*$

Lemma 4.2.1. *The vector rotation as described in Algorithm 1 ensures a smooth vector summing for weights $w_n \in [0,1]$ such that $\sum_{n=1}^{N^{\text{node}}} w_n = 1$, such that we reach converge to a vector when its corresponding weight is approaching one, i.e.,* $\lim_{w_i \to 1} \boldsymbol{v}^* = \boldsymbol{v}_i$.

Proof. When a specific direction-node n has weight $w_n = 1$, it follows from Algorithm 1 all the

child-nodes weight one and zero otherwise:

$$w_c = \begin{cases} 1 & \text{if } c \in \mathscr{C}_n \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

Hence, we have a rotation sequence of the form (4.19). Due to the sum of the weights of all sequence elements being equal to one, the summed weight equals $\boldsymbol{b}_{o,e}$ where e is the index of the ending element. Note successive vectors cannot be anti-parallel, as denoted in (4.15). □

Note, that the weighted evaluation of the direction tree can be reduced to a single vector $\boldsymbol{v}^*$, or the whole sequence can be kept by storing all $\boldsymbol{v}^*_{(\cdot)}$. If all node-parent pair had an angle $\beta < \pi$, then this will be the case for the resulting sequence, too.

4.3 Obstacle Avoidance with Robots Arms

Publication Note

The material in this section was originally published together with Chapter 5 in:
L. Huber, J.-J. Slotine, and A. Billard (2022a). "Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds". In: *IEEE Transactions on Robotics*

Many dynamical system algorithms are developed for point masses, as discussed in Chapter 3. While it is straightforward to extend this to control robots, which can be approximated by a circle (e.g., drones, wheel-based platforms) by creating a margin around all obstacles wide enough to account for the robot's shape. The method can also be extended to higher dimensions and multiple degrees of freedom robot arms by describing and evaluating the system (robot and obstacle) in joint space. This, however, requires representing the obstacle in configuration space, which is not always easy, especially when the obstacle moves.

Alternatively, in the rest of this section, we introduce a weighted evaluation of the desired dynamics along a robot arm's links to obtain a collision-free trajectory toward the desired goal in Cartesian space.

4.3.1 Goal Command Towards Attractor

Consider a robot arm with end-effector's position $\boldsymbol{\xi}$. The desired velocity towards the attractor is proposed in Section 3.4 and denoted as $\dot{\boldsymbol{\xi}}$. The goal command in joint space is evaluated through inverse kinematics as:

$$\dot{\boldsymbol{q}}^g = \hat{\boldsymbol{J}}^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{\xi}} \tag{4.21}$$

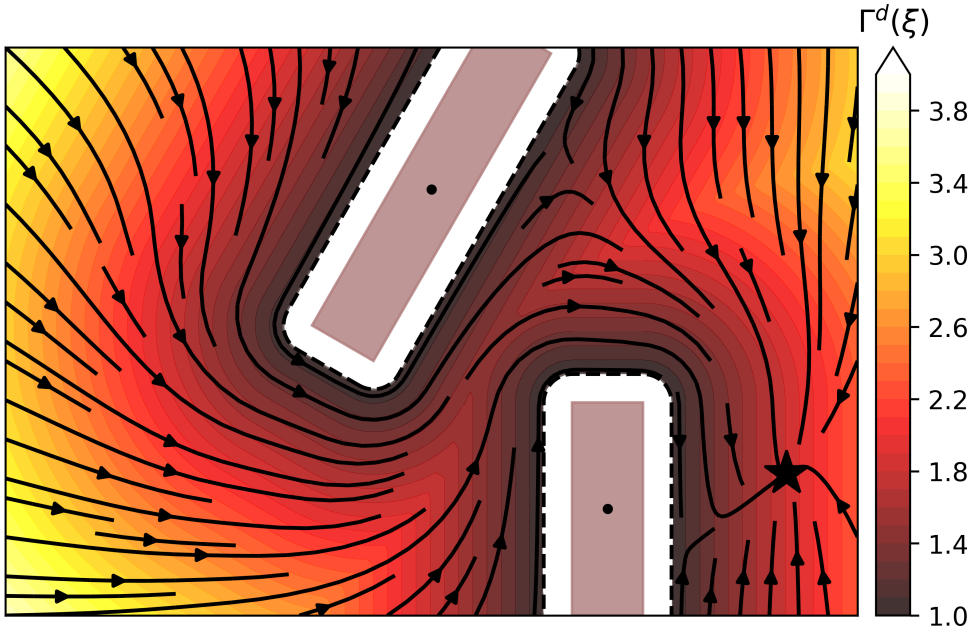


Figure 4.4: Γ-field and desired direction in a multi-obstacle environment as given in (4.22).

where $\hat{\boldsymbol{J}}(\boldsymbol{q})$ is the Jacobian of the robot arm with respect to position only. The pseudo-inverse of the Jacobian is obtained as described in (3.27).

Γ-Danger Field

The closer an obstacle is to the robot, the more it is in danger of colliding. Based on the $\Gamma(\boldsymbol{\xi})$-function introduced in Section 3.3.2, we introduce a Γ-danger field with respect to all N^{obs} obstacles as

$$\Gamma^d(\boldsymbol{\xi}) = \min_{o \in [1..N^{\text{obs}}]} \Gamma_o(\boldsymbol{\xi}) \tag{4.22}$$

The field is displayed around two obstacles in Fig. 4.4. This field evaluates the weights to continuously switch between the goal command $\dot{\boldsymbol{q}}^g$ to the avoidance command $\dot{\boldsymbol{q}}^m$.

4.3.2 Link Avoidance

The avoidance command $\dot{\boldsymbol{q}}^m$ ensures that each link avoids collision with the environment. To extend the obstacle avoidance algorithm to a rigid body, we introduce N^S section points $\boldsymbol{\xi}^S_{l,s}$ for each link l and $s \in [1..N^s]$. The dynamical system is evaluated at each section point and coupled with a section weight w^S_s. The section point weight w^S_s increases the lower the Γ-danger value.

4.3.3 Evaluation Weights for a Robot Arm

We introduce link weights w_l^L for each link $l \in [1..N^L]$, and section weights $w_{l,s}^S$ for each section point $\xi_{l,s}^S$ with $s \in [1,..N^S]$. The weights determine how much the obstacle avoidance command influences the control, i.e., zero weights only indicate following the goal velocity. In contrast, a weight of 1 indicates full avoidance at this specific position.
These weights are designed, such that their product is smaller than one, i.e.:

$$0 \leq \sum_l \left(w_l^L \sum_s w_{l,s}^S \right) \leq 1 \tag{4.23}$$

Furthermore, if a section point approaches the surface of an obstacle, it should dominate, i.e.:

$$\Gamma^d(\xi_{l,s}^S) \to 1 \quad \Rightarrow \quad w_l^L w_{l,s}^S \to 1 \tag{4.24}$$

And when the robot arm is far away from any surface, all weights should go to zero:

$$\min_{l,s} \left(\Gamma(\xi_{l,s}^S) \right) \to \infty \quad \Rightarrow \quad \sum_l \left(w_l^L \sum_s w_{l,s}^S \right) \to 0 \tag{4.25}$$

Base Weight

For each link l and section point s, we define a Γ-danger weight $w^\Gamma(\Gamma^d) :]1,\infty[\to]\infty,0[$ to represent the danger of colliding for a specific point: the higher the weight, the greater the chance of collision. It is defined as:

$$w^\Gamma(\Gamma^d) = \begin{cases} \frac{\Gamma^c - \Gamma^{\min}}{\Gamma^d(\xi_{s,l}^S) - \Gamma^{\min}} - 1 & \text{if} \quad \Gamma^d(\xi_{s,l}^S) < \Gamma^c \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

where $\Gamma^{\min} = 1$ is the lower bound, $\Gamma^c > 1$ the cutoff value and $\Gamma^d(\xi)$ defined in 4.22.

Link Weights

The influence of each link l is evaluated as:

$$\hat{w}_l^L = c^L |\dot{\boldsymbol{q}}_{[l]}^g| \frac{l}{N^L} \max_{s \in [1..N^S]} w^\Gamma(\xi_{s,l}^S) \quad \forall\, l \in [1..N^L] \tag{4.27}$$

where $c^L \in \mathbb{R}_{>0}$ is a constant link-weight factor and $\dot{\boldsymbol{q}}_{[l]}^g$ the l-th element of the goal vector defined in (4.21). The l-factor in the equation gives increasing importance for links closer to the end-effector.

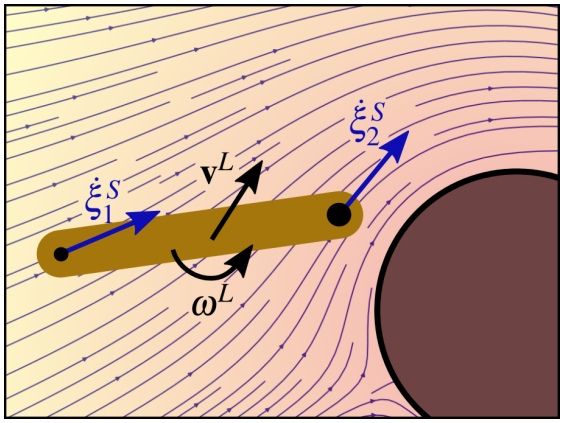


Figure 4.5: The resulting linear velocity $\boldsymbol{v}^L$ and angular velocity ω^L for a rigid body with to sections points.

The link weights are obtained through normalization:

$$w_l^L = \begin{cases} \hat{w}_l^L / \hat{w}^{\text{sum}} & \text{if } \hat{w}^{\text{sum}} > 1 \\ \hat{w}_l^L & \text{otherwise} \end{cases} \quad \text{with } \hat{w}^{\text{sum}} = \sum_{l=1}^{N^L} \hat{w}_l^L \tag{4.28}$$

Section Weights

For all links l with $w_l^L > 0$, the influence weight of each section point is evaluated as:

$$w_s^S = \frac{\hat{w}_s^S}{\sum_{s=1}^{N^S} \hat{w}_s^S} \quad \text{with} \quad \hat{w}_s^S = \frac{s}{N_S} w^\Gamma(\xi_{l,s}^S) \quad \forall\, s \in [1..N_S] \tag{4.29}$$

Rigid Body Dynamics

The linear and angular avoidance velocity for link l are obtained as:

$$\boldsymbol{v}^L = \sum_{s=1}^{N^S} w_s^S \dot{\boldsymbol{\xi}}_{l,s}^S \text{ and } \omega^L = \sum_{s=1}^{N^S} w_s^S (\boldsymbol{\xi}_{l,0}^S - \boldsymbol{\xi}_{l,s}^S) \times \left(\boldsymbol{v}_s^S - \boldsymbol{v}^L\right) \tag{4.30}$$

where $\boldsymbol{\xi}_{l,0}^S$ is the position of the root of a link l, and $\dot{\boldsymbol{\xi}}_{l,s}^S$ is the dynamical system based avoidance evaluated as described in Chapter 3. A single link avoiding a circle can be seen in Fig. 4.5.

Joint Modulation

The desired joint avoidance command is obtained through inverse kinematics as:

$$\dot{\boldsymbol{q}}^m = \boldsymbol{J}^\dagger(\boldsymbol{q}, \boldsymbol{\xi}^S_{l,0}) \begin{bmatrix} \boldsymbol{v}^L \\ \omega^L \end{bmatrix} \tag{4.31}$$

where $\boldsymbol{J}(\boldsymbol{q}, \boldsymbol{\xi}^S_{l,0})$ is the Jacobian up the root of the link l, i.e., the positionof the section point $\boldsymbol{\xi}^S_{l,0}$.

4.3.4 Evaluation Along the Robot Arm

The evaluation is performed by iterating over all N^L links, starting from the base of the robot arm. At each iteration, the joint control $\dot{\boldsymbol{q}}^c$ of all links that are part of the kinematic chain are updated, i.e., if the evaluation is performed for link l, the joint control is updated for $\dot{\boldsymbol{q}}^c_i \; \forall i \leq l$. The evaluation is weighted along with the link to ensure collision avoidance of all links while trying to follow the goal command $\dot{\boldsymbol{q}}^g$. The link weights w^L_l are further described in Sec. 4.3.3.

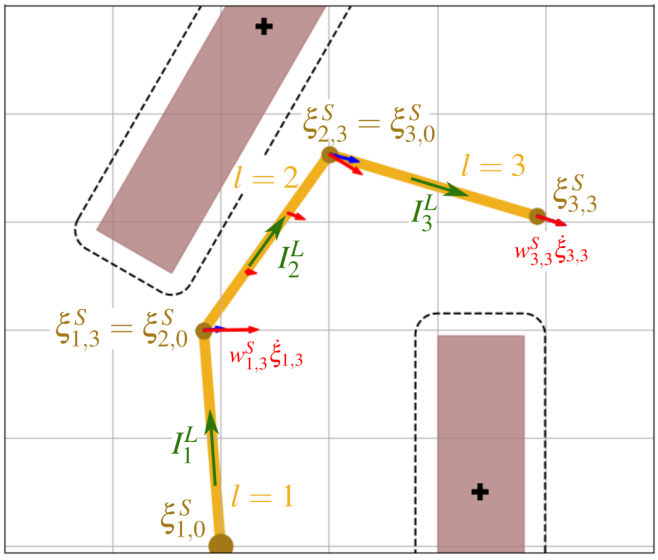


Figure 4.6: A robot arm with three links ($N^L = 3$), which each containing three sections points ($N^S = 3$) is surrounded by two obstacles. The blue arrows are the goal velocity obtained (only evaluated at the joints), and the red arrows are the weighted avoidance velocities. The direction of joint rotation $\boldsymbol{I}^\omega_l$ points out of the plane.

Initial Joint Control

The joint control is initialized based on the goal control from (4.21) as:

$$\dot{\boldsymbol{q}}^c \leftarrow \left(1 - \sum_{l=1}^{N^L} w^L_l\right) \dot{\boldsymbol{q}}^g \tag{4.32}$$

The first element of the joint command is then updated based on the avoidance velocity obtained in (4.31) as:

$$\dot{\boldsymbol{q}}^c_{[1]} \leftarrow \dot{\boldsymbol{q}}^c_{[1]} + w^L_1 \dot{\boldsymbol{q}}^m_{[1]} \tag{4.33}$$

Joint Control Correction

To ensure obstacle avoidance of each link, the obtained control command $\dot{\boldsymbol{q}}^c$ differs from the ideal goal command $\dot{\boldsymbol{q}}^g$. This difference is obtained at link l as:

$$\boldsymbol{v}^\Delta = \hat{\boldsymbol{J}}_l(\boldsymbol{q}) \left(\dot{\boldsymbol{q}}^g_{[1:l]} - \dot{\boldsymbol{q}}^c_{[1:l]} \right) \quad \forall\, l > 1 \tag{4.34}$$

where $\dot{\boldsymbol{q}}^c_{[:l]}$ is the current control command, evaluated up to link $l-1$, and $\hat{\boldsymbol{J}}_l(\boldsymbol{q})$ is the arm Jacobian with respect to position up to link l.
The joint speed of link l, which would best account for this difference, can be evaluated as:

$$\dot{q}^\Delta = \langle \boldsymbol{\omega}^\Delta, \boldsymbol{I}^\omega \rangle \quad \text{with } \boldsymbol{\omega}^\Delta = \boldsymbol{v}^\Delta \times \boldsymbol{I}^L \tag{4.35}$$

where $\boldsymbol{I}^\omega$ is the direction of rotation of the joint actuating link l and $\boldsymbol{I}^L$ is the direction along the link, i.e., pointing from one joint to the next.[2] The variables can be observed in Fig. 4.6.

Joint Control Update

The velocity of each joint is evaluated one by one, starting at the joint closest to the robot's base towards the end-effector (see Algorithm 2). The joint command $\dot{\boldsymbol{q}}^c$ is first updated by applying the correction control from (4.35) to the current joint l as:

$$\dot{\boldsymbol{q}}^c_{[l]} \leftarrow \dot{\boldsymbol{q}}^c_{[l]} + \dot{q}^\Delta \sum_{i=1}^{l-1} w^L_i \qquad \forall\, l > 1 \tag{4.36}$$

The modulation command $\dot{\boldsymbol{q}}^m$ from (4.31) is then applied to all underlying joints:

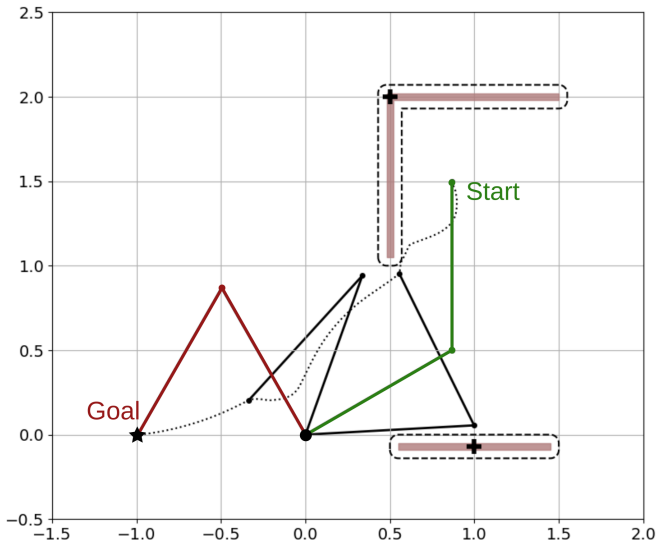
$$\dot{\boldsymbol{q}}^c_{[1:l]} \leftarrow \dot{\boldsymbol{q}}^c_{[1:l]} + w^L_l \dot{\boldsymbol{q}}^m_{[1:l]} \qquad \forall\, l > 1 \tag{4.37}$$

This is executed iteratively for all joints $l > 1$.

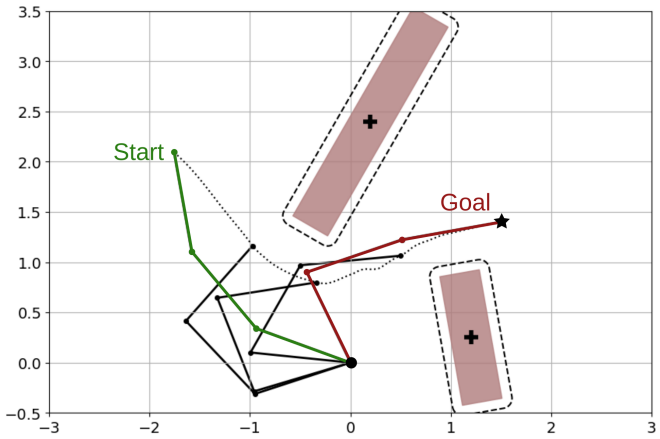
4.3.5 Validation in Simulation

We applied the algorithm to two scenarios with planar robotic arms (Fig. 4.7). The scenario in Fig. 4.7a was chosen similar to the one presented in Zanchettin, Lacevic, and Rocco, 2015. Our approach can avoid obstacles in a similar setup without a navigation function.
The scenario in Fig. 4.7b includes a more complex robot with three links. Additionally, the skew

[2] We assume single-degree of freedom joints.



(a) 2 DoF arm



(b) 3 DoF arm

Figure 4.7: Time sequence of robot arms navigating in two different environments.

placement of the obstacle and the position of the start end goal point require the robot to actively go around the obstacle guided by the reference point (see Fig. 4.4 for the full DS).

Algorithm 2 Joint Control Command for a Robot Arm

Input: N^L, N^S, $f(\boldsymbol{\xi})$, obstacle-environment
Output: $\dot{\boldsymbol{q}}^c$

$\boldsymbol{\xi}_o^r \ \forall o \in 1..N^{\text{obs}}$ {update dynamic obstacles as in Sec. 3.4.1}
$\dot{\boldsymbol{q}}^g \leftarrow \left(\boldsymbol{J}(\boldsymbol{q})\right)^\dagger \dot{\boldsymbol{\xi}}$ {compute goal command as in (4.21)}
for $l = 1$ **to** N^L **do**
  for $s = 1$ **to** N^S **do**
    $\Gamma(\boldsymbol{\xi}_{s,l})$ {compute *danger*-field as in (4.22)}
    w^Γ {compute danger-weight as in (4.26)}
  end for
  w_l^L {compute link weight as in (4.28)}
end for
$\dot{\boldsymbol{q}}^c \leftarrow \left(1 - \sum_l w_l^L\right) \dot{\boldsymbol{q}}^g$ {initialize control command}
$\dot{\boldsymbol{q}}^c_{[1]} \leftarrow \dot{\boldsymbol{q}}^c_{[1]} + w_1^L \dot{\boldsymbol{q}}^m_{[1]}$
for $l = 2$ **to** N^L **do**
  if $w_l^L > 0$ **then**
    for $s = 1$ **to** N^s **do**
      w_s^S {compute section weight as in (4.29)}
    end for
    $\boldsymbol{v}_l^L, \omega_l^L$ {compute avoidance velocities as in (4.30)}
    $\dot{\boldsymbol{q}}_l^m$ {compute modulation command as in (4.31)}
    $\dot{\boldsymbol{q}}^c_{[1:l]} \leftarrow \dot{\boldsymbol{q}}^c_{[1:l]} + w_l^L \dot{\boldsymbol{q}}^m_{[1:l]}$
  end if
  $\boldsymbol{v}^\Delta \leftarrow \boldsymbol{J}_l(\boldsymbol{q}) \left(\dot{\boldsymbol{q}}^g_{[1:l]} - \dot{\boldsymbol{q}}^c_{[1:l]}\right)$
  $\boldsymbol{\omega}^\Delta \leftarrow \boldsymbol{v}^\Delta \times \boldsymbol{I}^L$
  $\dot{q}^\Delta \leftarrow \langle \boldsymbol{\omega}^\Delta, \boldsymbol{I}^\omega \rangle$
  $\dot{\boldsymbol{q}}^c_{[l]} \leftarrow \dot{\boldsymbol{q}}^c_{[l]} + \dot{q}^\Delta \sum_i w_i^L$
end for

5 Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds

Publication Note

The material in this chapter is adapted from:
Huber, L., J.-J. Slotine, and A. Billard (2022a). "Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds". In: *IEEE Transactions on Robotics.*
Source Code:

- Algorithm: https://github.com/hubernikus/dynamic_obstacle_avoidance.git
- Implementation on QOLO: https://github.com/epfl-lasa/qolo_modulation.git

Multimedia: Supplementary video can be found on:

https://youtu.be/qrz1cx8sGB4

This chapter presents a closed-form approach to constraining a flow within a given volume and around objects. The flow is guaranteed to converge and to stop at a single fixed point. The obstacle avoidance problem is inverted to enforce that the flow remains enclosed within a volume defined by a polygonal surface. We formally guarantee that such a flow will never contact the boundaries of the enclosing volume or obstacles. It asymptotically converges towards an attractor. We further create smooth motion fields around obstacles with edges (e.g., tables). Both obstacles

and enclosures may be time-varying, i.e., moving, expanding, and shrinking. The technique enables a robot to navigate within enclosed corridors while avoiding static and moving obstacles. It was applied on an autonomous robot (QOLO) in a static complex indoor environment and tested in simulations with dense crowds. The final proof of concept was performed in an outdoor environment in Lausanne. The QOLO-robot successfully traversed a marketplace in the center of town in the presence of a diverse crowd with a non-uniform motion pattern.

5.1 Introduction

Robots navigating in human-inhabited environments will encounter disturbances constantly, for instance, when autonomous delivery robots drive around pedestrians. The robot must have a flexible control scheme to avoid collisions. As the number of obstacles increases and their motion becomes less predictable, the robot needs to reevaluate its path within milliseconds to prevent a crash while moving actively towards its goal.
Control using dynamical systems (DS) is ideal for addressing such situations. In contrast to classical path planning, the control law is closed-form, hence requires no replanning, and can ensure impenetrability of obstacles (Feder and J.-J. Slotine, 1997; S. M. Khansari-Zadeh and Billard, 2011). DS offer stability and convergence guarantees in addition to the desired on-the-fly reactivity.

This chapter addresses the need for a reactive (closed-form) obstacle avoidance approach with formal guarantees of impenetrability to handle highly dynamic environments and realistic obstacles, such as obstacles with sharp edges. Thiis chapter extends obstacle avoidance technniques described in Chapter 3, in which we presented a closed-form obstacle avoidance approach guaranteed to not penetrate smooth concave, albeit star-shaped, obstacles. We present three novel theoretical contributions:

1. We extend the modulation parameters of the obstacle avoidance with surface friction and repulsive value (Section 5.2). They allow an agent to move slower and further away from obstacles, respectively. The extensions result in more *cautious* behavior.

2. We invert the obstacle description to ensure that the robot moves within the enclosed space defined by the boundary while preserving convergences guarantees towards an attractor. This boundary may represent walls, furniture, or even joint limits of manipulators (Section 5.3).

3. We extend the approach to handle *nonsmooth* surfaces, i.e., obstacles with sharp edges. The novelty comes from creating a smooth dynamical system around an obstacle without approximation of the curvature (Section 5.4).

4. We show that the approach can be extended to tackle dynamic environments, with obstacles that have *deforming* shapes (Section 5.5).

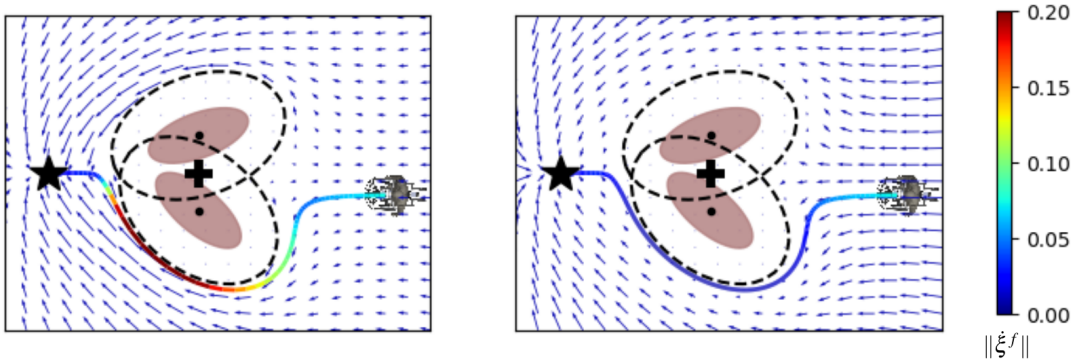


Figure 5.1: The velocity obtained with the isometric inspired eigenvalue (left) results in accelerations close to the obstacle. The modulation inspired by surface friction (right) reduces the velocity with decreasing distance to the obstacle. The black star is the attractor $\boldsymbol{\xi}^a$, and the black cross is the (shared) reference point $\boldsymbol{\xi}^r$.

We validate these contributions with a wheelchair robot moving in a simulated crowd of pedestrians and an office environment with real furniture (Section 5.7).

This work focuses on motion towards a goal of an autonomous dynamical system, i.e., $\lim_{t\to\infty} f(\boldsymbol{\xi}_a) = 0$. The most direct dynamics towards the attractor is a linear dynamical system of the form:

$$\boldsymbol{f}(\boldsymbol{\xi}) = -k(\boldsymbol{\xi} - \boldsymbol{\xi}^a) \tag{5.1}$$

where $k \in \mathbb{R}$ is a scaling parameter.

5.2 Obstacle Avoidance

5.2.1 Surface Friction Imitation

The choice of eigenvalues given in in (3.18) is inspired by the harmonic potential flow, i.e., the description of the potential flow of an incompressible fluid. The incompressibility constraint forces the velocity to increase in regions where the flow is pushed around the obstacle, i.e., the eigenvalues in the tangent direction increase. This leads to acceleration close to the surface (see Fig. 5.1).
We propose to mimic surface friction, i.e., slowing down in tangent direction close to an obstacle ($\lim_{\Gamma\to 1} \boldsymbol{\xi}^e = 0$). A friction parameter $\lambda^f(\boldsymbol{\xi})$ ensures the slowing down close to the surface. The friction dynamics $\dot{\boldsymbol{\xi}}^f$ are obtained by applying the factor to tangent and reference direction as follows:

$$\dot{\boldsymbol{\xi}}^f = \lambda^f(\boldsymbol{\xi}) \frac{\|\boldsymbol{f}(\boldsymbol{\xi})\|}{\|\dot{\boldsymbol{\xi}}\|} \dot{\boldsymbol{\xi}} \qquad \text{with} \quad \lambda^f(\boldsymbol{\xi}) = 1 - 1/\Gamma(\boldsymbol{\xi}) \tag{5.2}$$

where the $\dot{\boldsymbol{\xi}}$ is the modulated velocity from (3.13).

5.2.2 Repulsive Eigenvalue

The eigenvalues in reference direction given in (3.18) are always positive, as follows from $\Gamma(\boldsymbol{\xi}) \geq 1$. This results in eigenvalues in the range of $\lambda^r(\boldsymbol{\xi}) \in [0, 1]$, hence reduces the velocity in radial direction.
Conversely, active repulsion can be achieved through negative eigenvalues, i.e., $\lambda^r(\boldsymbol{\xi}) < 0$. Active repulsion increases the distance by which the an agents avoids the obstacle (Fig. 5.2). Since the repulsive eigenvalues are incorporated in the modulation matrix in (3.13), it is ensured that attractors are preserved.
The eigenvalue in radial direction is defined for repulsive obstacles as:

$$\lambda^r(\boldsymbol{\xi}) = \begin{cases} 1 - \left(c^{\text{rep}}/\Gamma(\boldsymbol{\xi})\right)^{1/\rho} & \text{if } \langle \boldsymbol{f}(\boldsymbol{\xi}), \boldsymbol{r} \rangle < 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.3}$$

with the repulsive coefficient $c^{\text{rep}} \geq 1$. A repulsive coefficient $c^{\text{rep}} = 1$ corresponds to no repulsion. Note, that the repulsive eigenvalues are coupled with no *tail-effect*, i.e. $\lambda^r(\boldsymbol{\xi}) = 0$ in the wake of an obstacle, see (S. M. Khansari-Zadeh and Billard, 2012).

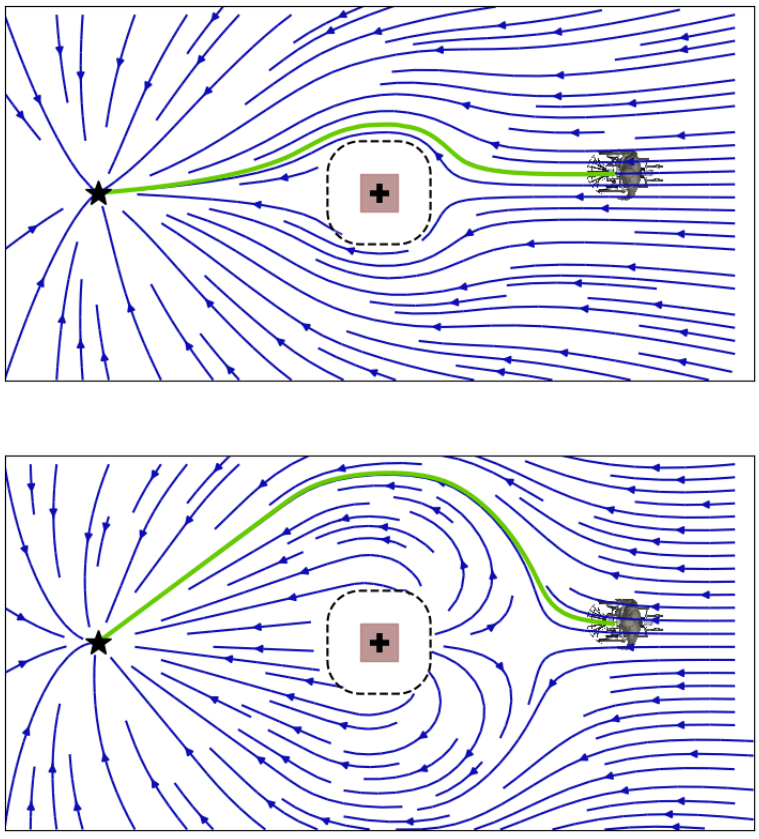


Figure 5.2: A repulsion coefficient of $c^{\text{rep}} = 1$ results in strictly positive eigenvalues (top). An increased distance to the obstacle is obtained with higher repulsion coefficients, such as $c^{\text{rep}} = 2$ (bottom).

5.3 Inverted Obstacle Avoidance

An autonomous robot often encounters scenarios where it has boundaries that it cannot pass. This might be a wall for a wheeled robot or the joint limits for a robot arm. These constraints can be interpreted as staying within an obstacle, where the obstacle represents the limits of the free space.

5.3.1 Distance Inversion

The distance function $\Gamma(\boldsymbol{\xi})$ from (3.9) can be evaluated within the obstacle $\boldsymbol{\xi} \in \mathscr{X}^o$.[1] For interior points, our boundary function is monotonically decreasing along the radial direction, i.e. the Lie derivative with respect to the reference direction is positive. Furthermore it is bounded. This can be written as:

$$L_{\boldsymbol{r}}\Gamma = \left\langle \frac{\partial \Gamma(\boldsymbol{\xi})}{\partial \boldsymbol{r}(\boldsymbol{\xi})}, \boldsymbol{r}(\boldsymbol{\xi}) \right\rangle > 0, \qquad \Gamma(\boldsymbol{\xi}) \in [0, 1[\quad \forall\, \mathscr{X}^o \setminus \boldsymbol{\xi}^r \tag{5.4}$$

We consider the obstacle boundary $\mathscr{X}^b$ as the description of an enclosing hull. It follows that the interior points of the classical obstacle become points of free space of the enclosing hull and vice versa. Boundary points stay boundary points. We define the distance function of wall-obstacles as the inverse of the obstacle distance function:

$$\Gamma^w(\boldsymbol{\xi}) = 1/\Gamma^o = \left(R(\boldsymbol{\xi})/\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\|\right)^{2p} \qquad \forall\, \mathbb{R}^d \setminus \boldsymbol{\xi}^r \tag{5.5}$$

This new distance function fulfills the condition for the three regions as given in (3.3.2). The distance function Γ^w is now monotonically decreasing along radial direction and reaches infinity at the reference point, i.e., $\lim_{\boldsymbol{\xi}\to\boldsymbol{\xi}^r} \Gamma^w(\boldsymbol{\xi}) \to \infty$ (see Fig. 5.3).

5.3.2 Modulation Matrix

The modulation matrix, defined in (3.13), consists of the eigenvalue $\boldsymbol{D}(\boldsymbol{\xi})$ and basis matrix $\boldsymbol{E}(\boldsymbol{\xi})$. For inverted obstacles, the diagonal eigenvalue matrix from (6.7) is a function of the inverted distance function $\Gamma^w(\boldsymbol{\xi})$.
Conversely, the basis matrix is constant along the radial direction. It can be evaluated everywhere (including the interior of a boundary) except at the reference point ($\boldsymbol{\xi} = \boldsymbol{\xi}^r$). Since the reference direction from (3.5) is a zero vector, no orthogonal basis is defined. However, the distance value $\Gamma^w(\boldsymbol{\xi}^r)$ reaches infinity, hence from (6.7) we get that the diagonal matrix is equal to the identity matrix:

$$\Gamma^w(\boldsymbol{\xi}^r) \to \infty \quad \Rightarrow \quad \boldsymbol{D}(\boldsymbol{\xi}^r) = \boldsymbol{I} \tag{5.6}$$

[1] In the classic obstacle avoidance case, this is of no use, since theoretically the obstacle does never reach the boundary (L. Huber, Billard, and J.-J. Slotine, 2019), and practically an *emergency* control has to be applied in this case.

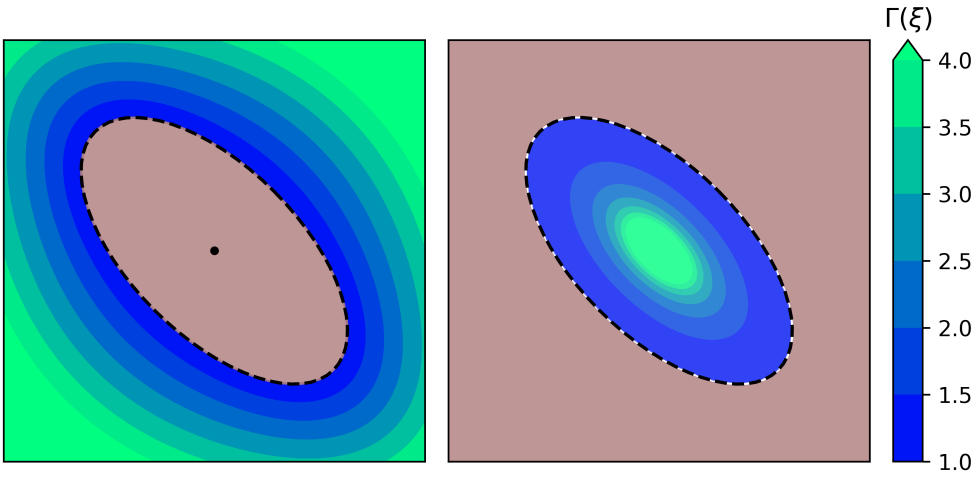


Figure 5.3: Forward and inverted Γ-function for the same shape. The brown region marks the inside of the obstacle and wall, respectively.

Using (3.13), it follows for the modulated dynamics at the reference point:

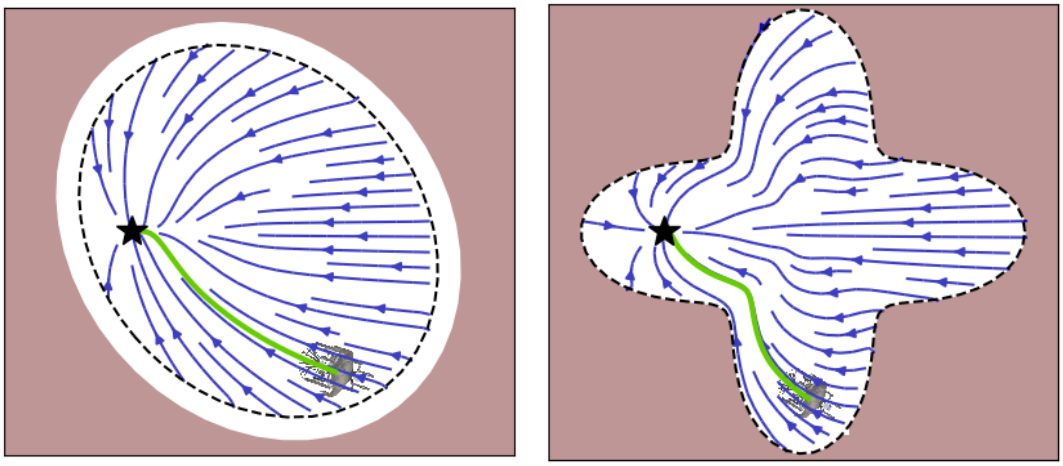
$$\boldsymbol{M}(\boldsymbol{\xi}^r) = \boldsymbol{E}(\boldsymbol{\xi}^r)\boldsymbol{D}(\boldsymbol{\xi}^r)\boldsymbol{E}(\boldsymbol{\xi}^r)^{-1} = \boldsymbol{E}(\boldsymbol{\xi}^r)\boldsymbol{I}\boldsymbol{E}(\boldsymbol{\xi}^r)^{-1} = \boldsymbol{I} \qquad \Rightarrow \quad \dot{\boldsymbol{\xi}}^r = \boldsymbol{f}(\boldsymbol{\xi}^r) \tag{5.7}$$

The influence of modulation approaches zero when reaching the reference point. Hence the dynamical modulation is continuously defined.

Theorem 5.3.1. *Consider a star-shaped enclosing wall in $\mathbb{R}^d$ with respect to a reference point inside the obstacle $\boldsymbol{\xi}^r$, and a boundary $\Gamma^w(\boldsymbol{\xi}) = 1$ as in (7.12). Any trajectory $\{\boldsymbol{\xi}\}_t$, that starts within the free space of an enclosing wall, i.e., $\Gamma(\{\boldsymbol{\xi}\}_0) > 1$ and evolves on a smooth path according to (3.13), will never reach the wall, i.e., $\Gamma(\{\boldsymbol{\xi}\}_t) > 1$, $t = 0..\infty$ and converges towards an attractor $\boldsymbol{\xi}^a \in \mathcal{X}^f$, i.e., $\lim_{t\to\infty} \boldsymbol{\xi} \to \boldsymbol{\xi}^a$.*

Proof. **Applicability of General Proofs**
In (L. Huber, Billard, and J.-J. Slotine, 2019) convergence has been proven for star-shaped obstacles. The proof was developed based on the distance function $\Gamma(\xi)$. Due to inverting the distance function for enclosing wall obstacles and a continuous definition of the modulation, the proof of star-shaped obstacles applies to the case of enclosing walls.

Continuity Across Reference Point
In Sec. 5.3.1 the Inverted distance function $\Gamma^w(\xi)$ was not defined at the reference point, as it reaches an infinite value. The continuous definition for the eigenvalue is a unit value, i.e. $\lambda^e(\xi^r) = \lambda^r(\xi^r) = 1$, it follows that the diagonal matrix is equal to the identity matrix $D(\xi^r) = I$.



(a) Ellipsoid boundary

(b) Star-shaped boundary

Figure 5.4: A smooth flow with full convergence towards the attractor (black star) can be observed within any star-shaped wall with reference point $\boldsymbol{\xi}^r$ (black plus).

As shown in (5.7), we get:

$$\xi = \xi^r \;\rightarrow\; \dot{\xi} = \boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^{-1}\boldsymbol{f}(\xi^r) = \boldsymbol{E}\boldsymbol{I}\boldsymbol{E}^{-1}\boldsymbol{f}(\xi^r) = \boldsymbol{f}(\xi^r) \tag{5.8}$$

i.e. no modulation of the initial DS. In fact, this is equivalent to the case far away for a classical obstacle with $\lim_{\|\xi-\xi^r\|\to\infty} \Gamma^o(\xi) \to \infty$.
Even though the basis matrix $\boldsymbol{E}(\xi)$ is not defined at ξ^r, the DS is continuously defined across this point since the modulation has no effect.
The trajectory that traverses the reference point ξ^r of the inverted obstacle corresponds to the trajectory that gets stuck in a saddle point for a common obstacle. As a result, there is full convergence for the inverted obstacles. ■

□

5.3.3 Guiding Reference Point to Pass Wall Gaps

In many practical scenarios, a hull entails gaps or holes through which the agent enters or exits the space (e.g., door in a room). The modulation-based avoidance slows the agent down when approaching the boundary and does not let it pass through such an exit. We introduce a *guiding reference point* $\boldsymbol{\xi}^g$ for boundary obstacles to counter this effect. We assume convex walls, and the robot size being smaller than the gap width. It is assumed that gap is a priori known from a map. In (5.7), it was shown that at the center of an inverted obstacle, the influence of the modulation vanishes.
Let us define the *gap region* $\mathcal{X}^g$ enclosed by the lines connecting the gap edges and the reference

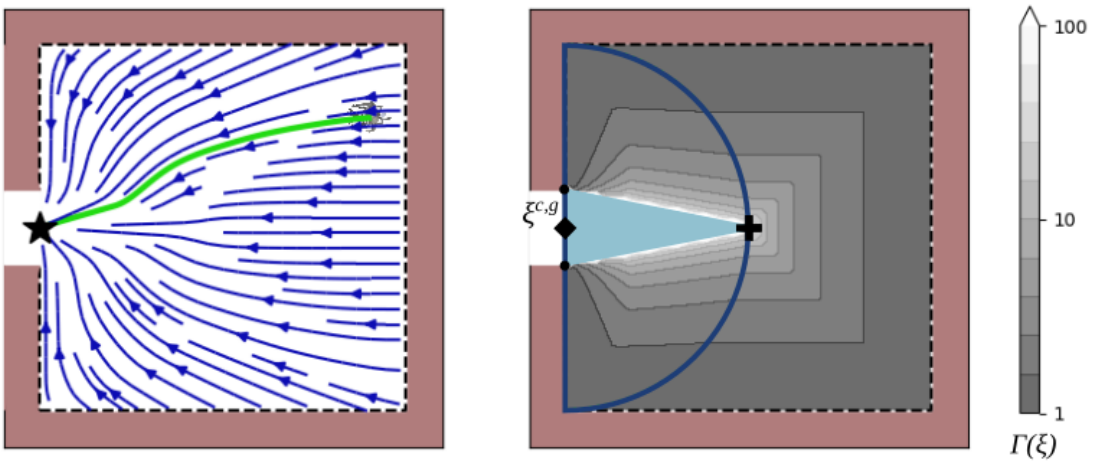


Figure 5.5: The dynamical system (left) is not modulated in front of the gap since the Γ-function reaches infinity (right). The *gap region* $\mathscr{X}^g$ is the blue region. The influence of the gap is limited by the blue half-circle around the center of the gap $\boldsymbol{\xi}^{c,g}$ (black square).

point $\boldsymbol{\xi}^r$ (see Fig. 5.5). The guiding reference point is designed in the following manner: close to the gap, the guiding reference point is equal to the position of the evaluation, hence no influence of the modulation. Far away from the gap, the guiding reference point is equal to the reference point; hence the wall has no influence. In between the two regions, the guiding reference point is projected onto the gap region $\mathscr{X}^g$. This can be written as:

$$\boldsymbol{\xi}^g = \begin{cases} \boldsymbol{\xi} & \text{if } \boldsymbol{\xi} \in \mathscr{X}^g \\ \text{argmin}_{\hat{\boldsymbol{\xi}} \in \mathscr{X}^g} \|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}\| & \text{else if } \frac{\|\boldsymbol{\xi}^{c,g} - \boldsymbol{\xi}\|}{\|\boldsymbol{\xi}^{c,g} - \boldsymbol{\xi}^r\|} > 1 \\ \boldsymbol{\xi}^r & \text{otherwise} \end{cases} \tag{5.9}$$

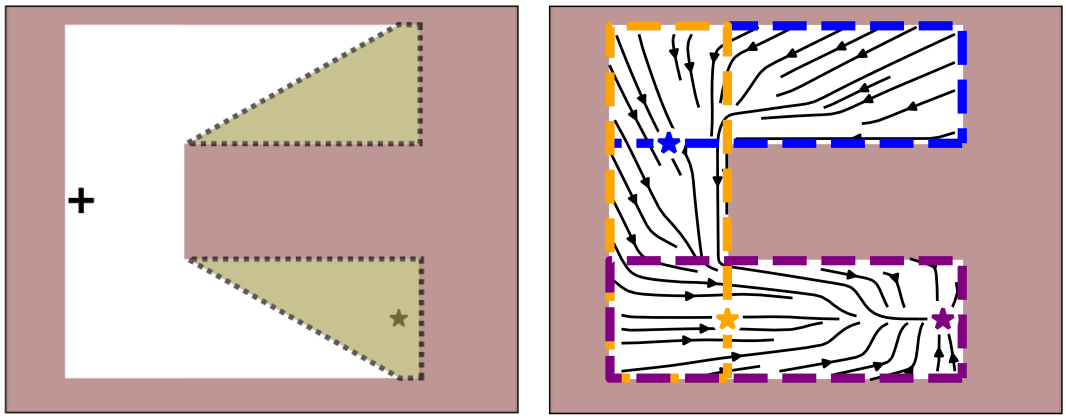
with $\boldsymbol{\xi}^{c,g}$ the center point of the gap (see Fig. 5.5).

5.3.4 Concave Environments

The presented obstacle avoidance algorithm applies to *star-shaped* environments. The extension of environments with various concave obstacles to fulfill the constraints is described in (L. Huber, Billard, and J.-J. Slotine, 2019).

Similarly, the presented method for inverted obstacles requires *star-shaped* boundaries. A general concave wall can be extended to meet the constraints (see Fig. 5.6a). However, this extension results in certain regions not being accessible anymore, i.e., the yellow patches.

Alternatively, we propose to combine the modulation avoidance algorithm with a high-level planner. The planner divides the boundary into several sub-boundaries with corresponding local dynamics. For this, we use the method to pass gaps in a wall as described in Sec. 5.3.3 to pass



(a) Boundary extensions (b) Consecutive dynamics

Figure 5.6: For the navigation in a concave corridor (brown) the boundary can be extended to be of *star-shape*, see the yellow patches in (a). Alternatively, the motion can be divided into partial boundaries, the blue, orange, and purple rectangles. They each have a corresponding dynamical systems with attractors, see the black stars (b).

from one local environment to the next one. The modulated DS can be summed according to the local weights:

$$w_i^p(\boldsymbol{\xi}) = \begin{cases} \hat{w}_i^p(\boldsymbol{\xi}) / \sum_i \hat{w}_b^p(\boldsymbol{\xi}) & \text{if } \sum_i \hat{w}_i^p(\boldsymbol{\xi}) > 0 \\ \hat{w}_i^p(\boldsymbol{\xi}) = 0 & \text{otherwise} \end{cases} \tag{5.10}$$

$$\text{with } \hat{w}_i^p(\boldsymbol{\xi}) = \max\left(\Gamma_i(\boldsymbol{\xi}) - 1, 0\right) \tag{5.11}$$

The high-level planners which decompose the environment autonomously is part of ongoing research.

5.4 Nonsmooth Surfaces

Human-designed environments often contain obstacles and enclosing walls with nonsmooth surfaces, e.g., a table with edges or a building with corners. An approximation with a high gradient of these surfaces can lead to undesired results. On the one hand, a smoothing of the edges increases the risk of colliding with them. On the other hand, an increased, smooth hull conservatively increases the boundary region, and certain parts of the space are not reachable with such a controller.
Moreover, the obstacle avoidance algorithm applied to a surface with a high gradient can lead to a fast change of the flow. Even though the trajectories are smooth, the curvature of the flow can be high and induce fast accelerations. This can result in dangerous behavior in the presence of humans or simply exceed the robot's torque limits. We propose an approach to avoid obstacles

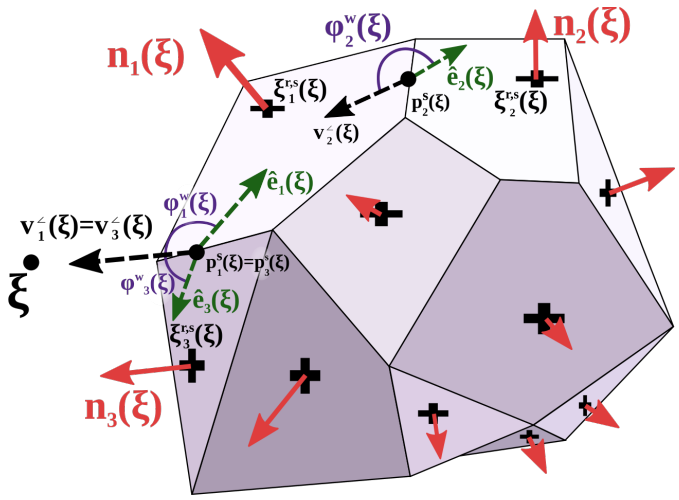


Figure 5.7: The variables for the evaluation of the pseudo normal $\hat{\boldsymbol{n}}(\boldsymbol{\xi})$ of a nonsmooth star-shaped obstacle are displayed for three surface tiles. Only the surface reference point (cross) and the surface normal (red arrow) are visualized for the other tiles. The angle φ^w is evaluated for each surface at the edge-point closest to $\boldsymbol{\xi}$.

with nonsmooth surfaces without smoothing the boundary.
A polygonal obstacle consists of $i = 1...N^s$ individually smooth surface planes which form a star shape in $d-1$ such that:

$$\mathcal{X}_i^s = \{\boldsymbol{\xi}, \hat{\boldsymbol{\xi}} \in \mathcal{X}^b, \exists \boldsymbol{n}_i : \langle \boldsymbol{n}_i, (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}) \rangle = 0\} \tag{5.12}$$

5.4.1 Pseudo Normal Vector

The normal to the surface of the obstacle is not defined continuously. As a result, the modulated flow would not be smooth.
We create a smoothly defined pseudo normal $\hat{\boldsymbol{n}}(\boldsymbol{\xi})$. It is equal to the normal on the surface of the obstacle. While far away from the obstacle, the pseudo normal approaches the reference direction, i.e.:

$$\hat{\boldsymbol{n}}(\boldsymbol{\xi}) = \boldsymbol{n}_i(\boldsymbol{\xi}) \;\; \forall \boldsymbol{\xi} \in \mathcal{X}_i^s \quad \text{and} \quad \lim_{\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| \to \infty} \hat{\boldsymbol{n}}(\boldsymbol{\xi}) = \boldsymbol{r}(\boldsymbol{\xi}) \tag{5.13}$$

The pseudo normal is the weighted sum of the normals, with the weights being evaluated as follows. At first the vector from the closest edge point $\boldsymbol{p}_i^s$ (Fig. 5.7) to the agent's state is created:

$$\boldsymbol{v}_i^{\angle}(\boldsymbol{\xi}) = \boldsymbol{\xi} - \boldsymbol{p}_i^s \;\; \text{with} \;\; \boldsymbol{p}_i^s = \underset{\hat{\boldsymbol{\xi}} \in \mathcal{X}_i^e}{\operatorname{argmin}} \|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}\| \tag{5.14}$$

with $\mathcal{X}_i^e$ the set of all points at the edge of a surface tile i. This vector is further projected onto

the surface plane:

$$\hat{\boldsymbol{e}}_i(\boldsymbol{\xi}) = \left(\boldsymbol{v}_i^{\angle}(\boldsymbol{\xi}) - \langle \boldsymbol{n}_i(\boldsymbol{\xi}),\, \boldsymbol{v}_i^{\angle}(\boldsymbol{\xi})\rangle\, \boldsymbol{n}_i(\boldsymbol{\xi})\right) \operatorname{sign}\langle \boldsymbol{v}_i^{\angle}(\boldsymbol{\xi}),\, \boldsymbol{\xi} - \boldsymbol{\xi}_i^{r,s}\rangle$$

The angle to the plane (Fig. 5.7) in the range $[0, \pi]$ is evaluated as:

$$\varphi_i^w(\boldsymbol{\xi}) = \arccos\left(\frac{\langle \hat{\boldsymbol{e}}_i(\boldsymbol{\xi}),\, \boldsymbol{v}_i^{\angle}(\boldsymbol{\xi})\rangle}{\|\hat{\boldsymbol{e}}_i(\boldsymbol{\xi})\|\, \|\boldsymbol{v}_i^{\angle}(\boldsymbol{\xi})\|}\right) \operatorname{sign}\langle \boldsymbol{n}_i(\boldsymbol{\xi}),\, \boldsymbol{v}_i^{\angle}(\boldsymbol{\xi})\rangle \tag{5.15}$$

The edge-weight is evaluated as:

$$\tilde{w}_i^s(\boldsymbol{\xi}) = \begin{cases} \left(\frac{\pi}{\varphi_i^w(\boldsymbol{\xi})}\right)^p - 1 & \text{if } \varphi_i^w(\boldsymbol{\xi}) \in]0, \pi] \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

with the the weight power $p \in \mathbb{R}$, a free parameter. A lower weight power p results in an increased importance for the closest polygon face compared to the other faces, we simply choose $p = 3$. The final step is the normalization of the weights:

$$w_i^s(\boldsymbol{\xi}) = \begin{cases} \tilde{w}_i^s(\boldsymbol{\xi}) / \sum_j \tilde{w}_j^s(\boldsymbol{\xi}) & \text{if } \boldsymbol{\xi} \in \mathbb{R}^d \setminus \mathcal{X}_i^s \\ 1 & \text{otherwise} \end{cases} \tag{5.17}$$

The pseudo normal is evaluated as the directional weighted mean (see Section 4.1.1) of normal vectors of the surface tiles $\boldsymbol{n}_i(\boldsymbol{\xi})$, the weights w_i^s, and with respect to the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$. The basis matrix from (3.14) is redefined for nonsmooth surfaces as:

$$\boldsymbol{E}(\boldsymbol{\xi}) = [\boldsymbol{r}(\boldsymbol{\xi})\ \ \hat{\boldsymbol{e}}_1(\boldsymbol{\xi})\ ..\ \hat{\boldsymbol{e}}_{d-1}(\boldsymbol{\xi})] \tag{5.18}$$

with $\hat{\boldsymbol{e}}_i(\boldsymbol{\xi})$ the orthonormal basis to $\hat{\boldsymbol{n}}(\boldsymbol{\xi})$. The resulting smooth vectorfield can be observed in Fig. 5.9.

5.4.2 Inverted Obstacles

For an inverted obstacle (Sec. 5.3) the pseudo normal is evaluated at the mirrored position $\boldsymbol{\xi}^{\text{mir}}$. It is obtained by flipping the current robot state $\boldsymbol{\xi}$ along the reference direction $\boldsymbol{r}(\boldsymbol{\xi}) = \boldsymbol{\xi} - \boldsymbol{\xi}^r$ onto the other side of the boundary (3.9):

$$\boldsymbol{\xi}^{\text{mir}} = \Gamma(\boldsymbol{\xi})^2 \left(\boldsymbol{\xi} - \boldsymbol{\xi}^r\right) + \boldsymbol{\xi}^r \tag{5.19}$$

The mirrored position allows the evaluation of the distance function as described in Sec. 5.4.1. Further, the inverted obstacle is treated as described in Sec. 5.3. This allows avoiding nonsmooth obstacles and boundaries in Fig. A.1.

Theorem 5.4.1. *Consider a polygon composed of N^s surfaces as given in* (5.12) *or alternatively*

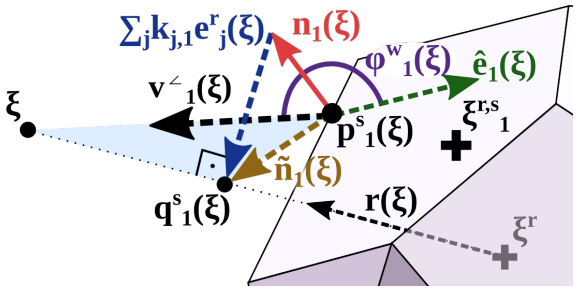


Figure 5.8: Visualization of variables used for the weighted directional mean.

an inverted polygon as described Sec. 5.4.2. Any trajectory $\{\boldsymbol{\xi}\}_t$*, that starts in free space, i.e.* $\Gamma(\{\boldsymbol{\xi}\}_0) > 1$ *and evolves on a smooth path according to (3.13), will never reach the surface, i.e.* $\Gamma(\{\boldsymbol{\xi}\}_t) > 1, t = 0..\infty$ *and will converge towards the attractor as long as it is placed outside all obstacles, i.e.* $\lim_{t\to\infty} \boldsymbol{\xi} \to \boldsymbol{\xi}^a \in \mathcal{X}^f$.

Proof. We show first that the modulation has full rank and hence that the dynamics does not vanish outside the attractor and that it is smooth.

Full Rank

The basis matrix from (5.18) has full rank everywhere outside of the obstacle, if the following condition holds:

$$\arccos\left(\langle \boldsymbol{r}(\xi), \hat{\boldsymbol{n}}(\xi)\rangle\right) < \pi/2 \tag{5.20}$$

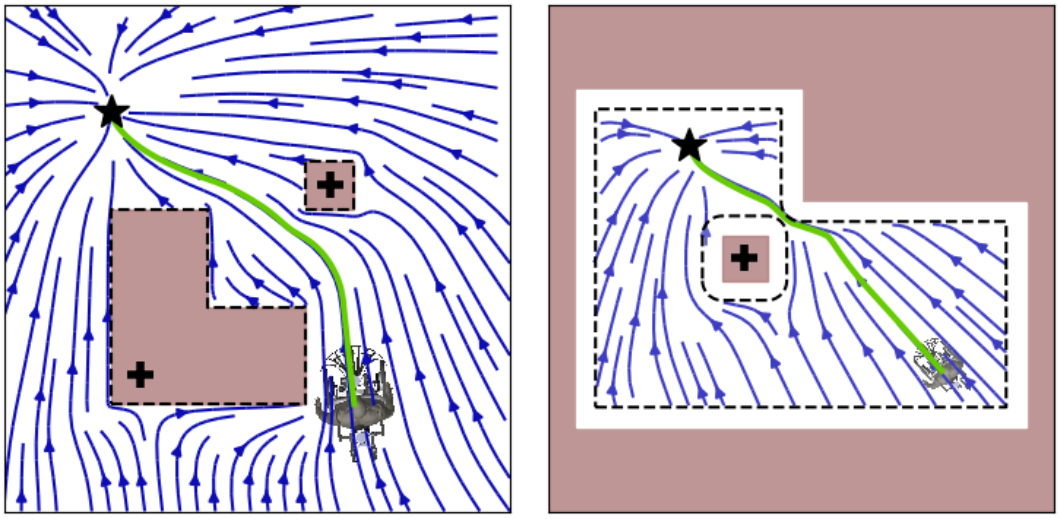
The angle between the normal to each surface i, $\boldsymbol{n}_i(\xi)$ and the reference direction $\boldsymbol{r}(\xi)$ can be evaluated by defining an vector $\tilde{\boldsymbol{n}}_i(\xi) = \boldsymbol{n}_i(\xi) + \sum_{j=1}^{d-1} k_i^e \boldsymbol{e}_j(\xi)$ with $\langle \boldsymbol{e}_j(\xi), \boldsymbol{r}(\xi)\rangle = 0$, $k_i^e \in \mathbb{R}$ such that $\boldsymbol{p}_i^s(\xi) + \tilde{\boldsymbol{n}}_i(\xi)$ intersects with $\xi^r + k^r \boldsymbol{r}(\xi)$ at $\boldsymbol{q}_i^s(\xi)$ with $k^r \in \mathbb{R}$ (Fig. 5.8).
This allows to create a triangle spanned by the lines ξ, $\boldsymbol{p}_i^s(\xi)$ and $\boldsymbol{q}_i^s(\xi)$, colored in blue in Fig. 5.8. Using the associative law of the dot product, the geometry constraint of the blue triangle and (5.16), the maximum angle results in:

$$\langle \boldsymbol{n}_i, \boldsymbol{r}\rangle = \langle \tilde{\boldsymbol{n}}_i, \boldsymbol{r}\rangle \geq \langle \xi - \boldsymbol{p}_i^s, \boldsymbol{r}\rangle \geq 0 \quad \forall\, w_i(\xi) > 0 \tag{5.21}$$

Hence the directional transformation of (4.4) results in $\|\kappa_i\| < \pi/2\|$, $\forall\, w_i(\xi) > 0$. Using additionally the *triangle equality* for vectors: $\|\kappa_1 + \kappa_2\| \leq \|\kappa_1\| + \|\kappa_2\|$ applied to all surface directions, it follows with (4.5) that:

$$\|\bar{\kappa}\| = \|\sum_{i=1}^{N^v} w_i \kappa_i\| \leq \sum_{i=1}^{N^v} w_i \|\kappa_i\| \leq \left(\sum_{i=1}^{N^v} w_i\right) \|\max_{i \text{ with } w_i>0} \kappa_i\| \leq \frac{\pi}{2}$$

Since the basis vector of the directional mean is $\boldsymbol{r}(\xi)$, with (4.6) condition (5.20) holds true.



(a) Nonsmooth obstacles (b) Nonsmooth boundaries

Figure 5.9: Nonsmooth inverted obstacles representing rooms or boundary conditions.

Smooth Vector Field
The continuous extension across the reference point of Theorem 5.3.1 is applicable, too. The reference direction $\boldsymbol{r}(\xi)$ and the distance function $\Gamma(\xi)$ does not have any other discontinuity. The pseudo normal $\hat{\boldsymbol{n}}(\xi)$ is smoothly defined across space. Even in the case when the edge point with the minimum is switching (5.14), no discontinuity occurs since the angle will stay the same due to the flat surface.

Applicability of General Proofs
Since we have a smooth field of normal vectors $\boldsymbol{n}(\xi)$, we further need to define any smooth distance function which decreases its value with increasing distance. The two properties are sufficient to comply with the proof of Theorem 5.3.1. □

5.4.3 Implementation

Pseudo-normals are useful for surroundings with sharp boundaries, for example, pieces of furniture with edges or maps of buildings with sharp corners. These cases consist of polygons with a small number of faces. They allow the evaluation of the algorithm in real-time. It is designed for scenarios with obstacles known from previously learned libraries or a known map retrieved at runtime, such as the tracker of (Jia, Hermans, and Leibe, 2020).

Conversely, if the input data is a point cloud (e.g., Lidar) or an obstacle-mesh, the surface can be approximated using a standard regression technique. The surface normal can be obtained directly by taking the derivative or by learning the normal at regression time (Ao et al., 2021).

5.5 Dynamic Environments

In changing environments with moving or deforming obstacles the system is modulated with respect to the relative velocity as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi}) \left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\tilde{\boldsymbol{\xi}}}^{\text{tot}} \right) + \dot{\tilde{\boldsymbol{\xi}}}^{\text{tot}} \tag{5.22}$$

The local, relative velocity is summed up over all obstacles

$$\dot{\tilde{\boldsymbol{\xi}}}^{\text{tot}} = \sum_{o=1}^{N_o} w_o^l \dot{\tilde{\boldsymbol{\xi}}}_o \tag{5.23}$$

with the dynamic weight being a function of the distance $\Gamma(\boldsymbol{\xi})$:

$$w_o^l = \frac{w_o^l}{\sum_o \tilde{w}_o^l} \quad \text{with } \tilde{w}_o^l = \frac{1}{\Gamma_o(\boldsymbol{\xi}) - 1} \quad \forall\, \Gamma_o(\boldsymbol{\xi}) > 1 \tag{5.24}$$

The relative velocity consists of the obstacle's velocity $\dot{\tilde{\boldsymbol{\xi}}}_o^v$ and deformation $\dot{\tilde{\boldsymbol{\xi}}}_o^d$:

$$\dot{\tilde{\boldsymbol{\xi}}}_o = \dot{\tilde{\boldsymbol{\xi}}}_o^v + \dot{\tilde{\boldsymbol{\xi}}}_o^d \tag{5.25}$$

Note that avoiding dynamic obstacles is not only a modulation of the DS, i.e., a matrix multiplication. This can result in the velocity at the attractor being non-zero, even though the initial dynamical system has zero value there: $\boldsymbol{f}(\boldsymbol{\xi}^a) = \boldsymbol{0}$.
For the rest of this section, we will assume the application to each obstacle implicitly without using the subscript $(\cdot)_o$.

5.5.1 Moving Obstacles

The relative velocity of a moving obstacle is obtained similarly to (S. Khansari-Zadeh, 2012):

$$\dot{\tilde{\boldsymbol{\xi}}}^v = \dot{\boldsymbol{\xi}}^{L,v} + \dot{\boldsymbol{\xi}}^{R,v} \times \tilde{\boldsymbol{\xi}} \tag{5.26}$$

the linear velocity $\dot{\boldsymbol{\xi}}^{L,v}$ and angular velocity $\dot{\boldsymbol{\xi}}^{R,v}$ are with respect to the center point of the obstacle $\boldsymbol{\xi}^c$. The relative position is $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}^c$.

5.5.2 Deforming Obstacle

Obstacles and hulls can not only move, but they can also change their shape with respect to time, e.g. breathing body for a surgery robot. Conversely, the deformation of the perceived obstacle can be the result of uncertainties in real-time obstacle detection and position estimation.
The deformation velocity of an obstacle is evaluated as:

$$\dot{\tilde{\boldsymbol{\xi}}}^d = \dot{\boldsymbol{\xi}}^{L,d} + \dot{\boldsymbol{\xi}}^{R,d} \times \tilde{\boldsymbol{\xi}}^d \tag{5.27}$$

Where the linear velocity $\dot{\boldsymbol{\xi}}^{L,d}$ and angular velocity $\dot{\boldsymbol{\xi}}^{R,d}$ are evaluated on the surface position in reference direction, and $\tilde{\boldsymbol{\xi}}^d = \boldsymbol{\xi} - \boldsymbol{\xi}^b$ is the relative position with respect to the boundary point (see Fig. 5.10).
The surface deformation should be explicitly given to the algorithm whenever it is known. Alternatively, it can be estimated from sensor readings, such as Lidar or camera.

Repulsive Mode

In many scenarios, the consideration of the obstacle's deformation is only of importance when it reduces the robot's workspace, and puts the robot at risk. It is sufficient to consider the deformation only along positive normal direction. For example for a circular object, we have:

$$\dot{\tilde{\boldsymbol{\xi}}}^d = \dot{\boldsymbol{\xi}}^{L,d} = \begin{cases} \dot{r}\, \boldsymbol{n}(\boldsymbol{\xi}) & \dot{r} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.28}$$

where $\dot{r}$ is the rate of change of the circle radius.

5.5.3 Impenetrability with Respect to Maximum Velocity

Many agents have a maximum velocity they can move with, further referred to as $v^{\max}$. This limits the motion of obstacles which an agent can avoid to:

$$v^n = \langle \dot{\tilde{\boldsymbol{\xi}}}, \boldsymbol{n}(\boldsymbol{\xi}) \rangle < v^{\max} \qquad \text{as} \quad \Gamma(\boldsymbol{\xi}) \to 1 \tag{5.29}$$

When close to an obstacle, the agent must prioritize moving away from the obstacle over following the desired motion. Since the modulated velocity $\dot{\boldsymbol{\xi}}$, see (3.13), does not take into account the maximal velocity, smart cropping needs to be applied. We propose the following method, which prioritizes avoidance in critical situations but sticks to the initial DS as when safe:

$$\dot{\boldsymbol{\xi}}^s = \begin{cases} v^n \boldsymbol{n}(\boldsymbol{\xi}) + v^e\, \boldsymbol{e}(\boldsymbol{\xi}) & \text{if } \langle \frac{\dot{\boldsymbol{\xi}}}{\|\dot{\boldsymbol{\xi}}\|}, \boldsymbol{n}(\boldsymbol{\xi}) \rangle < \frac{v^n}{v^{\max}} \\ v^{\max} \dot{\boldsymbol{\xi}} / \|\dot{\boldsymbol{\xi}}\| & \text{else if } \|\dot{\boldsymbol{\xi}}\| > v^{\max} \\ \dot{\boldsymbol{\xi}} & \text{otherwise} \end{cases} \tag{5.30}$$

with $v^e = \sqrt{(v^{\max})^2 - (v^n)^2}$, see Fig. 5.10.

Theorem 5.5.1. *Consider the dynamic environment with N^{obs} obstacles which have a weighted local velocity $\dot{\tilde{\boldsymbol{\xi}}}^{\text{tot}}$, resulting from the obstacles' movements and surface deformations, defined*

in (5.23). An agent is moving in this space and has a maximum velocity of $v^{\max}$, further the obstacles' surface velocities are limited by (5.29). The agent which starts in free space, i.e. $\Gamma_o(\{\boldsymbol{\xi}\}_0) > 1,\ \forall o \in N^{\text{obs}}$ and moves according to (5.30), will stay in free space for infinite time, i.e. $\Gamma_o(\{\boldsymbol{\xi}\}_t) > 1,\ t = 0..\infty,\ \forall o \in N^{\text{obs}}$.

Proof. As an agent approaches the surface of an obstacle, the importance weight from this obstacle is approaching one, i.e., $\lim_{\Gamma_{\hat{o}}(\xi)\to 1} w_{\hat{o}} = 1$, see (5.24). It follows with (5.23) that the relative velocity is $\dot{\tilde{\xi}} = \dot{\tilde{\xi}}^{\text{tot}} = \dot{\tilde{\xi}}_{\hat{o}}$.
As a result it is sufficient to analyze impenetrability for each obstacle individually. The next step is to ensure impenetrability of the three cases in (5.30):

Evaluation in Moving Frame:

$\dot{\xi} = \dot{\xi}$
The simplest case comes with no stretching, but the evaluation in the local frame of the moving boundary of the obstacle. It follows that the Neuman-boundary condition for impenetrability holds (see also (L. Huber, Billard, and J.-J. Slotine, 2019)).

Contraction within Margin

$\dot{\xi} = v^{\max}\|\dot{\xi}\|\dot{\xi}$
This contraction is only performed, if it results in a normal velocity which is larger than the velocity of the obstacle $\dot{\tilde{\xi}}$, i.e. the evaluation in the moving frame results $\langle(\dot{\xi} - \dot{\tilde{\xi}}, \boldsymbol{n}(\xi)\rangle \geq 0$, hence ensuring impenetrability.

Contraction in Tangent Direction

$$\dot{\xi} = v^n \boldsymbol{n}(\xi) + \sqrt{(v^{\max})^2 - \|\dot{\xi}^n\|^2}\, \boldsymbol{e}(\xi) \tag{5.31}$$

This limited contraction along the normal direction ensures that the velocity in normal direction remains equal to the obstacles' velocity. The evaluation of the Neuman boundary condition in the moving frame leads to:

$$\langle(v^n \boldsymbol{n}(\xi) + \sqrt{(v^{\max})^2 - \|\dot{\xi}^n\|^2}\, \boldsymbol{e}(\xi)) - \dot{\tilde{\xi}}, \boldsymbol{n}(\xi)\rangle = v^n - v^n = 0 \tag{5.32}$$

using the definition of (5.29) and the fact that the normal $\boldsymbol{n}(\xi)$ and the tangent $\boldsymbol{e}(\xi)$ are orthogonal.

Hence, we have impenetrability for multiple obstacles. □

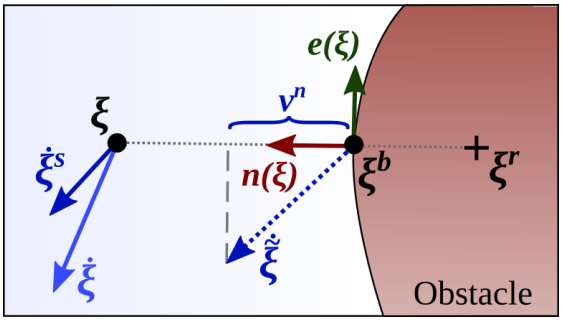


Figure 5.10: In order to comply with a velocity limit of the robot while avoiding a collision with an obstacle of velocity $\dot{\boldsymbol{\xi}}$, the modulated velocity $\dot{\boldsymbol{\xi}}$ might be stretched only in normal direction to obtain the safe velocity command $\dot{\boldsymbol{\xi}}^s$.

5.5.4 Reference Point Placement

Dynamic Extension of Hull

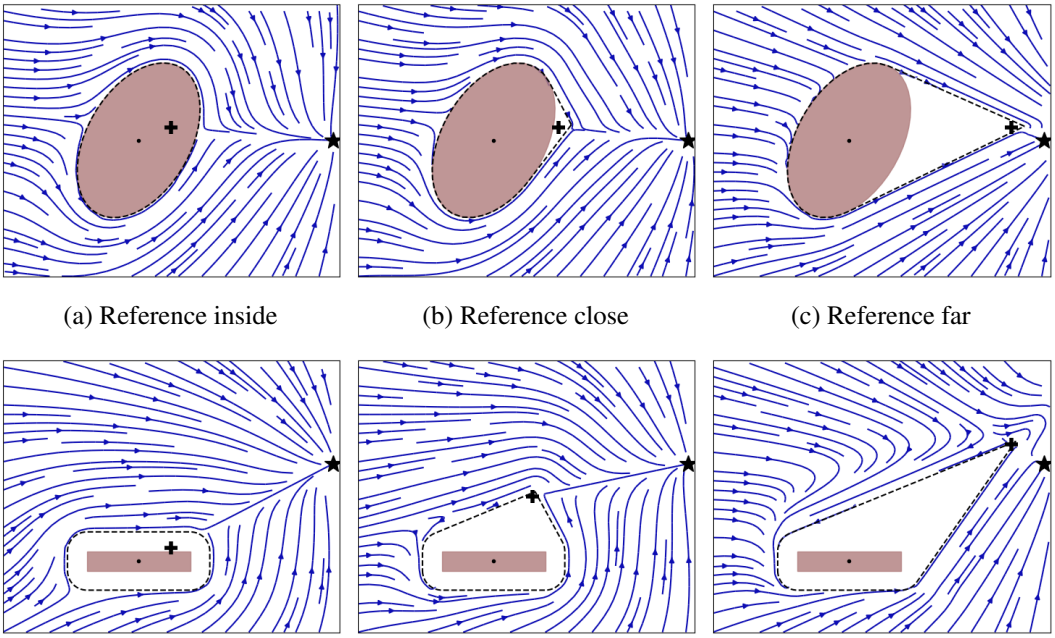
Clusters of more than two convex obstacles do often not form a *star-shape*. In such cases, we propose to extend the hull of each obstacle such that they all include a common reference point. The new hull is designed to be convex for each obstacle, and hence the cluster is star-shaped. The extended hull creates a cone that is tangent to the obstacle's surface and has the reference point at its tip (Fig. 5.11). Since the clusters are *star-shapes*, there is a global convergence of the vector field towards the attractor.
The extension of the surfaces can be done dynamically, as a collision-free trajectory is ensured around deforming obstacles (Sec. 5.5.2).

Cluttered Environments with Obstacles and Boundaries

For obstacles which intersect with the boundary, the reference point has to be placed inside the wall, i.e., $\Gamma_b(\boldsymbol{\xi}_o^r) < 1$ (see Fig. 5.12). This enforces all trajectories to avoid the intersecting obstacles by moving away from the wall (counterclockwise in this example). The boundary modulates them in the same direction. Hence, there is full convergence of all trajectories towards the attractor. This is true for a boundary-obstacle with a positive (local) curvature:

$$c^b(\boldsymbol{\xi}) > 0 \qquad \forall\, \boldsymbol{\xi} \tag{5.33}$$

with the curvature given defined in (4.14).



(a) Reference inside (b) Reference close (c) Reference far

(d) Reference inside (e) Reference close (f) Reference far

Figure 5.11: Dynamic extension of the hull for an ellipsoid object without margin (a)-(c) and a nonsmooth polygon object with constant margin (d)-(e).

5.6 Comparison Algorithms

5.6.1 Qualitative Comparison

We have selected multiple time-invariant, local obstacle avoidance algorithms for qualitative comparison with the presented method Table 5.1. **Dynamic** (proposed) method is compared to **Reference** (L. Huber, Billard, and J.-J. Slotine, 2019), Orthogonal (S. M. Khansari-Zadeh and Billard, 2012), Repulsion (Khatib, 1986), Navigation Functions D. E. Koditschek and Rimon, 1990; Rimon and D. E. Koditschek, 1991; Rimon and D. Koditschek, 1992, **Lyapunov QP** (Reis, Aguiar, and Tabuada, 2020), **Sphere World QP** (Notomista and Saveriano, 2021) and **Danger Fields** (Lacevic, Rocco, and Zanchettin, 2013; Zanchettin, Lacevic, and Rocco, 2015). The *Navigation Functions*, *Lyapunov QP*, *Sphere World QP* and *Danger Fields* all rely on critical tuning parameters. This is a Lyapunov function for the *Lyapunov QP*, and parameters for the diffeomorphic transformation or navigation function for the other three methods. While the parameters and functions can be obtained through manual tuning, no solution exists to set them in real-time automatically. Hence, these methods cannot be easily applied to dynamic environments. Navigation functions and the diffeomorphic transform are defined globally. Their tuning parameter depends on the distribution of the obstacles all across space. Two obstacles close together far from an agent will determine the possible tuning parameters and influence the avoidance behavior. As a result, the methods cannot be transferred easily to cluttered dynamic environments.
While other methods have already allowed navigation inside walls, this has not been done in

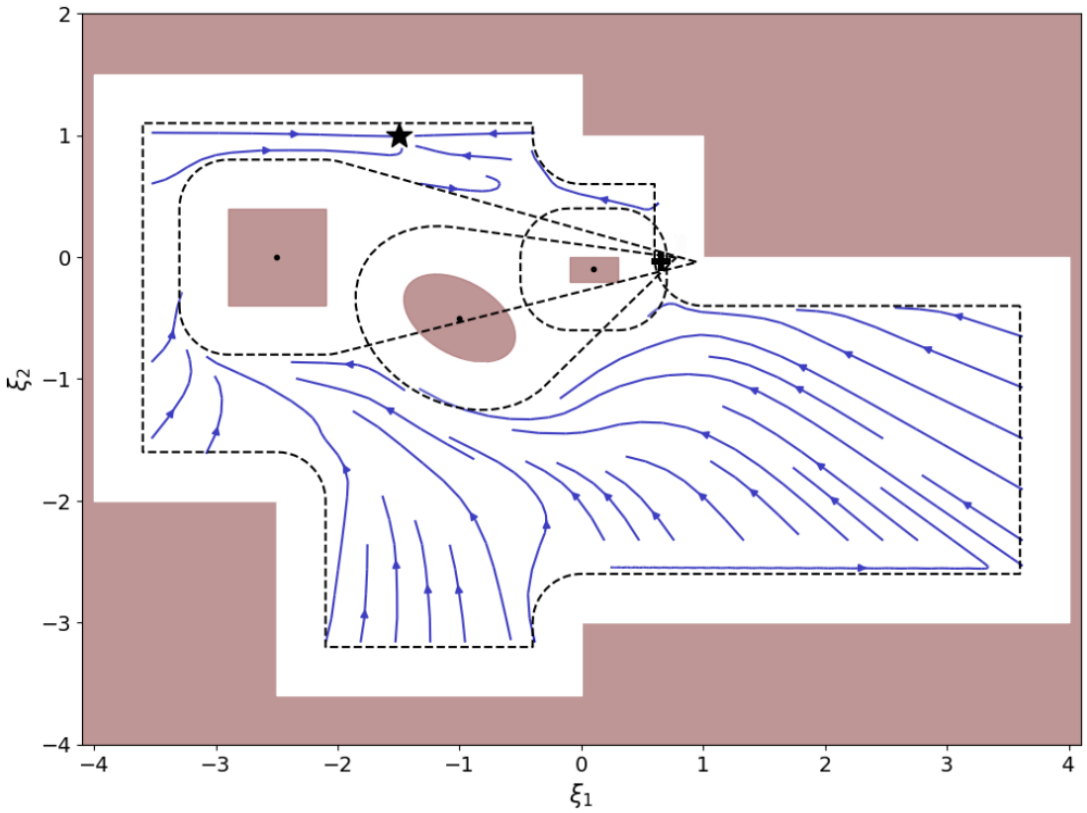


Figure 5.12: Full convergence towards the attractor (black star) in an environment of three obstacles intersecting with the boundary.

combination with proven *star-shape* world convergence and dynamic surroundings.

5.6.2 Quantitative Comparison

For a quantitative comparison, we chose algorithms that can function in cluttered, dynamic environments and can handle external hulls (see Table 5.1).
The method for the modulation algorithm in dynamic environments is presented in this paper (referred as *Dynamic* during this section). It is compared to (S. M. Khansari-Zadeh and Billard, 2012), which uses modulation matrix based on an orthogonal decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ (referred as *Orthogonal*) and the potential field algorithm (Khatib, 1986) (referred to as *Repulsion*).
The comparison is made in a simulated environment (Fig. 5.13). Two ellipse-shaped obstacles randomly change shape, and the movement is inspired by *random walk*. The combined maximum expansion and obstacle's velocity are lower than the maximum speed of the agents of $v^{\max} = 1\,\mathrm{m/s}$. The three algorithms are given the same attractor as a goal. They start at the same time and encounter the same environment.
The *Dynamic* algorithm is observed to have the highest rate of convergence (Tab. 5.2) resulting from the increased environment information through the reference point. The *Repulsion* has a

	Dynamic	Reference	Orthogal	Repulsion	Navigation Functions	Lyapunov QP	Sphere World QP	Danger Fields
Time invariant	✓	✓	✓	✓	✓	✓	✓	✓
History invariant	✓	✓	✓	✓	✓			✓
Convergence: convex	✓	✓	(✓)		✓	✓	✓	✓
Convergence: star-shape	✓	✓			✓	(✓)	✓	✓
No critical tuning parameter (incl. Lyapunov function)	✓	✓	✓	✓				
Avoidance behavior independent of global distribution	✓	✓	✓	✓		✓		
Closed from solution (no optimization)	✓	✓	✓	✓	✓			✓
Considering initial dynamics (not goal position only)	(✓)	(✓)	(✓)	✓		✓	✓	(✓)
Navigation inside walls	✓		✓	✓	✓	✓	✓	✓
Smooth motion around corners	✓			✓	✓	✓	✓	✓
Cluttered dynamic environment	✓	(✓)	(✓)	✓				

Table 5.1: The proposed dynamic obstacle avoidance is compared to different state-of-the-art methods. The last three items refer to the main contributions of this work.

	Converged	Collided	Minimum
Dynamic	77%	23%	0%
Orthogonal	20%	23%	57%
Repulsion	39%	1%	60%

Table 5.2: The outcome of the 300 trials runs were either full convergence, collision with an obstacle or getting stuck in a local minimum.

	d $[m]$	t $[s]$	$\bar{v}$ $[m/s]$	σ_v $[m/s]$
Dynamic	9.69 ± 1.19	1.03 ± 0.13	0.61 ± 0.05	0.34 ± 0.02
Orthogonal	10.06 ± 1.53	1.17 ± 0.21	0.55 ± 0.05	0.34 ± 0.02
Repulsion	9.8 ± 1.29	1.39 ± 0.2	0.46 ± 0.03	0.21 ± 0.02

Table 5.3: The mean and the standard deviation (after the ±) are compared for the three algorithms from the 54 trials where all three agents converged. The metrics of distance (d), duration of the run (t), the mean velocity ($\bar{v}$) and the standard deviation of the velocity (σ_v) are listed.

preferable behavior on avoiding collisions because of its conservative behavior around obstacles (moving far away and only approaching them slowly). This influences the distance traveled and the time needed to reach a goal (Tab. 5.3). The mean of the velocity is lower for the *Dynamic* algorithm. This is a result of no *tail-effect* behind the obstacles (see Sec. 5.2.2). The variance of the velocity is similar for the three algorithms.[2]

5.6.3 Computational Complexity

The presented algorithm is closed-form, whereas the *matrix inverse* is the most complex computation. Since it is applied to all obstacles, the complexity follows as $\mathcal{O}(d^{2.4}N^{\text{obs}})$, a function of the number of dimensions d and the number of obstacles N^{obs}. It had an average time of 3.48 ms on a computer with *8 Intel Core i7-6700 CPU @ 3.40GHz*.
This is more complex than the *Orthgonal*, where the matrix multiplication is the most complex operation. The complexity follows as $\mathcal{O}(d^2N^{\text{obs}})$. This reflects in the slightly faster evaluation time of 2.84 ms.
The potential field is the simplest of the three algorithms, with the norm being the most complex operation and a complexity $\mathcal{O}(dN^{\text{obs}})$. It has the lowest evaluation time of 1.75 ms.
The search for the optimal reference point is the computationally most extensive calculation because it requires the (iterative) closest-distance evaluation between obstacles. The Python library *shapely*[3] was used for the implementation, and an evaluation time of 5.14 ms was obtained.

[2]The source code of the implementation is available on https://github.com/epfl-lasa/dynamic_obstacle_avoidance
[3]https://github.com/Toblerity/Shapely

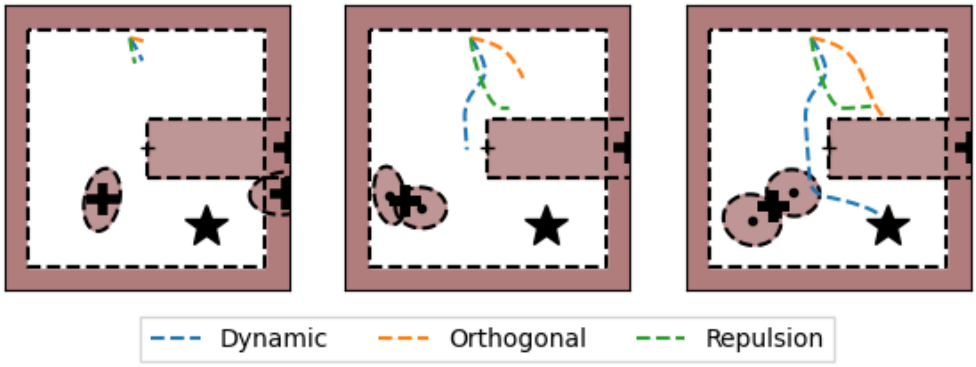


Figure 5.13: Three snapshots of one experimental run are placed from left to right. The obstacles are randomly initialized and move according to a random walk. One obstacle moved from the right to the left, while the other one was stationary. The three algorithms start at the same randomly chosen position and move towards the attractor.

5.7 Empirical Validation

The empirical validation is performed with the mobile robot *QOLO* (Granados, Kadone, and Suzuki, 2018), see Fig. 5.14; first in simulation and then on a real robot platform. QOLO, a semi-autonomous wheelchair, is designed to navigate in pedestrian environments and indoors. This platform is hence suited to test our algorithm's ability to avoid moving obstacles (pedestrians) and non-convex obstacles (walls, indoor furniture) containing sharp edges (tables, shelves). In all our experiments, we assume that QOLO has information about the goal, i.e., the attractor $\boldsymbol{\xi}^a$ of our nominal DS.[4].

5.7.1 Static Environment

We task QOLO to navigate in an office-like environment. The room is a square (5 m x 5 m), modeled as a boundary obstacle. Further, two tables are located in the room, one at the side and one at the center. The robot starts from the bottom left, and the attractor $\boldsymbol{\xi}^a$ is placed at the opposite side of the room (illustrated with a star). All objects, including the wall, are static and known a priori, and the localization is performed using the SLAM algorithm. The robot evaluates the modulated avoidance in real-time. We run the following two scenarios:

1. QOLO is in the room, and there are two possible paths to go around the center table. The dynamical system is split by the obstacle at the center (Fig. 5.15a). The robot chooses its preferred trajectory at runtime.

2. Additionally there is a (static) person in the room, which blocks the center passage.

[4]A video of the experiments is available at https://youtu.be/WKso-wu68v8

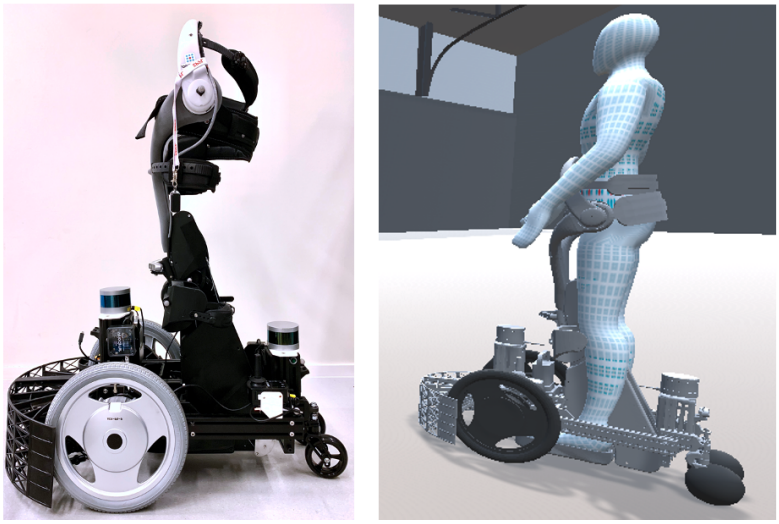


Figure 5.14: A picture of the semi-autonomous wheelchair real (left) and the simulation rendering including an operator (right).

The reference point of the obstacles is automatically placed inside the wall. The robot finds its path around the obstacles (Fig. 5.15b).

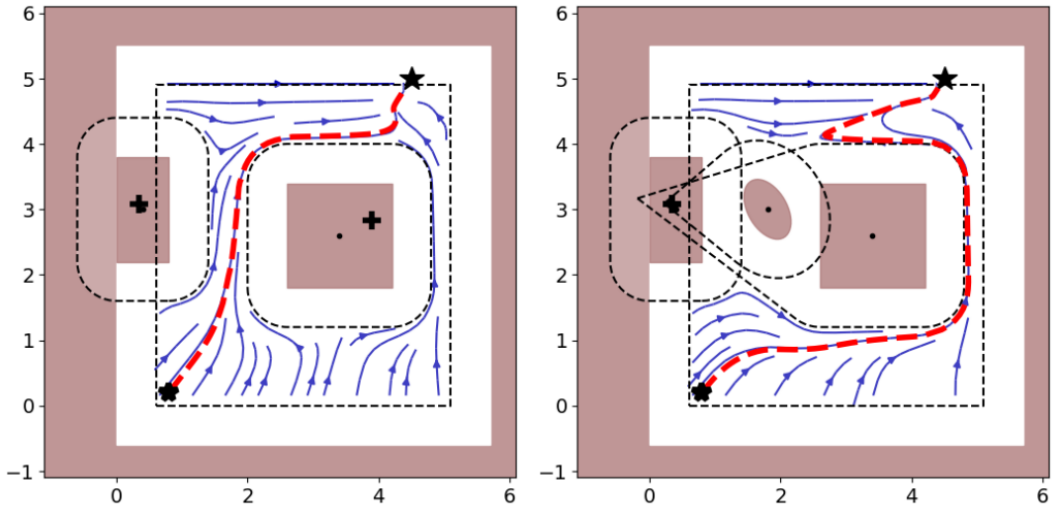
5.7.2 Dense Crowd (Simulation)

The robot is navigating in a corridor within a dense, simulated crowd. The motion of the crowd is created according to (Grzeskowiak et al., 2020). Two hundred people are moving uniformly along the 6 m wide corridor, either in the same or opposite direction as the robot.
QOLO is tasked to travel from one end of the corridor to the other. It is guided by the attractor of the nominal DS. All pedestrians are modeled as circular obstacles with radius of 0.6 m (Fig. 5.17a).
At each timestep, the problem is reduced to avoiding a subset of the pedestrians. Due to the crowd's density, the robot could realistically perceive only a subset of the pedestrians in real-time. The number of perceived people is set to $N^c = 10$. The rest of the people is hidden behind a virtual, circular wall. The center of the circular wall $\boldsymbol{\xi}^{c,w}$ is displaced from the position of the robot $\boldsymbol{\xi}^Q$ based on the remaining obstacles:

$$\boldsymbol{\xi}^{c,w} = \boldsymbol{\xi}^Q + \sum_{i=N^c+1}^{N^{\text{obs}}} \frac{\boldsymbol{\xi}_i^c - \boldsymbol{\xi}^Q}{\|\boldsymbol{\xi}_i^c - \boldsymbol{\xi}^Q\|} e^{-(\|\boldsymbol{\xi}_i^c - \boldsymbol{\xi}^Q\| - r^p - r^Q)} \tag{5.34}$$

where i is iterating over the list of the obstacles which are ordered based on their distance to the robot. The displacement factor is with respect to the radius of each pedestrian, r^p =0.6 m, and the robot radius, r^Q =0.5 m.[5]

[5]For a real-world implementation sensory distance measurements in the horizontal plane can be used to create the virtual circular wall and its displacement since the detection of people is still a time-intensive task.



(a) Office with two passages

(b) Office with passage at side

Figure 5.15: Two static office environments with two tables in a rectangular room. The center table divides the room into two passages in (a). In (b) one of the passages is blocked by a static person. Convex expansion of the hull around the reference point (black cross) ensures convergence towards the attractor (black star). The path followed by the agent is visualized in red.

The radius of the hull is chosen such that the next closest obstacle $N^c + 1$ is fully within the hull. The resulting environment has a dynamic hull with changing center-position and radius (Fig. 5.17).
Reducing the environment to only sphere obstacles decreases the computational time since there is a closed-form solution for the closest distance between two spheres. The evaluation on ROS1[6] and Python 2.7 run at around 200 Hz on a *Up Board: Intel Celeron N3350* with 2.4 GHz (CPU) and 8 GB of RAM. The number of nearby obstacles(including the wall) was eleven, while the wall remained far from the agent, it helped guide the robot around the local crowd (see Sec. 5.5.4). This is done by placing a reference point inside the wall if a crowd-cluster touches the wall (see small cluster at the bottom in Fig. 5.17b). Further, fast contraction of the boundary can happen when the local crowd density is high. This forces the obstacle to stay away from surrounding obstacles.

Quantitative Analysis

We evaluate the effect of the crowd size on the time it takes for the robot to travel through the corridor. The crowd moves along the (infinite) corridor at an average velocity of $1\,\mathrm{m\,s^{-1}}$. The simulation runs with a steady-state crowd-flow. QOLO is tasked to move in the same or opposing direction of the crowd's flow with the desired velocity of $1\,\mathrm{m\,s^{-1}}$.

[6] https://www.ros.org/

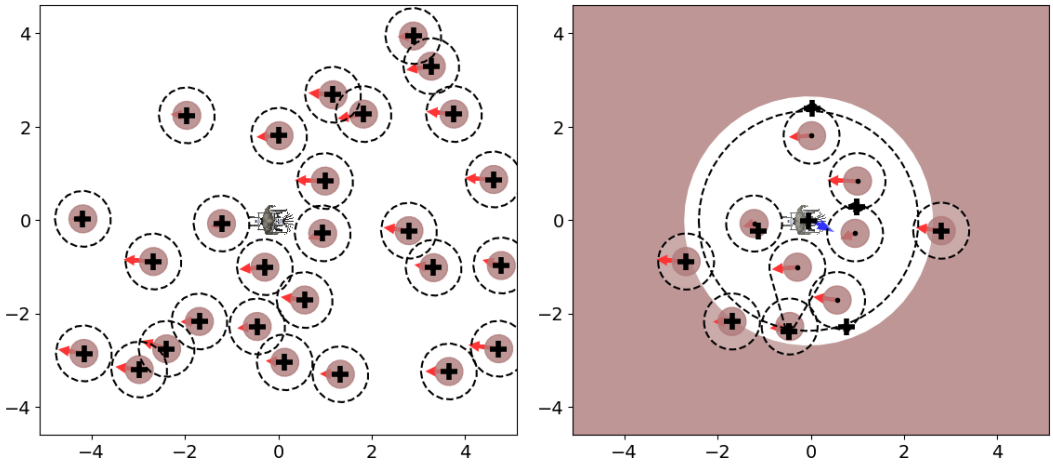
Figure 5.16: QOLO moving in a crowd.

We assess the time, speed, and distance the robot travels when moving with and opposite to the flow, see Fig. 5.18. When moving with the flow (parallel flow), the crowd does not significantly affect the distance traveled by the agent or the velocity. When moving against the flow of the crowd, a decrease in the robot's velocity can be observed for crowds denser than 20 agents per 1000 square meters. (No effect on the crowd was observed since only a single robot was moving against a large crowd.) The distance traveled increases significantly for densities above 100 agents per 1000 square meters. As a result, the average time needed to reach the goal doubles for a crowd size of 100 people compared to 2 people.
In counterflow scenarios, the standard deviation of the flow increases. This results from situations where the robot has to slow down or stop to avoid the upcoming agents.

5.7.3 Proof of concept: Outdoor Environment

A qualitative proof of concept was performed in an outdoor environment. We brought the QOLO robot to the center of Lausanne, Switzerland[7]. The robot was tasked to travel back and forth across a small marketplace (Fig. 5.19). The location is restricted to pedestrians, and six streets meet at the crossing. This results in a large diversity in the pedestrians' speed and direction of movement. The robot's controller is initialized with a linear DS to reach a goal 20 m away from the onset position. Pedestrians are detected with a camera and Lidar-based tracker developed by the authors of (Jia, Hermans, and Leibe, 2020). The tracker's output is displayed in Fig. 5.20. Recordings were taken on Saturday morning when the market was ongoing, and the crowd had a high density.
The non-holonomic constraints of QOLO are considered by evaluating the dynamical system 0.53 m in front of the center of the wheel-axes. The linear command of the robot is the velocity

[7]Approval was obtained from the EPFL Ethics board and the police of Lausanne City. A driver was on board the robot during the experiment. He could start and stop at all times. A second experimenter was watching the scene and verified the tracker's output. This was necessary in case the detector/tracker malfunctioned.



(a) All crowd agents (b) Local Obstacle Hull

Figure 5.17: The environment with many agents (left) is reduced to a scenario with ten obstacles and an enclosing hull (right).

part in the moving direction. The angular velocity follows the perpendicular part of the velocity. These velocities are provided to the low-level controller of the robot.
The geometry of QOLO is considered by placing a margin of 0.5 m around each pedestrian.
A total of five runs were executed. The robot reached its goal autonomously without intervention. The driver reported *high angular acceleration* during parts of the trip.
Post-hoc analysis of the video recordings revealed that the crowd density varied with a mean between 150 and 260 people per 1000 square meters (Fig. 5.21). The time to complete the runs ranged from 115 s to 150 s. No correlation was observed between the crowd's density and the time to reach the goal. We expect this to result from external factors influencing the run, such as the speed and direction of the crowd.
We see this as a successful proof-of-concept of the obstacle avoidance algorithm in a real crowd scenario. The crowd motion was more complex than the streamlined simulation, as people would come from all directions and would not group in a uniform flow. Moreover, the crowd included diverse pedestrians, from families with small children to elderly people.

5.8 Discussion

In this work, we have introduced a Dynamical System-based obstacle avoidance algorithm. The modulation-based approach has a theoretical proof of convergence and can be applied to higher-dimensional space. The implementations presented in this work focus on collision avoidance in two dimensions: navigation of mobile robots and obstacle avoidance for simplified robot arms.

The inverted obstacle has shown to be a great representation of mobile robots' workspace boundaries, such as room walls or the local window in dense crowd navigation. These boundaries could be interpreted as control constraints similar to control barrier functions (CBFs). Recent

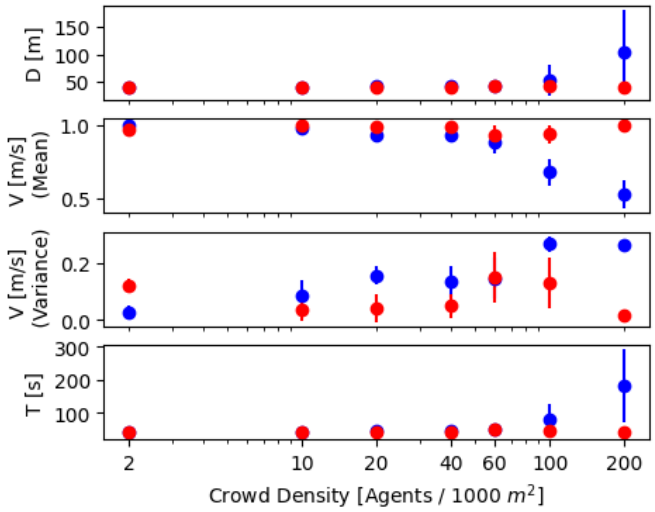


Figure 5.18: The QOLO agent is moving in parallel (red) and opposite direction (blue) to the crowd. When the robot moves with the crowd, the crowd's density has a negligible effect. When the robot moves in the opposite direction, the denser the crowd, the larger the cumulative distance (D) and mean velocity (V), as well as time (T), needed to reach the end of the corridor. The standard deviation of the velocity (V Variance) increases for dense crowds with counterflow

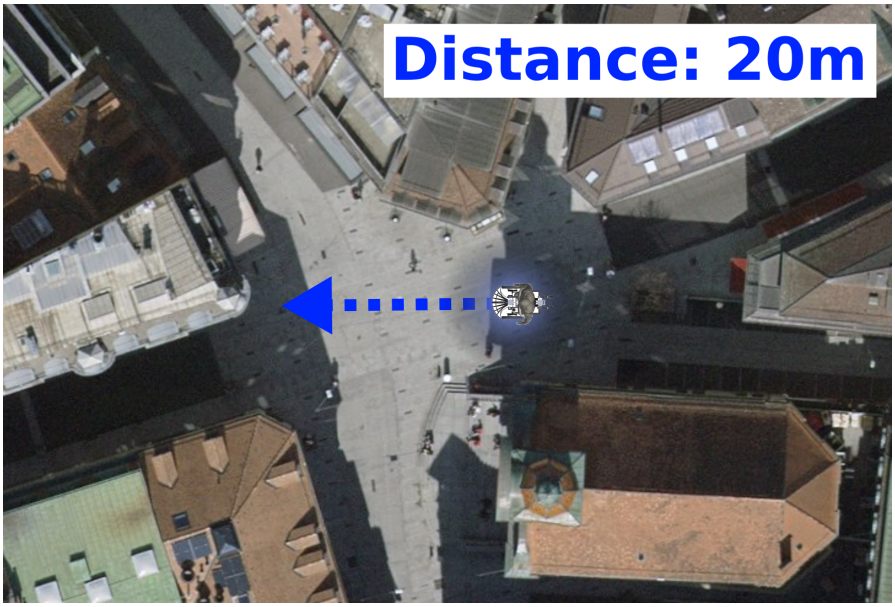
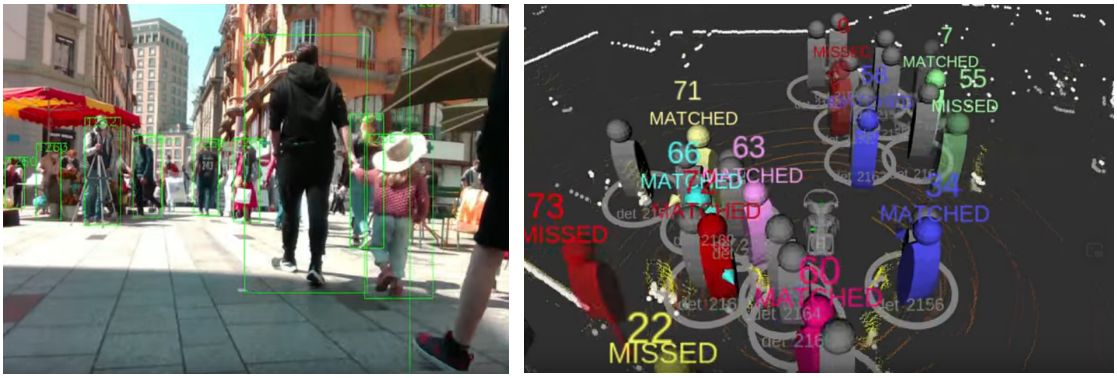


Figure 5.19: The desired path of the robot in the outdoor environment is the direct line from the robot's initial position (right) to the target position on the left.



(a) Camera view (b) Detector interpretation

Figure 5.20: The camera (a) and the LIDAR of the robot are interpreted by the detector (b), which is used for the obstacle avoidance algorithm.

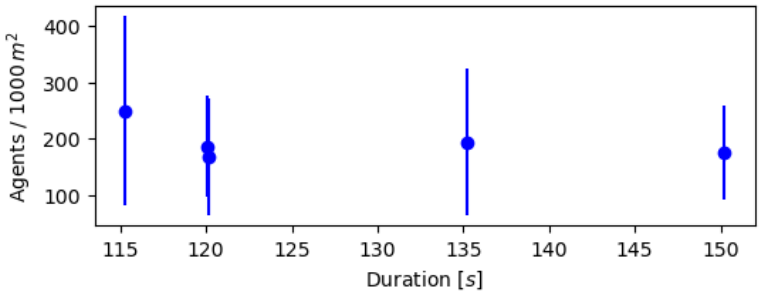


Figure 5.21: The crowd-density is highly varying during the five runs.

works have used such control barrier function in safe learning by demonstration (Ames, Coogan, et al., 2019; Taylor et al., 2020) or reinforcement learning (R. Cheng et al., 2019). Other than existing methods, we propose a closed-form solution for star-shaped barriers.

Since many human-made environments contain nonsmooth surfaces (e.g., tables, corners of rooms), the solution for providing a smooth flow without extending the hull has shown to be suitable for practical implementations. The concept of dividing dynamical systems into direction and magnitude and the presented method summing vectors to avoid local minima has been used throughout (1) to create a smooth pseudo-normal for polygon obstacles and walls and, from that, a smooth flow in their presence, (2) to sum the flow created by several obstacles without creating local minima, and (3) to solve the optimization problem to find the closest distance for pairs of obstacles. Finally, the algorithm has been successfully applied to a non-holonomic robot in a static indoor environment and dynamic outdoor crowds.

The method's advantage comes from the algorithm's low complexity and speed. Furthermore, tuning-free convergence is obtained in star-world scenarios. This will allow us to scale to higher dimensions and transfer to various scenarios.

5.8.1 Scalability and Speed

The implementation used for the experiments on QOLO was running on Python 2.7. Note that already the update to Python >3.6 gives around a 1.5-speed improvement. Switching to a compiled language like C++ can increase the speed by around ten (Fourment and Gillings, 2008). The proposed algorithm can run at a frequency of >1 kHz onboard a mobile robot. This is well above the human reaction time of around 250 ms.
The bottleneck of the current implementation is the image recognition/tracking since it was only able to run at an average frequency of around 5 Hz. This is due to the computationally heavy evaluation of the deep neural networks used for obstacle recognition. Nevertheless, we believe that the approach of separating perception and motion-planning is favorable, supported by current trends in self-driving cars and other autonomous vehicles (Badue et al., 2021; Schwarting, Alonso-Mora, and Rus, 2018).

5.8.2 Contribution

The proposed method adds value to the current state-of-the-art, not for global path planning but for reactive obstacle avoidance with convergence proofs. The modulation-based avoidance algorithm fully converges in (local) *star-shaped* environments towards the desired attractor $\boldsymbol{\xi}^a$. The behavior is similar to approaches using potential artificial fields. However, the presented approach does not require finding and deriving an artificial potential function, but the modulation directly outputs the desired velocity.
Compared to other closed-form and QP-based obstacle avoidance algorithms, our method does not require any tuning of critical parameters for its convergence or finding a Lyapunov candidate function. All parameters presented in this paper can be chosen within the defined range, and theoretical convergence is ensured.
In the presented paper (and initially introduced in (L. Huber, Billard, and J.-J. Slotine, 2019)), we presented several methods of how to place the reference point (i.e., tune its position). Even though its position is critical for convergence, it is known for *star-worlds*:

- **Convex Obstacles:** The reference point can be placed anywhere within the obstacle, i.e., $\boldsymbol{\xi}^r \in \mathscr{X}^i$.

- **Star-shapes:** The reference point has to be placed within the kernel of the obstacle. Most of the time, this is just the geometrical center of the obstacle. An algorithm to find the kernel has been described in (D.-T. Lee and Preparata, 1979) for polygons.

- **Intersecting Convex Obstacles** If several convex obstacles intersect and form a *star-shape*, the reference point can be placed anywhere at the center of all intersecting obstacles o, i.e., $\boldsymbol{\xi}^r \in \mathscr{X}_o^i \ \forall o$.
  If the convex obstacles do not form a *star-shape* or intersect with the boundary, the hull can be extended dynamically, as described in Sec. 5.5.4.

The placement of the reference point only becomes challenging when obstacles merge or separate dynamically. While we propose approximations for many cases (see Sec. 4.1.1 and Sec. 4.1.2), we do not provide a solution for all scenarios.
To the best of the authors' knowledge, no closed-form method currently exists to generate a flow around merging and dividing obstacles, which has convergence guarantees. The placement of the reference point for such dynamic scenarios is ongoing research.

5.8.3 Future Work

Future work can extend the proposed work in the following areas:

- **Low-level controller:** The low-level controller used in crowd navigation displaced the evaluation point away from the center of the robot. This resulted in an increased (conservative) margin around the robot. The way the shape of the robot is considered in the algorithm should be improved.
- **Environment recognition:** The update rate of the (deep-learning-based) tracker was approximately 5 Hz, while the present avoidance algorithm ran from 50 Hz to 100 Hz. The obstacle avoidance was often evaluated with old environment information (but updated robot position). An intermediate estimator could predict how the crowds move in between.
- **High-level planning:** Combining the fast obstacle avoidance controller with slower planning algorithms could allow handling more complex environments, i.e., avoiding surrounding (non-star-shaped) environments.
- **Evaluation in high dimensional space**: The experimental implementation is executed on an autonomous wheelchair (a two-dimensional scenario). However, this work provides a theoretical solution, which can be applied to three and higher-dimensional spaces. The next step will be the implementation and evaluation in a higher-dimensional space.

6 Fast Obstacle Avoidance Based on Real-Time Sensing

Publication Note

The material in this chapter is adapted from:
Huber, L., J.-J. Slotine, and A. Billard (2022b). "Fast obstacle avoidance based on real-time sensing". In: *IEEE Robotics and Automation Letters.*
Source Code:

- Algorithm: https://github.com/hubernikus/fast_obstacle_avoidance
- Implementation on QOLO: https://github.com/epfl-lasa/qolo_modulation.git

Multimedia: Suplementary video can be found under:

https://youtu.be/kr7R_cJoaYI

Humans excel at navigating and moving through dynamic, complex spaces like crowded streets. For robots to do the same, they must be endowed with highly reactive obstacle avoidance, and adept at partial and poor sensing. We address the issue of enabling obstacle avoidance based on sparse and asynchronous perception. The proposed control scheme combines a high-level input command provided by a planner or a human operator with fast reactive obstacle avoidance (FOA). The sampling-based sensor data can be combined with an analytical reconstruction of the obstacles for real-time collision avoidance. Thus, we can ensure that the agent does not become

stuck when a feasible path exists between obstacles. Our algorithm was evaluated experimentally on static laser data from cluttered, indoor office environments. Additionally, it was used in shared-control mode in a dynamic and complex outdoor environment in the center of Lausanne. The proposed control scheme successfully avoided collisions in both scenarios. During the experiments, the controller took 1 millisecond to evaluate over 30000 data points.

6.1 Introduction

Local avoidance algorithms must consider new sensory information in real-time and adapt their global paths to ensure safe navigation in various environments. With walking speeds in crowds of around 1.4 m/s and distances between proximal agents as low as 30 cm, robots need a reaction time, i.e., control update rate, well below a quarter of a second to ensure collision-free movement.

In this chapter, we propose a fast obstacle avoidance algorithm (FOA) with the following contributions:

- **Low computational complexity** ensures fast evaluation in cluttered environments, such as crowds (see Sec. 6.2).
- **Obstacle avoidance using sampled data** makes the method directly applicable to Lidar data (see Sec. 6.3).
- **Unifying disparate obstacle descriptions,** such as sample-based sensor data and (delayed) analytic reconstruction of obstacles, is enabled through asynchronous evaluation and dynamic weighting (see Sec. 6.4).

6.2 Fast Obstacle Avoidance

When navigating in cluttered environments, such as dense crowds requires the application of modulation for all obstacles at each time step. This results in a high computational cost which increases linearly with the number of obstacles. We propose a computationally cheaper algorithm by creating a single *virtual* obstacle which encapsulates all obstacles. The modulation is applied once for all obstacles while ensuring collision avoidance. The virtual obstacle has a single modulation matrix $\boldsymbol{M}(\boldsymbol{\xi})$ with corresponding decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ and diagonal stretching matrix $\boldsymbol{D}(\boldsymbol{\xi})$

6.2.1 Single-Modulation Obstacle Avoidance

To evaluate the decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ as defined in (3.14), we use the normalized reference direction $\boldsymbol{r}(\boldsymbol{\xi})/\|\boldsymbol{r}(\boldsymbol{\xi})\|$, and the tangent directions $\boldsymbol{e}_i(\boldsymbol{\xi})$ to form the orthonormal basis to $\boldsymbol{n}(\boldsymbol{\xi})$.

Hence, the normal is evaluated as

$$\boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{v}) = \begin{cases} \boldsymbol{E}(\boldsymbol{\xi})\boldsymbol{D}(\boldsymbol{\xi}, \boldsymbol{v})\boldsymbol{E}(\boldsymbol{\xi})^{-1} & \text{if } \|\boldsymbol{r}(\boldsymbol{\xi})\| \neq 0 \\ \boldsymbol{I} & \text{otherwise} \end{cases} \tag{6.1}$$

Note that using the identity matrix at $\|\boldsymbol{r}(\boldsymbol{\xi})\| = 0$ is the continuous expansion of the modulation; see Theorem 6.2.1.

Averaged Reference Direction

The averaged reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ is evaluated as the weighted sum:

$$\boldsymbol{r}(\boldsymbol{\xi}) = \frac{1}{\Gamma^{\min}} \sum_{o=1}^{N^{\text{obs}}} w_o(\boldsymbol{\xi}) \boldsymbol{r}_o(\boldsymbol{\xi}) \qquad \text{with} \qquad \Gamma^{\min} = \min_{o \in 1..N^{\text{obs}}} \Gamma_o(\boldsymbol{\xi}) \tag{6.2}$$

and the reference directions $\boldsymbol{r}_o(\boldsymbol{\xi})$ are evaluated for all obstacles as described in (3.16). Additionally, the influence weights $w_o(\boldsymbol{\xi})$ are given by

$$w_o(\boldsymbol{\xi}) = \begin{cases} \hat{w}_o(\boldsymbol{\xi}) / \hat{w}^{\text{sum}} & \text{if } \hat{w}^{\text{sum}} > 1 \\ \hat{w}_o(\boldsymbol{\xi}) & \text{otherwise} \end{cases}, \; w^{\text{sum}} = \sum_o \hat{w}_o(\boldsymbol{\xi}) \tag{6.3}$$

which is a function of the distance Gamma, see (3.7), and evaluated for each obstacle o as follows:

$$\hat{w}_o(\boldsymbol{\xi}) = \left(\frac{D^{\text{scal}}}{D_o(\boldsymbol{\xi})} \right)^s \qquad \text{with} \qquad D_o(\boldsymbol{\xi}) = \Gamma_o(\boldsymbol{\xi}) - 1 \tag{6.4}$$

where $s \in \mathbb{R}_{>0}$ is the scaling potential and $D^{\text{scal}} \in \mathbb{R}_{>0}$. We choose $s = 2$ and $D^{\text{scal}} = 1$.

Summing of the Normal Direction

The basis matrix $\boldsymbol{E}(\boldsymbol{\xi})$ in (3.14) needs to have full rank; thus, the tangent basis $\boldsymbol{e}_{(\cdot)}(\boldsymbol{\xi})$ and the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ have to be linearly independent (see Theorem 6.2.1). The normal direction $\boldsymbol{n}(\boldsymbol{\xi})$, which is used to obtain the tangent basis, is designed never to be perpendicular to the reference direction, i.e., $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{r}(\boldsymbol{\xi}) \rangle > 0$, $\{\boldsymbol{\xi} \in \mathbb{R}^d : \boldsymbol{r}(\boldsymbol{\xi}) \neq \boldsymbol{0}\}$.

For this, we first compute the normal offset for all obstacles:

$$\boldsymbol{n}^{\Delta}(\boldsymbol{\xi}) = \sum_{o=1}^{N^{\text{obs}}} w_o(\boldsymbol{\xi}) \left(\boldsymbol{n}_o(\boldsymbol{\xi}) - \boldsymbol{r}_o(\boldsymbol{\xi}) \right) \tag{6.5}$$

where $N^{\text{obs}} \in \mathbb{N}_{\geq 0}$ is the number of obstacles.

Finally, the weighted normal vector is obtained as:

$$\boldsymbol{n}(\boldsymbol{\xi}) = \hat{\boldsymbol{n}}(\boldsymbol{\xi}) / \|\hat{\boldsymbol{n}}(\boldsymbol{\xi})\| \quad \text{with} \quad \hat{\boldsymbol{n}}(\boldsymbol{\xi}) = c \frac{\boldsymbol{r}(\boldsymbol{\xi})}{\|\boldsymbol{r}(\boldsymbol{\xi})\|} + \boldsymbol{n}^{\Delta}(\boldsymbol{\xi}) \tag{6.6}$$

$$c = \begin{cases} 1 & \text{if } p < \frac{\sqrt{2}}{2} \\ \sqrt{2}p & \text{otherwise} \end{cases}, \quad p = (-1) \frac{\langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{n}^{\Delta}(\boldsymbol{\xi}) \rangle}{\|\boldsymbol{r}(\boldsymbol{\xi})\| \, \|\boldsymbol{n}^{\Delta}(\boldsymbol{\xi})\|}$$

The scaling factor $c \in [1, \sqrt{2}]$ ensures that the decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ is invertible.

6.2.2 Eigenvalue Matrix

The diagonal eigenvalue matrix is given as:

$$\boldsymbol{D}(\boldsymbol{\xi}) = \operatorname{diag}\left(\lambda^r(\boldsymbol{\xi}), \lambda^e(\boldsymbol{\xi}), \ldots, \lambda^e(\boldsymbol{\xi})\right) \tag{6.7}$$

with eigenvalues in reference direction $\lambda^r(\boldsymbol{\xi}) \in]0,1[$ and in tangent $\lambda^e(\boldsymbol{\xi}) \in]1,2[$ are a function of the averaged reference direction $\boldsymbol{r}(\boldsymbol{\xi})$:

$$\lambda^r(\boldsymbol{\xi}) = 1 - \|\boldsymbol{r}(\boldsymbol{\xi})\|^{\rho} \qquad \lambda^e(\boldsymbol{\xi}) = 1 + \|\boldsymbol{r}(\boldsymbol{\xi})\|^{\rho} \tag{6.8}$$

where $\rho \in \mathbb{R}_{>0}$ is the reactivity, see (3.18).

Theorem 6.2.1. *Consider a star-world as defined in Eq. (3.16) with N^{obs} obstacles. Any trajectory $\{\boldsymbol{\xi}\}_t$, which starts within the free space $\mathcal{X}^e$ and evolves according to Eq. (3.13) and (6.6), will stay in free space, i.e., $\{\boldsymbol{\xi}\}_t \in \mathcal{X}^e \; \forall \, t$. Furthermore, if the motion results from straight nominal dynamics $\boldsymbol{v}$ as given in Definition 3.2.2, the final dynamics are stable and have a unique minimum at the attractor $\boldsymbol{\xi}^a \in \mathcal{X}^f$, i.e., $\{\boldsymbol{\xi} : \|\dot{\boldsymbol{\xi}}\| = 0, \nabla \dot{\boldsymbol{\xi}} > 0\} = \{\boldsymbol{\xi}^a\}$*

Proof. As the eigenvalues $\lambda^e(\boldsymbol{\xi})$ are equal in all tangent directions $\boldsymbol{e}_i$, we only analyze the two-dimensional case. Generalization to higher order system follow analogously as demonstrated by (L. Huber, Billard, and J.-J. Slotine, 2019).

Continuously Defined Vector field

Special consideration has to be given to the case where $\boldsymbol{r}(\boldsymbol{\xi}) \to \mathbf{0}$, as stated in (6.1). The final velocity is obtained as:

$$\begin{aligned}\lim_{\boldsymbol{r}(\boldsymbol{\xi})\to\mathbf{0}} \dot{\boldsymbol{\xi}} &= \lim_{\boldsymbol{r}(\boldsymbol{\xi})\to\mathbf{0}} \boldsymbol{E}(\boldsymbol{\xi})\boldsymbol{D}(\boldsymbol{\xi},\boldsymbol{v})\boldsymbol{E}(\boldsymbol{\xi})^T\boldsymbol{v} \\ &= \lim_{\boldsymbol{r}(\boldsymbol{\xi})\to\mathbf{0}} \boldsymbol{E}(\boldsymbol{\xi})\mathrm{diag}\left(\lambda^r(\boldsymbol{\xi},\boldsymbol{v}), \lambda^e(\boldsymbol{\xi})\right)\boldsymbol{E}(\boldsymbol{\xi})^T\boldsymbol{v} \\ &\approx \boldsymbol{E}(\boldsymbol{\xi})\mathrm{diag}\left((1+\sin(0)), \cos(0)\right)\boldsymbol{E}(\boldsymbol{\xi})^T\boldsymbol{v} \\ &= \boldsymbol{E}(\boldsymbol{\xi})\,\boldsymbol{I}\,\boldsymbol{E}(\boldsymbol{\xi})^T\boldsymbol{v} = \boldsymbol{v}\end{aligned} \tag{6.9}$$

Hence, the identity matrix continuously extends the modulation at $\boldsymbol{r}(\boldsymbol{\xi}) -> \mathbf{0}$.

Absence of Local Minima in Free Space

The modulation (6.1) does not introduce any local minima, if the modulation matrix and, hence, its components $\boldsymbol{E}(\boldsymbol{\xi})$ and $\boldsymbol{D}(\boldsymbol{\xi})$ have full rank, see Section 6.1. Let us first analyse the normal direction $\boldsymbol{n}(\boldsymbol{\xi})$ and reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ which are used to construct the decomposition matrix. We have

$$\bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\hat{\boldsymbol{n}}(\boldsymbol{\xi}) = \bar{\boldsymbol{r}}(\boldsymbol{\xi})^T(c\bar{\boldsymbol{r}}(\boldsymbol{\xi}) + \boldsymbol{n}^\Delta(\boldsymbol{\xi})) = c + \bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\boldsymbol{n}^\Delta(\boldsymbol{\xi}) \tag{6.10}$$

where $\bar{\boldsymbol{r}}(\boldsymbol{\xi}) = \boldsymbol{r}(\boldsymbol{\xi})/\|\boldsymbol{r}(\boldsymbol{\xi})\|$ denotes the normalized reference direction. Further, from the star-shape constraint in (3.16), we have $\|\boldsymbol{n}^\Delta(\boldsymbol{\xi})\| < \sqrt{2}$. Consider the case where $\left\langle \bar{\boldsymbol{r}}(\boldsymbol{\xi}), \boldsymbol{n}^\Delta(\boldsymbol{\xi})\right\rangle / \|\boldsymbol{n}^\Delta(\boldsymbol{\xi})\| > -\sqrt{2}/2$:

$$\bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\hat{\boldsymbol{n}}(\boldsymbol{\xi}) = 1 + \bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\boldsymbol{n}^\Delta(\boldsymbol{\xi}) \geq 1 - \|\boldsymbol{n}^\Delta(\boldsymbol{\xi})\|\sqrt{2}/2 > 0 \tag{6.11}$$

For the case of $\left\langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{n}^\Delta(\boldsymbol{\xi})\right\rangle / \|\boldsymbol{n}^\Delta(\boldsymbol{\xi})\| \leq -\sqrt{2}/2$ we have:

$$\begin{aligned}\bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\hat{\boldsymbol{n}}(\boldsymbol{\xi}) &= -\sqrt{2}\bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\boldsymbol{n}^\Delta + \bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\boldsymbol{n}^\Delta(\boldsymbol{\xi}) \\ &= (\sqrt{2}-1)\left(-\bar{\boldsymbol{r}}(\boldsymbol{\xi})^T\boldsymbol{n}^\Delta(\boldsymbol{\xi})\right) \\ &\geq (\sqrt{2}-1)\|\boldsymbol{n}^\Delta(\boldsymbol{\xi})\|\sqrt{2}/2 > 0\end{aligned} \tag{6.12}$$

It follows that the decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ has full rank.
Let us ensure that the eigenvalues of the stretching matrix $\boldsymbol{D}(\boldsymbol{\xi})$ are strictly positive. In tangent direction, from (6.8), we have $\lambda^e(\boldsymbol{\xi}) \geq 1$. Conversely, the eigenvalue in reference direction is limited as:

$$\lambda^r(\boldsymbol{\xi})1 - \|\boldsymbol{r}(\boldsymbol{\xi})\|^\rho = 1 - \underbrace{\frac{1}{\Gamma^{\min}}}_{<1,\ \boldsymbol{\xi}\in\mathcal{X}^e,\ (3.7)} \underbrace{\left\|\sum_{o=1}^{N^{\mathrm{obs}}} w_o(\boldsymbol{\xi})\boldsymbol{r}_o(\boldsymbol{\xi})\right\|}_{\sum_o w_o=1,\ \|\boldsymbol{r}_o\|=1\ \forall\, o} > 0 \tag{6.13}$$

Thus, $\boldsymbol{D}(\boldsymbol{\xi})$ has full rank in free space. Since both matrices have full rank, it follows that the modulation has a non-trivial result as long as the input velocity is nonzero.

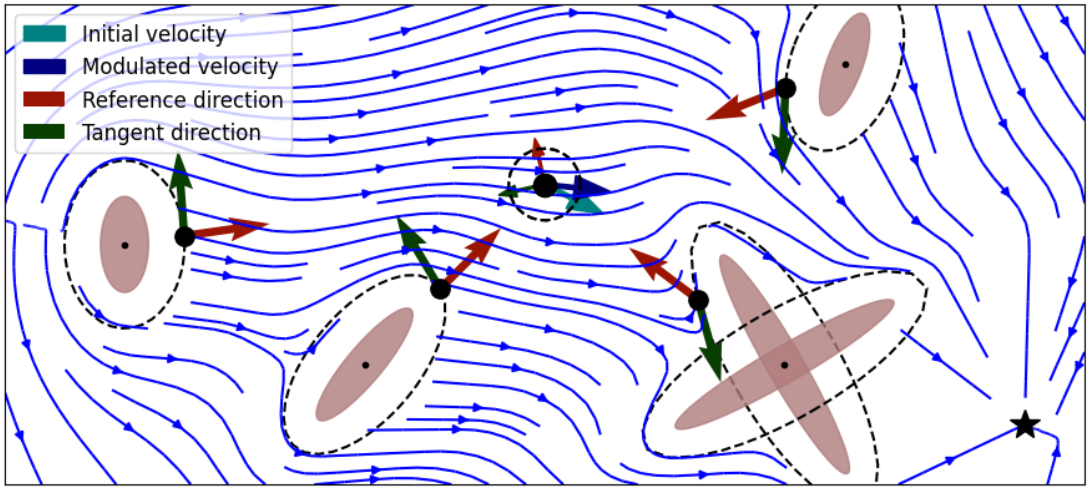


Figure 6.1: A single *virtual* obstacle is constructed with corresponding reference and tangent direction. This ensures the absence of minima in *star worlds* around analytical obstacles (brown).

Saddle Point Only on the Obstacles' Surfaces

The avoidance of multiple obstacles is defined as the modulation with respect to a single *virtual* obstacle with boundary function $\hat{\Gamma}(\boldsymbol{\xi})$. As we approach an obstacle o, we have $\hat{\Gamma}(\boldsymbol{\xi}) \to \Gamma_o(\boldsymbol{\xi})$ (see Sec. 6.2.1). Using the proofs from L. Huber, Billard, and J.-J. Slotine, 2019, we can conclude that: (1) there is no minimum on the surface of the obstacle o; hence, there are no minima for the virtual obstacle; and (2) there is impenetrability of the virtual obstacle $\hat{o}$ and, hence, any obstacle o as we approach its surface.

Finally, any trajectory starting in free space $\mathscr{X}^e$ will remain in free space. □

The proposed approach allows collision avoidance in a star-shaped world similar to Chapter 5 , but only executes a single modulation, resulting in lower computational cost (see Fig. 6.1).

6.3 Sample-Based Obstacle Avoidance

In many scenarios, the analytical obstacle description is not known at runtime, but the obstacles are perceived as a large number of data points, $\boldsymbol{\xi}_p \in \mathbb{R}^d$, $p = 1, ..., N^{\text{pnt}}$, where $N^{\text{pnt}} \in \mathbb{N}_{>0}$ is the total number of points.[1] We extend the notion of virtual obstacle to applying to many data points. Furthermore, throughout this section, we assume a circular d-dimensional robot with radius $R \in \mathbb{R}_{>0}$ and a sampling angle of $\delta \ll 1$, see Fig. 6.2.

[1] In this chapter, we focus on Lidar data, but this could be a point-cloud model.

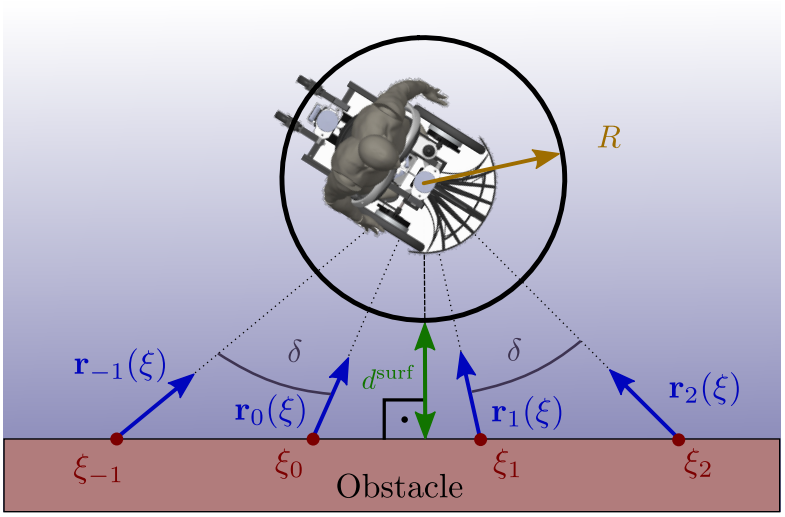


Figure 6.2: Mobile robot in front of a flat obstacle.

6.3.1 Reference Summing for Sampled Data

Let us define the weighted reference direction

$$\boldsymbol{r}(\boldsymbol{\xi}) = \sum_{o=1}^{N^{\text{pnt}}} w_p(\boldsymbol{\xi}) \boldsymbol{r}_p(\boldsymbol{\xi}) \quad \text{with} \quad \boldsymbol{r}_p(\boldsymbol{\xi}) = \frac{\boldsymbol{\xi}_p - \boldsymbol{\xi}}{\|\boldsymbol{\xi}_p - \boldsymbol{\xi}\|} \tag{6.14}$$

where the weights $w_p(\boldsymbol{\xi})$ are according to (6.3), but use the distance measure

$$D_p(\boldsymbol{\xi}) = \|\boldsymbol{\xi}_p - \boldsymbol{\xi}\| - R \qquad \forall\ p = 1..N^{\text{pnt}} \tag{6.15}$$

and limit the weighted sum

$$w^{\text{sum}} = \min\left(\sum_{p=1}^{N^{\text{pnt}}} \hat{w}_p(\boldsymbol{\xi}), w^{\text{norm}} \right) \tag{6.16}$$

where $w^{\text{norm}} > 0$ is the maximum distance weight; for more information, see Theorem 6.2.1.

6.3.2 Decomposition Matrix

Each data point be seen as an infinitesimally small circle. Hence, normal and reference directions are the same. The decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ is constructed to be orthonormal, i.e., the tangents $\boldsymbol{e}_{(\cdot)}(\boldsymbol{\xi})$ are orthonormal to the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$.
Alternatively, clustering the sample points can obtain an improved surface estimate. DBSCAN (Pedregosa et al., 2011) is a suitable clustering method, as the number of clusters is predicted by the algorithm and the choice of $\epsilon = 2R^{\text{robo}}$ for the neighborhood distance ensures the passage between two clusters. The surface point of each cluster is used to evaluate the normal direction

$\boldsymbol{n}(\boldsymbol{\xi})$, and the geometric center for the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$. Similarly to (6.6), it is also ensured that the decomposition matrix $\boldsymbol{E}(\boldsymbol{\xi})$ is invertible.

6.3.3 Eigenvalue Matrix

Eigenvalue Design Constraints

The magnitude of the reference direction $\|\boldsymbol{r}(\boldsymbol{\xi})\|$ reaches zero far away from an obstacle and approaches infinity when on the surface.
Accordingly, the modulated velocity $\dot{\boldsymbol{\xi}}$ needs to be tangent to the surface or directed away from it when approaching the obstacle to ensure collision avoidance. Conversely, the controller should have no effect when far away from obstacles.
These two constraints can be summarized with respect to the eigenvalues in tangent $\lambda^e(\boldsymbol{\xi})$ and reference direction $\lambda^r(\boldsymbol{\xi})$ from (6.7) as follows

$$\begin{aligned} \|\boldsymbol{r}(\boldsymbol{\xi})\| = 0 \quad &\Rightarrow \quad \lambda^r(\boldsymbol{\xi}) = \lambda^e(\boldsymbol{\xi}) = 1 \\ \|\boldsymbol{r}(\boldsymbol{\xi})\| = 1 \quad &\Rightarrow \quad \lambda^r(\boldsymbol{\xi}) / \lambda^e(\boldsymbol{\xi}) = 0 \\ \|\boldsymbol{r}(\boldsymbol{\xi})\| \to \infty \quad &\Rightarrow \quad |\lambda^e(\boldsymbol{\xi})| / \lambda^r(\boldsymbol{\xi}) = 0 \end{aligned} \tag{6.17}$$

Additionally, $\lambda^e(\boldsymbol{\xi}) \in \mathbb{R}_{>0}$ as the tangent direction should not be reflected.

Eigenvalues Based on Reference Direction

Based on the previous boundary conditions, we choose the eigenvalue in reference direction $\lambda^r(\boldsymbol{\xi}) \in [1, -1[$ as:

$$\lambda^r(\boldsymbol{\xi}) = \begin{cases} \cos(\frac{\pi}{2}\|\boldsymbol{r}(\boldsymbol{\xi})\|) & \text{if } \|\boldsymbol{r}(\boldsymbol{\xi})\| < 2 \\ -1 & \text{otherwise} \end{cases} \tag{6.18}$$

For close obstacles and, hence, large averaged reference directions, the eigenvalue is inverted:

$$\lambda^r(\boldsymbol{\xi}) \leftarrow (-1)\lambda^r(\boldsymbol{\xi}) \quad \text{if } \langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{v} \rangle < 0,\ \|\boldsymbol{r}(\boldsymbol{\xi})\| > 1 \tag{6.19}$$

Additionally, the eigenvalues in tangent direction $\lambda^r(\boldsymbol{\xi}) \in [1, 2]$ are given by

$$\lambda^e(\boldsymbol{\xi}) = \begin{cases} 1 + \sin\left(\frac{\pi}{2}\|\boldsymbol{r}(\boldsymbol{\xi})\|\right) & \text{if } \|\boldsymbol{r}(\boldsymbol{\xi})\| < 1 \\ 2\sin\left(\frac{\pi}{2\|\boldsymbol{r}(\boldsymbol{\xi})\|}\right) & \text{otherwise} \end{cases} \tag{6.20}$$

where π is the circle constant.
Note that the eigenvalues are C^1 smooth, because of $\frac{d}{dx}\sin(x)|_{x=\pi/2} = 0$ and $\frac{d}{dx}\cos(x)|_{x=\pi} = 0$. This enables collision-free movement around single (Fig. 6.3) or multiple obstacles (Fig. 6.4) based on point samples only.

Theorem 6.3.1. *Consider an agent with radius R and a nominal velocity* $\{\boldsymbol{v} \in \mathbb{R}^d : \boldsymbol{v} \neq \boldsymbol{0}\}$. *Let us assume the existence of* N^{obs} *obstacles with corresponding boundary points* $\mathcal{X}^b$ *and exterior points* $\mathcal{X}^e$, *i.e. the free space, as given in Eq. (3.7), from which the sampled surface points* $\boldsymbol{\xi}_p \in \mathbb{R}^d$, $p \in 1, ...N^{\text{pnt}}$ *are known at each time step* t.[2]
Any trajectory $\{\boldsymbol{\xi}\}_t$ *that starts in free space* $\mathcal{X}^e$, *and evolves according to Eq. (3.13), (6.18) and (6.20), will stay in free space, i.e.,* $\{\boldsymbol{\xi}\}_t \in \mathcal{X}^e$, *and will not stop outside of the gap distance, i.e.,* $\|\dot{\boldsymbol{\xi}}\| > 0 \ \forall \boldsymbol{\xi} \in \mathcal{X}^g$.

Proof. Analogously to the proof of Theorem 6.2.1, the two-dimensional analysis generalizes to higher dimensions.

Absence of Local Minima in Free Space

The modulated DS from Eq. (3.13) has the same equilibrium points as the original system, if $\det(\boldsymbol{M}(\boldsymbol{\xi})) \neq 0$; see (L. Huber, Billard, and J.-J. Slotine, 2019) for further information. Since the basis matrix $\boldsymbol{E}(\boldsymbol{\xi})$ is orthonormal, the modulation matrix is only *rank-deficient* if any of the eigenvalues are 0. From Eq. (6.18), we know that the only trivial eigenvalues are at $\lambda^e(\boldsymbol{\xi})|_{\|\boldsymbol{r}(\boldsymbol{\xi})\| \to \infty} = 0$ and $\lambda^r(\boldsymbol{\xi})|_{\|\boldsymbol{r}(\boldsymbol{\xi})\|=1} = 0$.

It is, hence, sufficient to limit the magnitude of the reference direction to $\|\boldsymbol{r}(\boldsymbol{\xi})\| < 1$. Let us analyze the *maximum repulsive environment* as drawn in Fig. 6.12a, i.e., where the distance to the surface D_p in one half-plane is D^{gap}, and infinite in the other half-plane:

$$D_p = \begin{cases} D^{\text{gap}} & \text{if } \langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{\xi}_p \rangle < 0 \\ \infty & \text{otherwise} \end{cases} \qquad p = 1..N^{\text{pnt}} \tag{6.21}$$

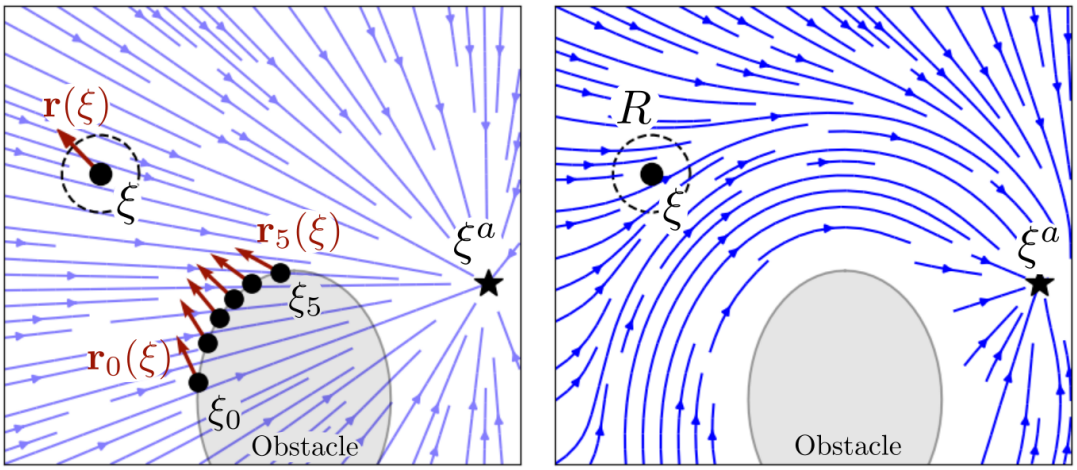
Hence, the magnitude of the reference direction is evaluated as:

$$\begin{aligned} \|\boldsymbol{r}(\boldsymbol{\xi})\| &\leq \sum_{i=0}^{N/2} \frac{D^{\text{scal}}}{D^{\text{gap}}} w^{\text{sum}} \sin(i\delta) \approx w^{\text{sum}} \frac{D^{\text{scal}}}{D^{\text{gap}}} \int_0^{\pi/\delta} \sin(\phi\delta)\, d\phi \\ &= w^{\text{sum}} \frac{D^{\text{scal}}}{D^{\text{gap}}} \frac{2}{\delta} \leq w^{\text{norm}} \frac{D^{\text{scal}}}{D^{\text{gap}}} \frac{2}{\delta} \end{aligned} \tag{6.22}$$

The approximation of the finite sum as an integral holds for a small sampling angle δ and many sample points, as is the case for most scanners.

By choosing $w^{\text{norm}} = (D^{\text{gap}}\delta)/(2D^{\text{scal}})$, the modulated DS does not vanish further than D^{gap} away from any obstacle.

[2] The sensor update rate $f = 1/(t_{i+1} - t_i)$ is expected to be larger than the relative motion $\Delta\boldsymbol{v}$ and the inverse of the distance, i.e., $f \gg \|\Delta\boldsymbol{v}\| / \|\boldsymbol{\xi} - \boldsymbol{\xi}_p\|$.



(a) Initial vector field

(b) Modulated vector field

Figure 6.3: At each position, the algorithm is only aware of the sample points $\boldsymbol{\xi}_p$, $p = 1..N^{\text{pnt}}$ and not the actual shape of the obstacle in grey (a). The summed reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ is used to guide the nominal velocity around the obstacle in grey (b).

Impenetrability

Let us observe the modulated velocity $\dot{\boldsymbol{\xi}}$ as we approach the surface, i.e., $\exists p : D_p \to 0$ (see Fig. 6.2), from (6.14) follows that $\|\boldsymbol{r}(\boldsymbol{\xi})\| \to \infty$ and hence $\lambda^e(\boldsymbol{\xi}) = 2\sin(\pi/(2\|\boldsymbol{r}(\boldsymbol{\xi})\|)) = 0$. With this, we can rewrite the modulated velocity as

$$\begin{aligned}\dot{\boldsymbol{\xi}} &= \boldsymbol{E}(\boldsymbol{\xi})\operatorname{diag}\left(\lambda^r(\boldsymbol{\xi}),\, \lambda^e(\boldsymbol{\xi})\right)\boldsymbol{E}(\boldsymbol{\xi})^T\boldsymbol{v} \\ &= [\bar{\boldsymbol{r}}(\boldsymbol{\xi})\ \boldsymbol{e}(\boldsymbol{\xi})]\operatorname{diag}\left(\lambda^r(\boldsymbol{\xi}),\, 0\right)[\bar{\boldsymbol{r}}(\boldsymbol{\xi})\ \boldsymbol{e}(\boldsymbol{\xi})]^T\boldsymbol{v} \\ &= \lambda^r(\boldsymbol{\xi})\,[\bar{\boldsymbol{r}}(\boldsymbol{\xi})\ \boldsymbol{0}]\operatorname{diag}(1,\, 0)\,[\bar{\boldsymbol{r}}(\boldsymbol{\xi})\ \boldsymbol{0}]^T\boldsymbol{v} \\ &= \lambda^r(\boldsymbol{\xi})\,\langle\bar{\boldsymbol{r}}(\boldsymbol{\xi}),\, \boldsymbol{v}\rangle\,\bar{\boldsymbol{r}}(\boldsymbol{\xi})\end{aligned} \tag{6.23}$$

Furthermore, from (6.18), we have

$$\lambda^r(\boldsymbol{\xi})\begin{cases} > 0 & \text{if } \langle\boldsymbol{r}(\boldsymbol{\xi}),\, \boldsymbol{v}\rangle > 0 \\ \leq 0 & \text{otherwise}\end{cases} \tag{6.24}$$

Consequently, the Neuman boundary condition holds true:

$$\begin{aligned}\left\langle\boldsymbol{n}(\boldsymbol{\xi}),\, \dot{\boldsymbol{\xi}}\right\rangle &= \boldsymbol{n}(\boldsymbol{\xi})^T\left(\lambda^r\,\langle\bar{\boldsymbol{r}}(\boldsymbol{\xi}),\, \boldsymbol{v}\rangle\,\bar{\boldsymbol{r}}(\boldsymbol{\xi})\right) \\ &\geq \underbrace{\arccos(\delta/2)}_{\approx 1,\ \text{since}\ \delta \ll 1}\ \underbrace{\lambda^r\,\langle\bar{\boldsymbol{r}}(\boldsymbol{\xi}),\, \boldsymbol{v}\rangle}_{\geq 0,\ \text{from (6.24)}} \geq 0\end{aligned} \tag{6.25}$$

As a result, we have a smooth vector field with no local minima away from the obstacles and proven impenetrability. □

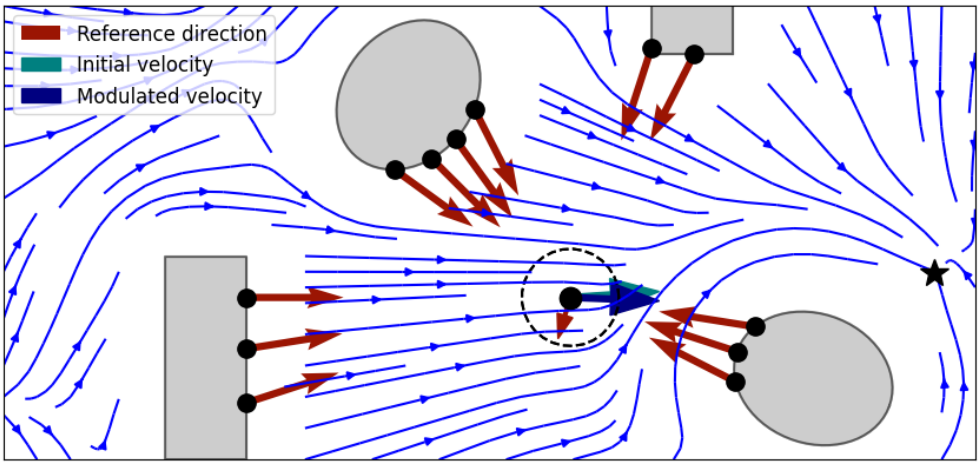


Figure 6.4: The algorithm does not need to estimate the number or shape of each obstacle. Calculating the weighted reference direction is sufficient to navigate safely in cluttered environments.

Tail Negligence

The modulation-based obstacle avoidance is active even when the nominal dynamics $\boldsymbol{v}$ are already moving away from an obstacle. This can be an undesired effect since, in this case, the unconstrained velocity already ensures collision avoidance. This *tail* effect can be reduced by modifying the wake eigenvalues while still ensuring a smooth vector field by adapting the eigenvalues as follows:

$$\begin{aligned} \tilde{\lambda}^e(\boldsymbol{\xi}) &= p\,s + (1 - p\,s)\lambda^e(\boldsymbol{\xi}) \\ \tilde{\lambda}^r(\boldsymbol{\xi}) &= \operatorname{sgn}(s)\,p\,\tilde{\lambda}^e(\boldsymbol{\xi}) + (1 - \operatorname{sgn}(s)\,p)\lambda^r(\boldsymbol{\xi}) \end{aligned} \tag{6.26}$$

with the weights given as

$$p = \min\left(1, \frac{1}{\|\boldsymbol{r}(\boldsymbol{\xi})\|}\right) \qquad s = \max\left(0, \frac{\langle \boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{v}\rangle}{\|\boldsymbol{r}(\boldsymbol{\xi})\|\,\|\boldsymbol{v}\|}\right)^c \tag{6.27}$$

where $c \in \mathbb{R}_{>0}$ is the power weight; we choose $c = 0.2$.

Decreasing Tail Weight

The importance of obstacles in the wake of the nominal velocity $\boldsymbol{v}$ can be further decreased by modifying the weight. For this, we restate the weight from Eq. (6.3) as:

$$\hat{w}_o \leftarrow \hat{w}_o \left(\frac{\hat{w}_o}{\sum_i \hat{w}_o}\right)^{1/k} \text{ with } k = 1 - \frac{\langle \boldsymbol{v}, \boldsymbol{r}_o(\boldsymbol{\xi})\rangle}{\|\boldsymbol{v}\|\,\|\boldsymbol{r}_o(\boldsymbol{\xi})\|} \quad \forall\, o = 1..N^{\text{obs}} \tag{6.28}$$

6.4 Disparate Obstacle Descriptions

Detection algorithms can fail to identify certain obstacles. Hence, we are left with a mix of analytic obstacles and sampled sensor data. Furthermore, the analytic reconstruction of the obstacles often comes with a significant delay. This results in the asynchronous reception of information on obstacles, with fast sample-based information and slow analytical model acquisition. We address this by proposing an approach to fuse analytic obstacle descriptions (see Sec. 6.2) with sample-based sensor data (see Sec. 6.3).

The first step is to remove sampled obstacle data which has been identified as an analytic obstacle. Hence, the laser scan weights of a point $p = 1..N^{\text{pnt}}$ defined in (6.16) are set to zero if they belong to an obstacle o:

$$\hat{w}_p(\boldsymbol{\xi}) \leftarrow 0 \qquad \text{if } \exists o \in \{1, .., N^{\text{obs}}\} : \Gamma_o(\boldsymbol{\xi}_p) \leq 1 \tag{6.29}$$

6.4.1 Fusion Weights of Sampled and Analytic Obstacles

The weight is used to evaluate the reference direction of the mixed environment:

$$\boldsymbol{r}(\boldsymbol{\xi}) = w^p(\boldsymbol{\xi})\boldsymbol{r}^p(\boldsymbol{\xi}) + w^o(\boldsymbol{\xi})\boldsymbol{r}^o(\boldsymbol{\xi}) \tag{6.30}$$

where $\boldsymbol{r}^o(\boldsymbol{\xi})$ is the averaged reference direction from all the obstacles, as given in (6.2), and $\boldsymbol{r}^p(\boldsymbol{\xi})$ the averaged reference direction from the sampled data points, as given in (6.14). The magnitude of the reference direction of the sampled and analytic obstacles gives us information about the proximity to obstacles. Hence, the reference magnitude can be used to calculate the importance weight of the sampled data $w^p \in [0,1]$ and analytic obstacles $w^o \in [0,1]$ as:

$$\begin{bmatrix} w^p \\ w^o \end{bmatrix} = \frac{1}{p+o} \begin{bmatrix} p \\ o \end{bmatrix} \quad \text{with} \begin{cases} p = 1/\left(1 - \|\bar{\boldsymbol{r}}^p(\boldsymbol{\xi})\|\right) - 1 \\ o = 1/\left(1 - \|\bar{\boldsymbol{r}}^o(\boldsymbol{\xi})\|\right) - 1 \end{cases} \tag{6.31}$$

As a result, navigation is possible in disparate environments, as shown in Fig. 6.5.

The analytical description has the additional advantage that it often provides information about the velocity and shape of the obstacles. This can be used to ensure collision avoidance in dynamic environments. We propose including the velocity similar to (L. Huber, Billard, and J.-J. Slotine, 2019) for a fused scenario, as follows:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{v})\left(\boldsymbol{v} - \dot{\tilde{\boldsymbol{\xi}}}^{\text{d,tot}}\right) + \dot{\tilde{\boldsymbol{\xi}}}^{d,\text{tot}} \qquad \dot{\tilde{\boldsymbol{\xi}}}^{d,\text{tot}} = w^o \dot{\boldsymbol{\xi}}^{\text{tot}} \tag{6.32}$$

where $\dot{\boldsymbol{\xi}}^{\text{tot}} \in \mathbb{R}^d$ is the mean velocity of all analytical obstacles.

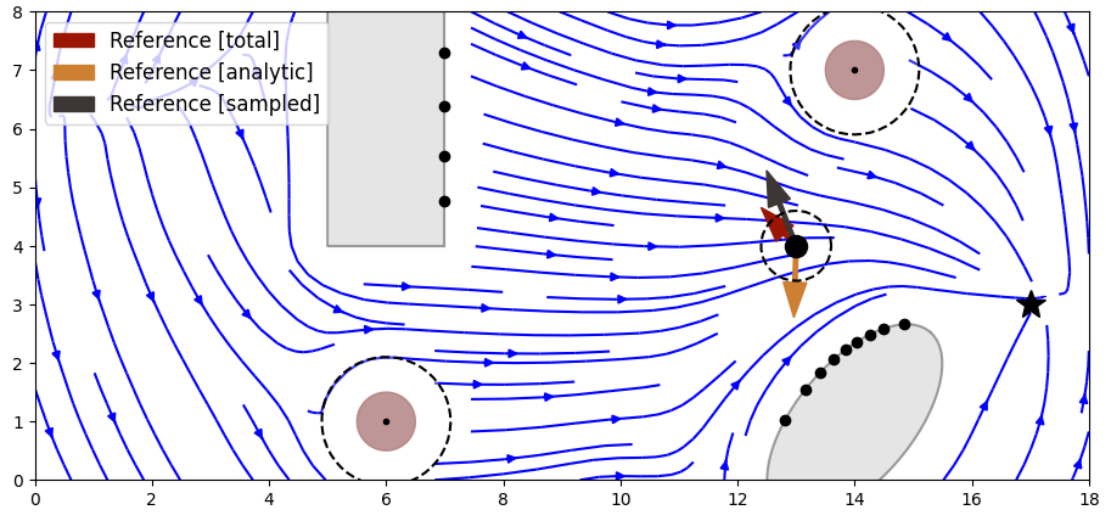


Figure 6.5: Analytical obstacle descriptions (brown) and sampled obstacle data (black dots on the gray obstacles) are fused by the algorithm. The weighting uses the corresponding reference directions.

6.4.2 Asynchronous Evaluation

The disparate data is combined using a controller with three inputs: the velocity command from the human operator $\boldsymbol{v}$, the laser scan reading $\boldsymbol{\xi}_p,\ p = 1..N^{\text{pnt}}$, and the estimated position of the robot $\boldsymbol{\xi}$ (see Fig. 6.6).
The two sensor loops run at different frequencies, with the sample-based evaluator running at a higher update rate than the analytic evaluator. The latter is delayed due to the processing time required to generate analytic obstacle descriptions. However, the analytic descriptions enable improved behavior due to the information about the obstacle's shape and velocity.

6.4.3 Uniform Importance Scaling

An obstacle described by sampled data points or an analytic function should have the same effect on the modulation. Adequate scaling is necessary since a single obstacle often corresponds to multiple data points.
Let us consider an obstacle that can be approximated by radius R^{obs} at a distance D. The number of sampling data points N^{pnt} can be approximated as:[3]

$$N^{\text{pnt}} \sim \left(R^{\text{obs}}/\tan(\delta)D\right)^{d-1} \sim \left(R^{\text{obs}}/\delta D\right)^{d-1} \tag{6.33}$$

It follows that the scaling weight of the sampled data is set to be proportional to the sampling

[3] If the obstacle sampling is volumetric, e.g., when obtaining a collision model of the robot in higher dimensions, the power value of the right side is d, i.e., $N^{\text{pnt}} \sim (\cdot)^d$.

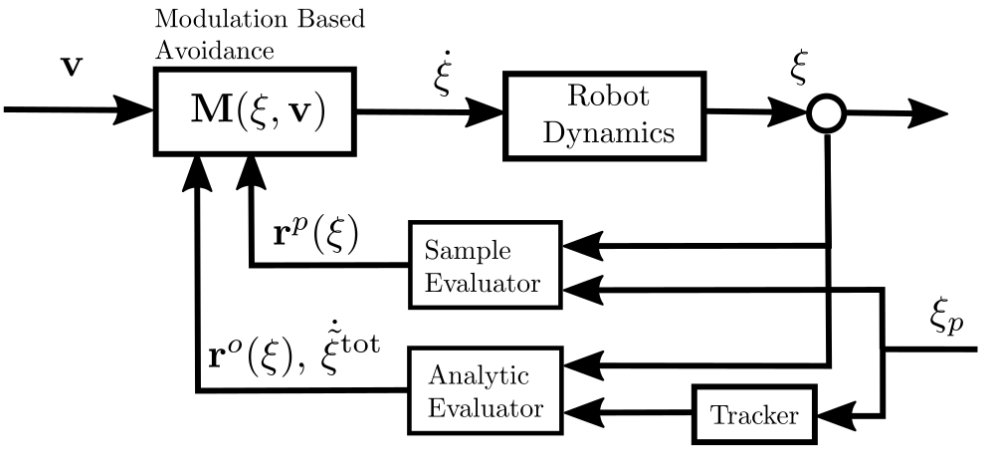


Figure 6.6: FOA uses the latest information from the sampled and analytic obstacle description. The reference directions are obtained at different rates and are combined asynchronously.

angle δ:

$$s^{\text{pnt}} \approx \delta^{d-1} \tag{6.34}$$

Moreover, the scaling weight of the analytic obstacle description is dependent on the size and distance.

$$s^{\text{obs}} \approx \left(R^{\text{obs}}/D\right)^{d-1} \tag{6.35}$$

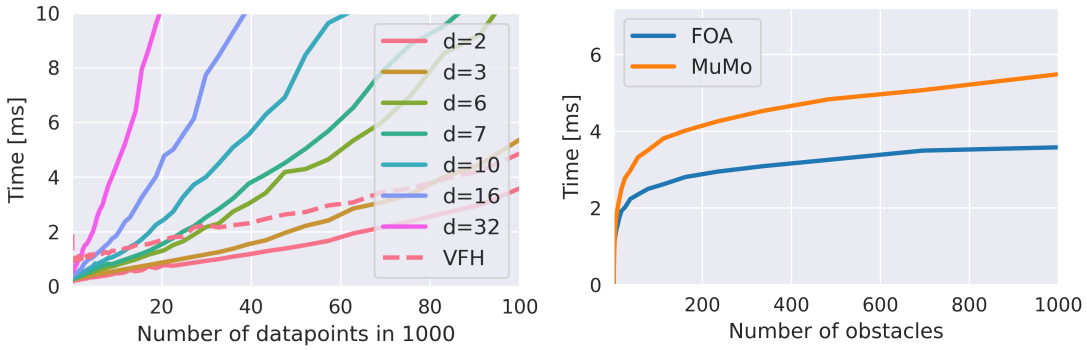
As a result, we set the distance scaling to $D^{\text{scal}} = 2\pi/\delta^{(d-1)}$ in mixed environments.

6.5 Experimental Validation

6.5.1 Computational Speed

FOA has a computational complexity of $\mathcal{O}(dN^{\text{obs}})$. Thus, it is lower than when compared to similar approaches that have a complexity of $\mathcal{O}(d^{2.4}N^{\text{obs}})$, see (L. Huber, Billard, and J.-J. Slotine, 2019). This was confirmed by the low computational time observed during the experiments for sampled data and analytic obstacle descriptions (see Fig. 6.7). Often the most important decrease of computation time of the FOA compared to the baseline is due to using sampled sensor data directly without the need for an analytic obstacle description, see the experiments in Section 6.5.3.

6.5.2 Convergence Analysis Using Disparate Sensor Data



(a) Sample data

(b) Analytic data

Figure 6.7: The computational complexity of the FOA grows linearly with the number of data points (a), but is lower than for VFH (Levine et al., 1999; Q. Li, W. Chen, and J. Wang, 2011) in two dimensions. Similarly, using analytic obstacles, the FOA outperforms the baseline with multiple modulations (MuMo), defined in Chapter 5.

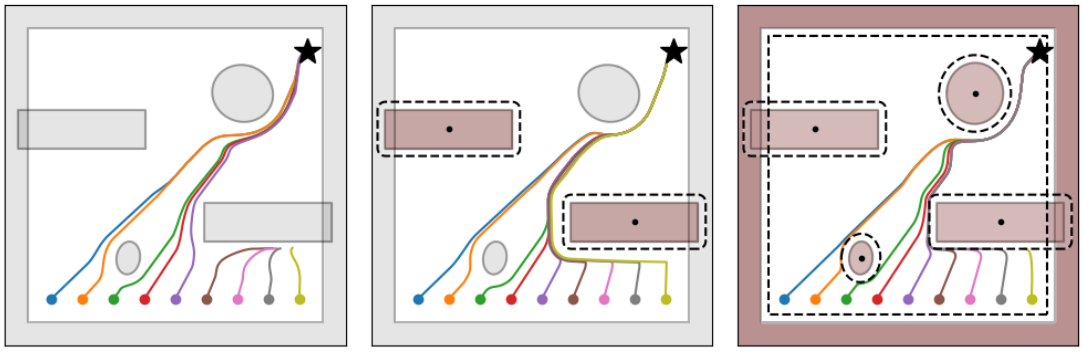
FOA and two baselines were applied to simulated, two-dimensional environments with four obstacles and a surrounding wall (Fig. 6.8).[4] The shape and pose of the elliptical obstacles were randomly obtained in the top-right and bottom-left corners, respectively. The positions of the two square obstacles and the wall are fixed. The sample-based description (obstacles in gray) is obtained by the algorithm from a simulated laser scan with a sampling angle 0.12 rad.
With increasing environment observations (from sampled to analytic), the convergence rate R increases (Tab. 7.1). While the Raw-FOA has an increased convergence rate compared to the VFH, the Full-FOA is outperformed by MuMo. However, in both cases, the evaluation time T of the FOA is lower: by a factor of 7 for the sampled environment, and a factor of 2 for the analytic environment. The convergence distance D is similar for all algorithms, but Raw-FOA has a lower mean velocity $\bar{v}$ compared to the other algorithms. Additionally, the FOAs have higher velocity variation Δv to the baselines.

6.5.3 Evaluation Using a Semi-Autonomous Wheelchair QOLO

The experimental validation was performed with the semi-autonomous, standing wheelchair QOLO. The nominal velocity $\boldsymbol{v}$ was obtained from the onboard operator. The wheelchair was non-holonomic with two rigid main wheels and two passive back wheels; its geometry is approximated with a circle of radius 0.45 m. The circle's center $\boldsymbol{\xi}$ denotes the front of the middle of the main wheels at a distance of $d = 6.25 \times 10^{-2}$ m. The linear l and angular a velocity command were obtained using the Jacobian $\boldsymbol{J}$:

$$\begin{bmatrix} l & a \end{bmatrix}^T = \boldsymbol{J}^{-1}\dot{\boldsymbol{\xi}} \qquad \text{with} \quad \boldsymbol{J}^Q = \operatorname{diag}(1 \;\; d) \tag{6.36}$$

[4] Source code on https://github.com/hubernikus/fast_obstacle_avoidance.git



(a) Sampled (b) Disparate (c) Analytic

Figure 6.8: FOA can access sampled information of the obstacles' surface in gray only (a), an analytic obstacle model (c), or a disparate mix of sampling and analytic description (b).

Table 6.1: In the three scenarios from Fig. 6.8, the FOAs are compared to the sample-based approach VFH (Levine et al., 1999; Q. Li, W. Chen, and J. Wang, 2011), and analytic-obstacles approach using multiple modulations (MuMo), see Chapter 5. The ratio of convergence R is with respect to the total of 100 runs. The computation time C in $1e^{-}4$ s is the average evaluation of one timestep. The distance traveled D in m is the total of all time steps, the mean velocity $\bar{v}$ in m / s is averaged from all time steps, and Δv in m / s is the variation in the velocity magnitude. o

	Sampled		Disparate	Analytic	
	Raw-FOA	VFH	Partial-FOA	Full-FOA	MuMo
R	62%	53%	76%	79%	85%
T	0.3 ± 0.0	2.2 ± 0.2	1.8 ± 0.2	0.9 ± 0.1	1.8 ± 0.3
D	18.9 ± 1.6	18.2 ± 1.6	18.4 ± 1.5	18.5 ± 1.5	18.3 ± 1.5
$\bar{v}$	13.8 ± 1.3	14.9 ± 0.0	14.9 ± 0.0	14.7 ± 1.2	14.9 ± 0.0
Δv	7.3 ± 2.3	5.5 ± 5.1	4.7 ± 5.2	6.0 ± 9.5	4.1 ± 1.8

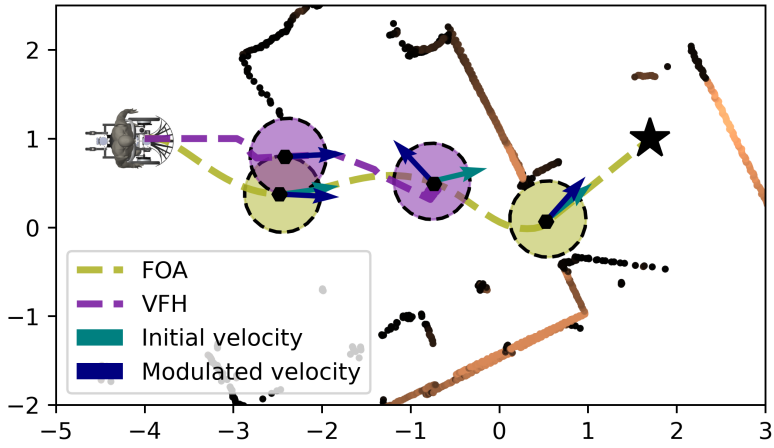
Passing Narrow Spaces Based on Sensor Data

The QOLO is equipped with two 3D *Velodyne* Lidars, which are both interpreted as a laser scan by extracting the horizontal 2D row of the Lidar. During the recording, the sensor was placed approximately 30 cm above ground. The resulting laser scans have an angle of view of $\pm 0.75\,\pi$ rad and an angle increment of $\delta = 7 \times 10^{-3}$ rad, resulting in approximately 650 data points each. While both FOA and the baseline VFH use the (raw) sensor data, VFH often does not outputs smooth control command in cluttered environments. Conversely, FOA controls the robot with a smooth velocity command, even when passing through narrow doorways (Fig. 6.9).

Control Update Rate in Fast-Changing Environments

FOA was implemented in *Python* and evaluated on the onboard computer of QOLO (*Intel(R) Atom(TM) Processor E3950 @ 1.60GHz*), taking 0.4 ms to evaluate a scenario with over 1000

(a) Doorway



(b) Comparison of the algorithms passing the doorway

Figure 6.9: During the experiments with QOLO passing a narrow doorway (a), FOA allowed the safe navigation of the passage, while VFH was blocked in front of the door (b).

data points. Furthermore, the main control loop ran at 100 Hz, with Lidar information obtained at 20 Hz and the pedestrian tracker providing an update at a frequency of 1 Hz to 5 Hz (depending on the number of tracked persons in close proximity). The operator control input was obtained at a frequency of 20 Hz.

FOA was directly applied on the sensor data; hence, the control delay was approximately 1 ms (algorithm only). Whereas, MuMo relies on the detection of the person; hence, the delay is approximately 200 ms to 1000 ms (tracker interpretation and control loop).[5] Comparing FOA to MuMo, this is an effective speed up of data reception to control output of over 200.

In the experiment, where a person suddenly appeared in front of the robot at a distance of approximately 0.4 m, the algorithm is required to update the motion in a fraction of a second to ensure the continuous movement of the robot at a speed of approximately 0.5 m/s. While FOA has can ensure a collision-free trajectory, the evaluation time of MuMo is too long to avoid collisions (Fig. 6.10).

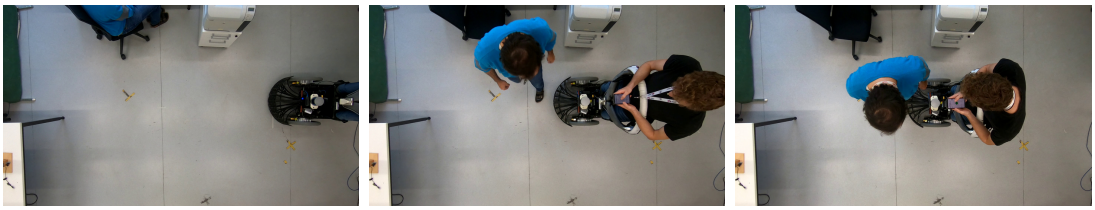
Navigation in Outdoor Crowds

The qualitative evaluation using the robot was conducted in the city center of Lausanne, Switzerland.[6] The chosen location is an intersection of six streets and is pedestrianized. This results in a large diversity in pedestrian speeds and directions of movement. The operator navigated with QOLO up and down the street along a transect with a distance of 15 meters (Fig. 6.11). During the experiments, the controller supported the operator while ensuring collision-free motion.

[5] Moreover, the tracker cannot guarantee detecting pedestrians, and often fails when they suddenly appear in close proximity.

[6] Approved by the EPFL ethics board and the city of Lausanne.

(a) Raw-FOA (b) Raw-FOA (c) Raw-FOA



(d) MuMo (e) MuMo (f) MuMo

Figure 6.10: The person was initially not detected by the algorithms as they were hidden behind the shelf, (a) and (d). However, as they moved abruptly in front of QOLO, see (b) and (e), the much-shorter reaction time of the pipeline using the Raw-FOA ensured collision avoidance (c), while with MuMo a collision occurred (f).

6.6 Discussion

This chapter presents a modulation-based approach for fast obstacle avoidance (FOA) based on dynamical systems. FOA excels in its speed and can be applied to sampled data, such as Lidar or laser scans combined with analytic obstacle descriptions. We proved the absence of local minima in free space, and an agent controlled by FOA only stopped when directly driving towards obstacles. Furthermore, FOA is scalable to higher dimensions. Experimental validation was performed in simulation, static sensor data, and the standing wheelchair QOLO. It was shown that the algorithm avoids collisions while navigating in static and dynamic environments. The evaluation with sampled sensor data showed that the algorithm could pass narrow doorways. Moreover, the algorithm had a short evaluation time onboard the robot.

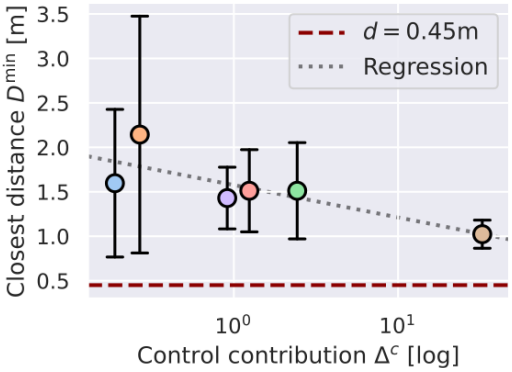
Small Obstacles

The discrete sampling of a laser scan might miss the closest point of an obstacle, i.e., an edge of an obstacle (see Fig. 6.12b). A small margin around the obstacle accounts for this:

$$R^{\text{marg}}/R^{\text{robo}} = \sin(\delta/2)/\tan(\phi^{\min}/2) + (1-\cos(\delta/2))$$

Note that the first summand comes from the obstacle's corner and the second summand is the contribution of the agent's curvature. A good choice for the minimum obstacle angles is $\phi^{\min} = 45°$.
Additionally, any obstacle with a robot-to-obstacle curvature-radius ratio of $R^{\text{obs}}/R^{\text{robo}} > \sin(\delta/2)/\cos(\phi^{\min}/2)$ is avoided safely, too (see Fig. 6.12b).

(a) QOLO in the city center



(b) Shared control analysis

Figure 6.11: During the evaluation with the QOLO robot in Lausanne (a), the control contribution ensured that the closest distance to any pedestrian stayed above the save margin of 0.45 m during six runs (b). The control contribution is the relative input of the controller, evaluated as $\Delta^c = \|\dot{\boldsymbol{\xi}} - \boldsymbol{v}\| / \|\boldsymbol{v}\|$. The closest distance $D^{\min}$ is the average distance of the ten closest Lidar measurements. The relation between the mean of the two measurements was observed to be $\Delta^c = \exp(10.0 - 6.3 D^{\min})$.

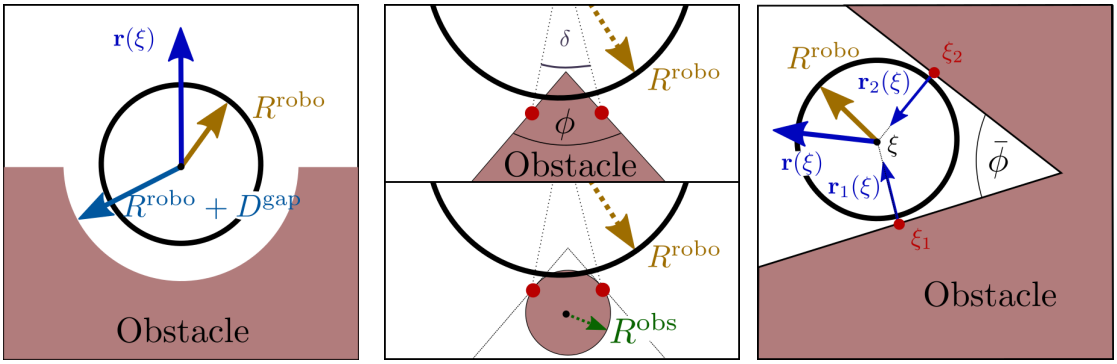
Concave Regions

When approaching a concave region, as in Fig. 6.12c, the algorithm is not aware that the resulting normal is obtained from multiple distinct walls. Nevertheless, under the assumption that the distance to both walls is close to zero and that we obtain approximately the same weight from both walls, the resulting reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ points away from the corner and has a large magnitude. It follows from (6.18) that the modulated DS will be pointing the same way as the reference direction, i.e. $\langle \dot{\boldsymbol{\xi}}, \boldsymbol{r}(\boldsymbol{\xi}) \rangle / (\|\dot{\boldsymbol{\xi}}\| \, \|\boldsymbol{r}(\boldsymbol{\xi})\|) \to 1$. Hence, the velocity is collision-free.

6.6.1 Future Work

Most of the theoretical developments in this chapter concern multiple-dimensional collision avoidance. While, the implementation was on a mobile agent in crowds for navigation in two dimensions, future implementations will focus on scalability to higher dimensions. Sampled data can be obtained using distance sensors for three-dimensional scenarios. Conversely, in a higher dimensional space, e.g., joint space, the collision data can be obtained in through sampling in simulation (Koptev, Figueroa, and Billard, 2021).

The algorithm was tested in a shared-controller setup. Future work will include a quantitative controller evaluation compared to existing algorithms in various crowd scenarios. The quantitative controller evaluation is intended to include several operators and identify how different driving styles influence the overall behavior. Finally, we believe that the algorithm's quantitative data will enable the design of an intelligent shared controller, which can learn and adapt its behavior based on the operator and the environment, similar to (Q. Li, W. Chen, and J. Wang, 2011).



(a) Repulsive gap (b) Small obstacles (c) Concave regions

Figure 6.12: Special environments for the sensor reading.

7 Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics

Publication Note

The material in this chapter is adapted from:
Huber, L., J.-J. Slotine, and A. Billard (2023). Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics". In: *IEEE Transactions on Robotics.*

Source Code:

- Algorithm: https://github.com/hubernikus/nonlinear_obstacle_avoidance
- Implementation on Robot:
  https://github.com/hubernikus/franka_obstacle_avoidance

Multimedia: Suplementary video can be found under:

https://youtu.be/Lkaa1_bNOX4

Controlling complex tasks in robotic systems, such as circular motion for cleaning or following curvy lines, can be dealt with using nonlinear vector fields. This Chapter introduces a novel approach called the rotational obstacle avoidance method (ROAM) for adapting the initial dynamics when obstacles partially occlude the workspace. ROAM presents a closed-form solution that effectively avoids star-shaped obstacles in spaces of arbitrary dimensions by rotating

the initial dynamics toward the tangent space. The algorithm enables navigation within obstacle hulls and can be customized to actively move away from surfaces while guaranteeing the presence of only a single saddle point on the boundary of each obstacle. We introduce a sequence of mappings to extend the approach for general nonlinear dynamics. Moreover, ROAM extends its capabilities to handle multi-obstacle environments and provides the ability to constrain dynamics within a safe tube. By utilizing weighted vector-tree summation, we successfully navigate around general concave obstacles represented as a tree-of-stars. Through experimental evaluation, ROAM demonstrates superior performance in minimizing occurrences of local minima and maintaining similarity to the initial dynamics, outperforming existing approaches in multi-obstacle simulations. Due to its simplicity, the proposed method is highly reactive and can be applied effectively in dynamic environments. This was demonstrated during the collision-free navigation of a 7-degree-of-freedom robot arm around dynamic obstacles.

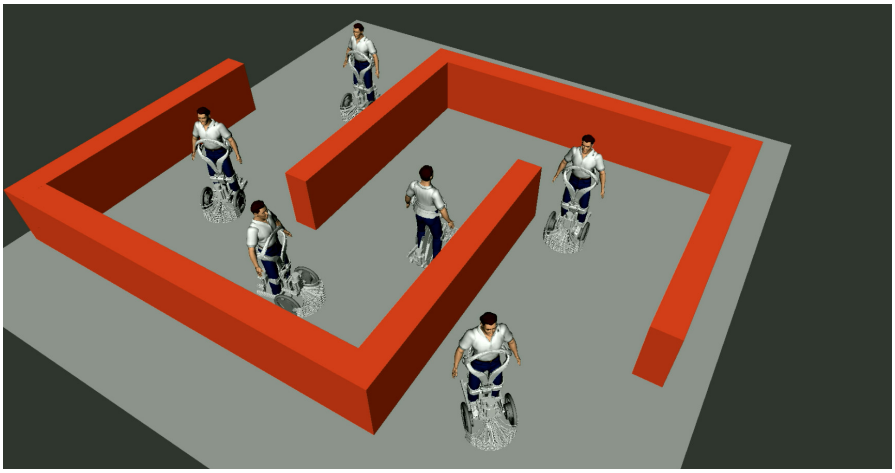
7.1 Introduction

Reactive motion plays a crucial role in numerous real-world robotics applications. When operating outside the controlled environments of factory floors, robots are exposed to unpredictable and dynamic surroundings, making precise estimation challenging. As a result, real-time adaptive controllers are essential to enable robots to adapt and reevaluate their actions in response to changing conditions.

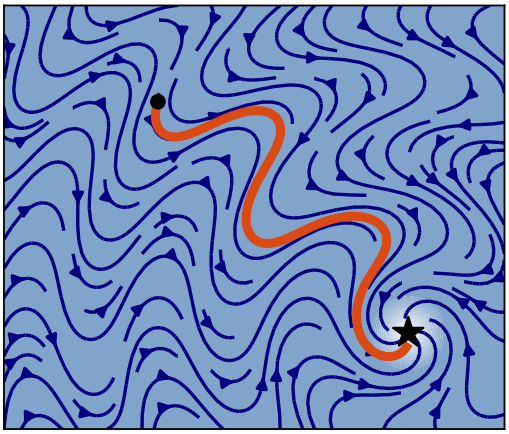
A primary constraint when navigating dynamic and cluttered environments is ensuring the safety of individuals moving around the robot. This requires the robot to constantly and rapidly replan its path to avoid collisions while maintaining its intended task and adhering to the originally intended movement dynamics. Furthermore, it is crucial to design a smooth system to protect physical hardware from potential damage caused by high accelerations. Additionally, collision avoidance needs to seamlessly integrate with reactive control techniques (Figure 7.1).

Control methods based on vector fields (Goncalves et al., 2010) and dynamical systems Chapter 3 have proven well-suited for addressing the challenges posed by such scenarios. Rather than precomputing a trajectory for the robot, these methods generate a (nonlinear) control field that is evaluated in real-time at the robot's position. This allows the robot to react instantaneously to disturbances and perceive environmental changes. In this work, we leverage the dynamical systems and vector fields framework to develop an adaptive obstacle avoidance approach capable of modifying nonlinear motion in dynamic and complex environments (Fig. 7.1).

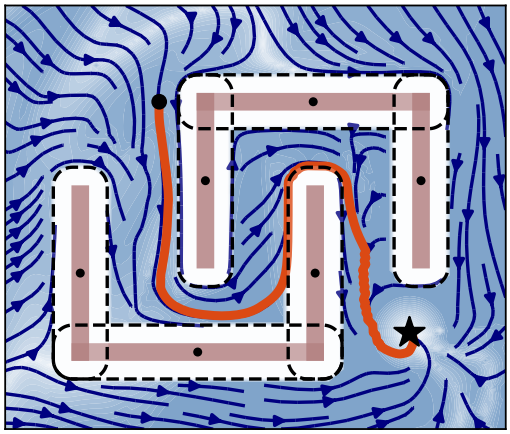
Dynamical systems represent the evolution of a system's state without considering its history, effectively capturing the system's dynamics as a vector field. In robotics, dynamical systems have been employed for learning complex dynamics (Figueroa and Billard, 2018) and enforcing stability guarantees through techniques such as Lyapunov Stability (J.-J. E. Slotine et al., n.d.) or Contraction Theory (L. Huber, Billard, and J.-J. Slotine, 2019). Force control in robotics often utilizes second-order dynamical systems (Salehian and Billard, 2018), while obstacle avoidance



(a) QOLO-robot navigating within small labyrinth



(b) Initial dynamics



(c) Rotated dynamics

Figure 7.1: An autonomous wheelchair is guided by obstacle avoidance to navigate in an environment of complex obstacles (c). The intensity of the shading in (a) and (b) indicates the magnitudes of the velocity.

typically relies on first-order systems that output desired velocities based on the current position (S. M. Khansari-Zadeh and Billard, 2012). Consequently, additional controllers are employed to ensure force- and torque-controlled robots follow the desired motion of the dynamical system (Kronander and Billard, 2015).

The desired dynamics based on the position can be analogously represented as a vector field (Goncalves et al., 2010). These nonlinear vector fields can be designed to ensure convergence and stability properties. Nonlinear vector fields designed for path following, known as "vector-field-guided path following," enable smooth convergence towards a desired path, reducing path-following errors (Kapitanyuk, Proskurnikov, and M. Cao, 2017).

In this chapter, we develop an adaptive obstacle avoidance framework capable of modifying

nonlinear motion in response to dynamic and complex environments. The proposed approach aims to navigate the control system safely while maintaining the integrity of the intended motion dynamics. The effectiveness of our approach is demonstrated through experiments.

7.1.1 Problem Statement

We establish the following requirements for our obstacle avoidance controller:

- **Collision free:** The flow must remain outside the obstacles at all times, i.e., $\{\boldsymbol{\xi}\}_t \in \mathcal{X}^e \; \forall \, t$ if $\{\boldsymbol{\xi}\}_0 \in \mathcal{X}^e$.
- **State dependent:** The dynamics are history-invariant and depend solely on the current state, i.e. $\dot{\boldsymbol{\xi}}_t = \boldsymbol{f}(\boldsymbol{\xi}_t, \boldsymbol{\xi}_{t-1}, .., \boldsymbol{\xi}_0) = \boldsymbol{f}(\boldsymbol{\xi}_t)$.
- **Local minima free:** As demonstrated in Yao, B. Lin, et al., 2022, each C^1-smooth vector field obstacle introduces at least one stationary point on the surface. However, it must be ensured that (1) this point is a saddle point and not a minimum and (2) there are no additional stationary points in space.
- **Limited and smooth dynamics;** The magnitude of the vector field is upper bounded, i..e., $\exists \boldsymbol{\nu}^{\max} = \text{const.} : \|\dot{\boldsymbol{\xi}}\| < \boldsymbol{\nu}^{\max}$. Additionally, the vector field is smooth, meaning that small displacements result in proportionally small velocities: $\lim_{\boldsymbol{\xi}_1 \to \boldsymbol{\xi}_2} \|\dot{\boldsymbol{\xi}}_2 - \dot{\boldsymbol{\xi}}_1\| = 0$.
- **General dimensions:** The obstacle avoidance algorithm is applicable in a space of dimensions $N \geq 2$.

This list does not contain any kinematic constraints. Hence, we assume that any dynamics can be assigned immediately (or within a negligibly short time). Such a system has to be fully actuated.

Furthermore, we assume that the environment and initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$, as described in (3.1), possess the following properties:

- **Star-shaped:** All obstacles are star-shaped as defined in Sec. 3.3.1, or are composed of obstacles (trees-of-stars) with a nonzero intersection region among the components of a tree, as discussed in Section 3.3.1.
- **Limited and smooth dynamics:** Analogously to the output, the magnitude of the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is required to be C^1-smooth and limited, i.e., $\|\boldsymbol{f}(\boldsymbol{\xi})\| < \nu^{\max}, \;\; \forall \boldsymbol{\xi} \in \mathbb{R}^N, \nu^{\max} \in \mathbb{R}_{>0}$.
- **Dynamics as a vector rotation:** The initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ can be evaluated as a local rotation or a sequence of rotations (Section 4.2) of globally straight dynamics with respect to a fixed point $\boldsymbol{\xi}^a$ (Definition 3.2.2).

7.1.2 Contributions

This Chapter significantly expands upon previously devloped DSM approach, enabling obstacle avoidance with nonlinear dynamical systems and for general concave obstacles. The chapter's technical contributions are outlined as follows:

- We introduce the Rotational Obstacle Avoidance Method (ROAM), which ensures local minima-free obstacle avoidance for nonlinear dynamics (Section 7.2).
- We present a diffeomorphic mapping that guarantees convergence while maintaining proximity to the general nonlinear dynamics while trying to preserve the initial flow (Section 7.3).
- We extend the ROAM formulation to handle obstacle avoidance of multiple dynamic obstacles and enclosed spaces, utilizing the concept of inverted obstacles as introduced in Chapter 5 (Section 7.4).
- Finally, we demonstrate the application of ROAM for avoiding general obstacles represented by trees-of-stars in the presence of nonlinear dynamics (Section 7.5).

To validate the properties of ROAM, we conducted the following evaluations:

- We performed a quantitative comparison of ROAM against two recent analytical obstacle avoidance methods through simulations, assessing convergence rate, similarity to unperturbed motion, and acceleration along the trajectories (Section 7.6).
- Furthermore, we conducted a qualitative evaluation of ROAM's application in controlling a 7DoF robot arm to avoid trees-of-stars in three dimensions (Section 7.6.4).

The proposed rotation obstacle avoidance method (ROAM) ensures collision avoidance in a local minima-free vector field in the presence of initial nonlinear dynamics. Moreover, the influence regions of the obstacle can overlap, and the method has been extended to inverted obstacles, too, as well as multiple obstacles with overlapping regions of influence, as can be seen in the comparison with similar algorithms in Table 7.1.

7.2 Obstacle Avoidance through Rotation

A smooth vector field that effectively avoids an obstacle should have the velocity $\dot{\boldsymbol{\xi}}$ directed away from or perpendicular to the normal $\boldsymbol{n}(\boldsymbol{\xi}) \in \mathbb{R}^N$ as defined in (3.15). This principle is commonly expressed as follows (S. M. Khansari-Zadeh and Billard, 2012; Feder and J.-J. Slotine, 1997):

$$\left\langle \boldsymbol{n}(\boldsymbol{\xi}), \dot{\boldsymbol{\xi}} \right\rangle \geq 0 \quad \forall\, \boldsymbol{\xi} \in \mathcal{X}^b \tag{7.1}$$

	RF	NF	MuMo	VF-CAPF	ROAM
Local minima free		✓	✓	(✓)	✓
Switching free	✓	✓	✓		✓
Overlapping regions	✓	✓	✓		✓
Nonlinear dynamics	✓		(✓)	✓	✓
Dynamic environments	✓		✓	✓	✓
(Optional) repulsion	✓				✓
Inverted obstacle	✓	✓	✓		✓
Trees of Obstacles	✓	✓			✓

Table 7.1: Repulsive fields (RF) (Khatib, 1986), navigation functions (NF) (D. E. Koditschek and Rimon, 1990), mulitple-modulation method (MuMo) (L. Huber, J.-J. Slotine, and Billard, 2022a), vector field collision avoidance path following (VF-CAPF) (Yao, B. Lin, et al., 2022), and the proposed rotational obstacle avoidance method (ROAM) are compared across multiple performance criteria.

7.2.1 Vector Rotation for Collision Avoidance

We propose a method called Rotational Obstacle Avoidance Method (ROAM) to achieve collision-free motion. ROAM smoothly adjusts the initial velocity $\boldsymbol{f}(\boldsymbol{\xi})$ as given in Eq. (3.1), rotating it towards a feasible half-space as the position approaches the obstacle (Fig. 7.2). The steps involved in ROAM for avoiding a single obstacle are as follows:

1. Evaluation of pseudo-tangent direction $\boldsymbol{e}(\boldsymbol{\xi})$ using the convergence direction $\boldsymbol{c}(\boldsymbol{\xi})$ (Sec. 7.2.1)

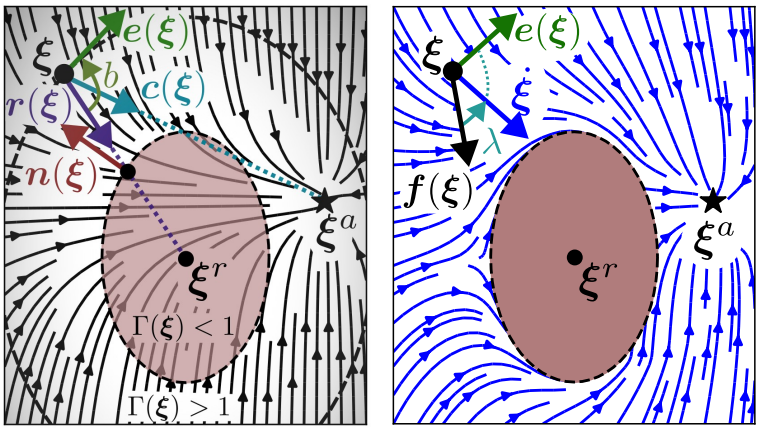
2. Rotation of the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ towards the pseudo-tangent $\boldsymbol{e}(\boldsymbol{\xi})$ to obtain the collision-free direction $\dot{\boldsymbol{\xi}}$ (Sec. 7.2.1).

3. Evaluation of the velocity magnitude $h(\boldsymbol{\xi})$ to ensure a smooth vector field (Sec. 7.2.1)

Evaluation of Preferred Tangent Direction

At each position on the surface of an obstacle, there can exist multiple tangent directions. In the case of $d = 2$, there are exactly two tangent directions, while for $d \geq 3$, there are infinitely many. To construct a smoothly defined pseudo-tangent $\boldsymbol{e}(\xi)$, we introduce the concept of convergence dynamics, which are used to obtain the pseudo-tangent:

Definition 7.2.1 (Convergence Dynamics)**.** The C^1-smooth convergence-dynamics $\boldsymbol{c} : \mathbb{R}^N \to \mathbb{R}^N$ guides the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ around obstacles. The convergence dynamics are locally straight according to Definition 3.2.2, on the surface of the obstacle $\mathscr{X}^b$ as defined in (3.3.2). Furthermore, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ is set to never be anti-collinear to the initial dynamics, i.e., $\left\langle \boldsymbol{f}(\boldsymbol{\xi}), \boldsymbol{c}(\boldsymbol{\xi}) \right\rangle \neq -\|\boldsymbol{f}(\boldsymbol{\xi})\| \, \|\boldsymbol{c}(\boldsymbol{\xi})\| \; \forall \boldsymbol{\xi} \in \mathbb{R}^N$.



(a) Initial dynamics (b) Rotated system

Figure 7.2: A nonlinear dynamical system $\boldsymbol{f}(\boldsymbol{\xi}) = \text{diag}(1 \;\; \Delta\boldsymbol{\xi}_{[2]})\|\Delta\boldsymbol{\xi}\|$ with $\Delta\boldsymbol{\xi} = \boldsymbol{\xi}^a - \boldsymbol{\xi}$ (a) is rotated to obtain the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ using ROAM with the help of the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$, the normal $\boldsymbol{n}(\boldsymbol{\xi})$, and convergence direction $\boldsymbol{c}(\boldsymbol{\xi})$. Using this, the global avoidance dynamics $\dot{\boldsymbol{\xi}}$ are computed, which ensure convergence towards the attractor $\boldsymbol{\xi}^a$ (black star) with a single saddle point on the surface (b).

In this section, we utilize convergence dynamics of the form $\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{\xi} - \boldsymbol{\xi}^a$. For more general convergence dynamics, please refer to Section 7.3.

The desired pseudo-tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is obtained by rotating the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ away from the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ until it lies in the tangent plane (Fig. 7.2). Since, in the angular space $\boldsymbol{k}(\boldsymbol{n}, \cdot)$, given in (4.7), any tangent vector lies at a distance $R^e = \pi/2$ to the normal direction, the tangent hyper-plane $\mathcal{T}$ forms a hyper-sphere in the direction space, as visualized in Figure 7.3. Hence, the rotation of the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ away from reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ can be evaluated by intersecting the line connecting $\boldsymbol{r}(\boldsymbol{\xi})$ and $\boldsymbol{c}(\boldsymbol{\xi})$ with a circle of radius $R^e \in [\pi/2, \pi]$, see Fig. 7.3. Thus, $\boldsymbol{e}(\boldsymbol{\xi})$ can be obtained through the following constraints[1]:

$$\begin{aligned} &\text{if} \quad \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c})\| \geq R^e \quad \text{then} \quad \boldsymbol{e} = \boldsymbol{c} \quad \text{otherwise} \\ &\qquad \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}) = (1-b)\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) + b\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c}) \\ &\text{such that} \quad \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e})\| = R^e, \quad b \in \mathbb{R}_{>0} \;\; \forall \boldsymbol{\xi} : \boldsymbol{c} \neq \boldsymbol{r} \end{aligned} \tag{7.2}$$

The solution to the above equality constraints is obtained analytically by solving a quadratic equation with respect to the scalar b.

As a result, the pseudo-tangent $\boldsymbol{e}(\xi)$ can either lie in the tangent plane $\mathcal{T}$ or point away from the obstacle, with a distance to the normal direction contained within the green region depicted in Figure 7.3. In the special case where $\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{r}(\boldsymbol{\xi})$, the intersection of the hyper-circle in Equation 7.2 does yield a solution. However, it is ensured that the final vector field $\dot{\boldsymbol{\xi}}$ is continuously

[1] For conciseness dependency on $\boldsymbol{\xi}$ is omitted.

defined, as discussed in Section 7.2.1.

Lemma 7.2.1. *Let us assume the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$, as given in Definition 7.2.1. The pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi}) \in \mathbb{R}^N$ obtained through* (7.2) *is smooth and satisfies the boundary inequality $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{e}(\boldsymbol{\xi}) \rangle \geq 0$, stated in* (7.1)*, at any position on the surface of the obstacle but the saddle point, i.e., $\boldsymbol{\xi} \in \mathscr{X}^b : \langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle \neq -1\}$*

Proof. For a starshaped obstacle, we have by definition of the starshaped kernel space in (3.4), that $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{r}(\boldsymbol{\xi}) \rangle < 0$. Hence, $\| \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) \| < \pi/2$, i.e., $\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})$ is strictly inside the hyper-sphere $\mathscr{T}$. Furthermore, as long as $\boldsymbol{c}(\boldsymbol{\xi}) \neq \boldsymbol{r}(\boldsymbol{\xi})$, the equality of finding the intersection of a line and a hyper-sphere from (7.2) has exactly one solution with $b > 0$.

Concerning the boundary condition; looking at the case of $\| \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c}) \| > R^e$. From the definition of the direction-space in (4.7), it follows that:

$$\langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0 \underset{\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{e}(\boldsymbol{\xi})}{\Rightarrow} \langle \boldsymbol{e}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0 \tag{7.3}$$

For the case that $\langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle < 0$, from (4.7) and (7.2) we can conclude:

$$\begin{aligned} \langle \boldsymbol{n}, \boldsymbol{e} \rangle &= \cos\left(\| \boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e}) \|\right) = -\cos\left(\| \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}) \|\right) \\ &\geq -\cos\left(R^e\right) \geq -\cos\left(\pi/2\right) = 0 \end{aligned} \tag{7.4}$$

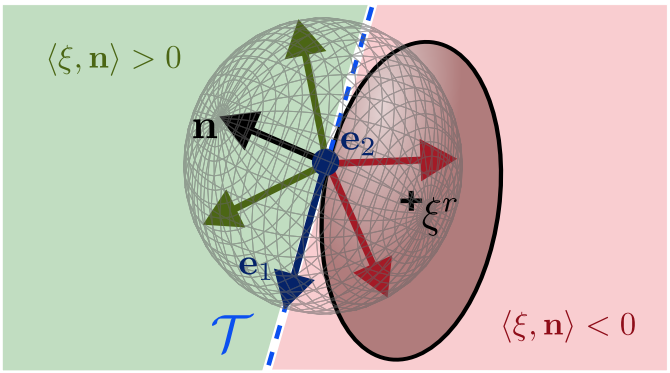
Hence, we have a uniquely defined pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$, which satisfies the boundary condition given in (7.1). □

Rotation Towards Tangent Direction

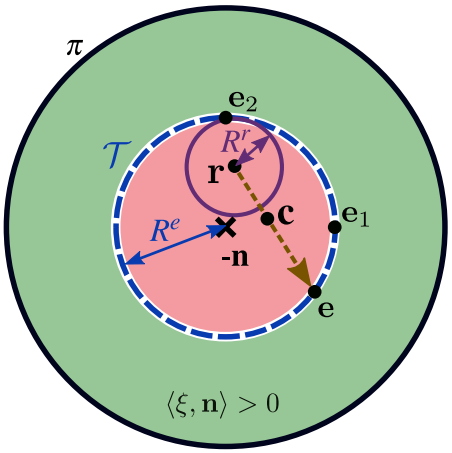
In a second step, the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is *rotated* towards the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$. This rotation operation is performed in the direction space, as described in Section 4.1, but this time with respect to the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$:

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \bar{\boldsymbol{k}}\Big(\boldsymbol{c}, \big(1 - \lambda(\boldsymbol{\xi})\big)\boldsymbol{k}\left(\boldsymbol{c}, \boldsymbol{f}\right) + \lambda(\boldsymbol{\xi})\, \boldsymbol{k}\left(\boldsymbol{c}, \boldsymbol{e}\right)\Big) \quad \text{with} \\ \lambda(\boldsymbol{\xi}) &= \left(\frac{1}{\Gamma(\boldsymbol{\xi})}\right)^q, \; R^r = \min\Big(R^e - \| \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) \|, \frac{\pi}{2}\Big) \\ q &= \max\left(1, \frac{R^r}{\Delta \boldsymbol{k}(\boldsymbol{c})}\right)^s, \quad \Delta \boldsymbol{k}(\boldsymbol{c}) = \| \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) - \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c}) \| \end{aligned} \tag{7.5}$$

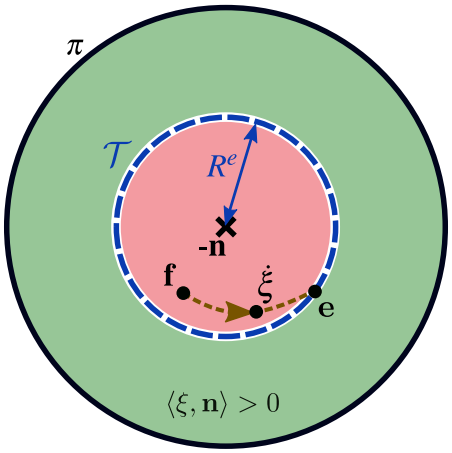
The rotation weight $\lambda(\boldsymbol{\xi}) \in [0, 1]$ is determined based on the inverse of the distance Γ to ensure that the rotation has a decreasing influence as the distance increases. This allows for a smooth transition in the avoidance behavior. Additionally, the smoothing factor q is introduced, which gradually decreases to zero when the convergence dynamics point towards the robot. This effectively cancels the rotation avoidance effect in regions where the tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is not defined,



(a) Euclidean space



(b) Obtaining pseudo-tangent $\boldsymbol{e}$



(c) Obtaining final velocity $\dot{\boldsymbol{\xi}}$

Figure 7.3: When approaching the surface of the obstacle, the collision-free vectors are located on one side of the tangent plane $\mathcal{T}$ formed by the linearly independent tangent vectors $\boldsymbol{e}_{(\cdot)}$ (a). These unit vectors can be projected onto a circular hypersphere of dimension $N-1$, where the collision-free pseudo tangent $\boldsymbol{e}$ is calculated (b). Any vector $\boldsymbol{\xi}$ pointing towards the obstacle, i.e., $\langle \boldsymbol{n}, \boldsymbol{\xi} \rangle < 0$ (a), is projected to lie inside the circle of radius $\pi/2$ (b, c), while a vector pointing away from the obstacle is projected to the outer ring.

resulting in a smooth avoidance behavior across those positions. The impact of the smoothness constant $s \in \mathbb{R}_{>0}$ can be observed in Figure 7.4, and unless otherwise specified, we use $s = 0.3$.

Lemma 7.2.2. *The vector field $\dot{\boldsymbol{\xi}}$ obtained through rotation as given in* (7.5) *satisfies the boundary condition as defined in* (7.1) *if the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is tangent or pointing away from the obstacle, i.e., $\langle \boldsymbol{e}, \boldsymbol{n} \rangle \geq 0$, $\forall \boldsymbol{\xi} : \Delta \boldsymbol{k}(\boldsymbol{c}) \neq 0$.*

Proof. When approaching the obstacle, the rotated velocity $\dot{\boldsymbol{\xi}}$ is evaluated as:

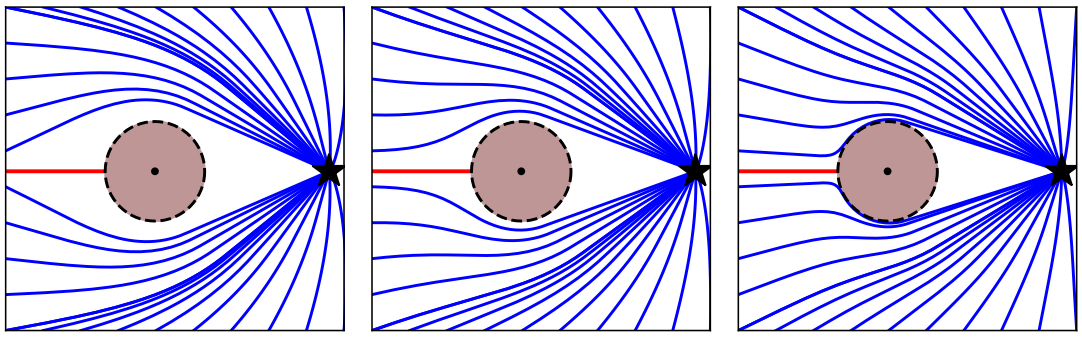
$$\lim_{\Gamma(\boldsymbol{\xi})\to 1,} \lambda(\boldsymbol{\xi}) = 1 \quad \underset{\text{with (4.6)}}{\Rightarrow} \quad \lim_{\lambda \to 1} \left\langle \boldsymbol{n}, \dot{\boldsymbol{\xi}} \right\rangle = \langle \boldsymbol{n}, \boldsymbol{e} \rangle \geq 0 \tag{7.6}$$

using Lemma 7.2.1 and that $\Delta \boldsymbol{k}(\boldsymbol{c}) \neq 0 \Rightarrow q > 0$ with the smoothness factor q given in (7.5).

Following the same logic, we can also analyze the behavior far away from obstacles:

$$\lim_{\Gamma(\boldsymbol{\xi})\to \infty} \lambda(\boldsymbol{\xi}) = 0 \;\Rightarrow\; \lim_{\lambda \to 0} \dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{7.7}$$

□



(a) s =1e-6 (b) $s = 0.3$ (c) $s = 1.0$

Figure 7.4: A smaller smoothness-constant s increases the reactivity when approaching an obstacle (a), while a larger value ensures a smoother transition across the (red) saddle point trajectory (b, c).

Evaluation of Speed

The directional space mapping $\boldsymbol{k}(\cdot)$ and its inverse mapping $\bar{\boldsymbol{k}}(\cdot)$ operate on unit vectors in the original space. As a result, the algorithm in (7.5) modifies the direction of the initial dynamics, rather than its magnitude. To incorporate magnitude control in a decoupled manner, we introduce an additional component using $h(\boldsymbol{\xi}) : \mathbb{R}^N \to [0, 1]$:

$$\|\dot{\boldsymbol{\xi}}\| = h(\boldsymbol{\xi}) \|\boldsymbol{f}(\boldsymbol{\xi})\| \tag{7.8}$$

The stretching factor $h(\boldsymbol{\xi})$ is designed to slow down when pointing towards an obstacle, and the effect decreases with increasing distance from the obstacle:

$$h(\boldsymbol{\xi}) = \min\left(1, \left(\frac{\|\Delta\boldsymbol{k}(\boldsymbol{c})\|}{R^r}\right)^2 + \left(1 - \frac{1}{\Gamma(\boldsymbol{\xi})}\right)^2\right) \tag{7.9}$$

where the reference radius R^r and the rotation space distance $\Delta\boldsymbol{k}(\boldsymbol{c})$ are given in (7.5). In Figure 7.5, the velocity scaling can be observed to decrease the magnitude of the vectors pointing toward the obstacle.

Theorem 7.2.3. *Consider a vector field $\dot{\boldsymbol{\xi}} \in \mathbb{R}^N$ obtained after a local rotation as defined in* (7.5) *of an initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ and fixed point at $\boldsymbol{\xi}^a$, with pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ defined in* (7.2) *with respect to initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$, is locally straight on the surface of the obstacle $\mathcal{X}^b$ according to Definition 3.2.2, and motion scaling $h(\boldsymbol{\xi})$ according to* (7.9)*. Any motion starting in free space $\{\boldsymbol{\xi}\}_0 \in \mathcal{X}^e$ which evolves according to $\dot{\boldsymbol{\xi}}$ will stay in free space for finite time $\{\boldsymbol{\xi}\}_t \in \mathcal{X}^e$ with $t \in \mathbb{N}_{>0}$ and maintains the stationary point, i.e., $\dot{\boldsymbol{\xi}} = \mathbf{0}$ if $\boldsymbol{f}(\boldsymbol{\xi}) = \mathbf{0}$*

Proof. From Lemma 7.2.1, we know $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{e}(\boldsymbol{\xi}) \rangle \geq 0$ and Lemma 7.2.2 states that $\langle \boldsymbol{e}, \boldsymbol{n} \rangle \geq 0 \ \forall \boldsymbol{\xi} : \Delta\boldsymbol{k}(\boldsymbol{c}) \neq 0$. Additionally, in the latter case, the magnitude scaling from (7.9) evaluates as

$$\lim_{\|\Delta\boldsymbol{k}(\boldsymbol{c})\| \to 0, \Gamma(\boldsymbol{\xi}) \to 1} h(\boldsymbol{\xi}) = 0 \tag{7.10}$$

Hence, the speed reaches smoothly zero as we approach the saddle point, and its direction does not matter to fulfill smoothness and the boundary condition given in (7.1).

Furthermore, far away from the obstacle, we have:

$$\lim_{\Gamma(\boldsymbol{\xi}) \to \infty} h(\boldsymbol{\xi}) = 1 \tag{7.11}$$

Hence, the magnitude is equal to the original magnitude and only vanishes around existing stationary points, i.e., $\boldsymbol{f}(\boldsymbol{\xi}) = 0$. □

The spurious stationary point on the surface is given by $\{\boldsymbol{\xi} : \boldsymbol{\xi} \in \mathcal{X}^b, \|\boldsymbol{k}(\boldsymbol{c})\| = 0\}$. Furthermore, by construction, the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ defined in (7.2) points away from the reference direction and hence points away from the stationary point on the surface of the obstacle. Hence, it is a saddle point with a single trajectory converging to it $\mathcal{X}^s$.

7.2.2 Surface Repulsion

n the vicinity of critical obstacles, it may be desirable to incorporate active repulsion from them without significantly altering their shape. This can be achieved by introducing a behavior similar to artificial potential fields (Rimon and D. Koditschek, 1992), but without introducing spurious attractors in free space. Building upon the developments presented in this section, we

can further refine the tangential radius. By increasing the tangential radius such that $R^e \in]\pi/2, \pi]$, values larger than $\pi/2$, leads to a repulsive behavior while maintaining the avoidance properties (Fig. 7.5).

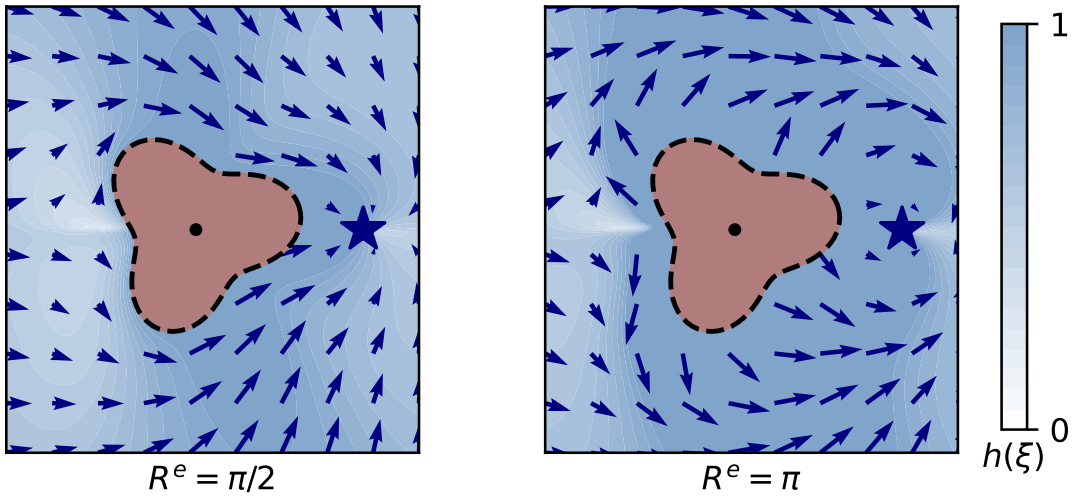


Figure 7.5: While a surface repulsion radius of $R^e = \pi/2$ ensures collision avoidance, an increased radius, i.e., $R^e \in]\pi/2, \pi]$ leads to increased repulsion around the obstacle. The initial dynamics are given by $\boldsymbol{f}(\boldsymbol{\xi}) = (\boldsymbol{\xi} - \boldsymbol{\xi}^a)$.

7.2.3 Inverted Obstacles

Initial dynamics might be contained within a boundary, such as an agent moving in a room or a robot staying within its joint limits. These constraints can be incorporated by inverting an obstacle and ensuring that the dynamics remain inside a boundary hull (Fig. 7.6). Analogously to Chapter 5 , this is achieved by inverting the distance function $\Gamma(\boldsymbol{\xi})$, the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ and normal vector $\boldsymbol{n}(\boldsymbol{\xi})$, defined in Sec. 3.4. The distance function divides the space into free, boundary, and interior points, as described in (3.7). Consequently, the distance for the boundary obstacle can be defined as follows:

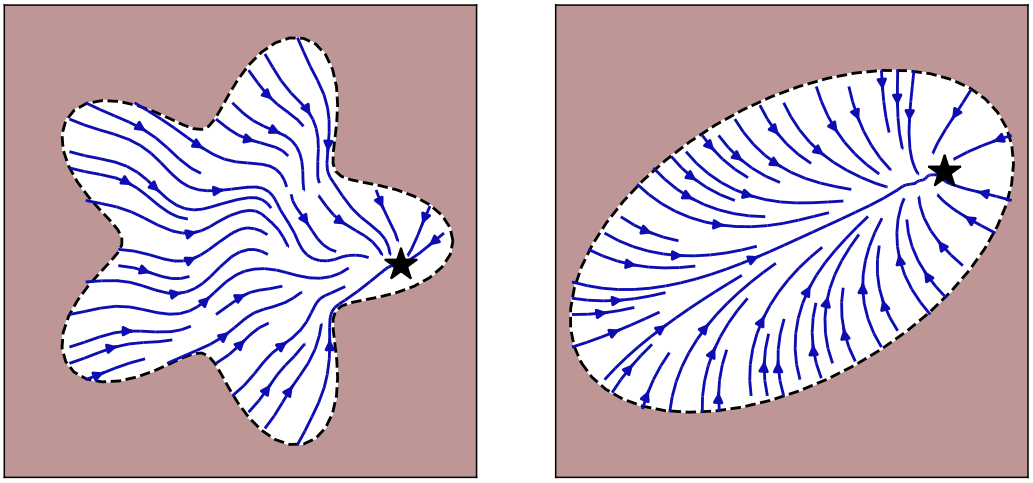
$$\Gamma(\boldsymbol{\xi}) = \left(R(\boldsymbol{\xi}) / \|\boldsymbol{\xi} - \boldsymbol{\xi}^r\|\right)^{2p} \quad \forall \boldsymbol{\xi} \in \mathbb{R}^N \setminus \boldsymbol{\xi}^r \tag{7.12}$$

with power weight $p \in \mathbb{R}_{>0}$, we choose $p = 1$.

As the normal direction points away from the surface, it naturally points towards the interior of an inverted obstacle, in contrast to the normal direction of a regular obstacle. Consequently, in order to utilize the star-shaped constraint from (7.5) of $\langle -\boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0$ for rotational obstacle avoidance, we need to flip the reference direction:

$$\boldsymbol{r}(\boldsymbol{\xi}) = \left(\boldsymbol{\xi}^r - \boldsymbol{\xi}\right) / \|\boldsymbol{\xi}^r - \boldsymbol{\xi}\| \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^N \setminus \boldsymbol{\xi}^r \tag{7.13}$$

Using the *inverted* values, the ROAM can be applied on as described throughout this section (Fig. 7.6).



(a) Wavy DS in starshape

(b) Linear DS in ellipse

Figure 7.6: The inverted obstacle description is used to keep nonlinear dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ within star-shaped boundaries, in (a) with $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{R}(\sin(\|\boldsymbol{\xi}^a - \boldsymbol{\xi}\|)(\boldsymbol{\xi}^a - \boldsymbol{\xi})$ where $\boldsymbol{R}(\cdot)$ is a two dimensional rotation matrix and $\boldsymbol{\xi}^a$ the attractor. Further, it can be used to create active repulsion from walls, as in (b) for globall straight dynamics with attractor $\boldsymbol{\xi}^a$.

Lemma 7.2.4. *The rotated dynamics $\dot{\boldsymbol{\xi}}$ evaluated according to* (7.5) *are C^1-smooth and collision-free as shown in Theorem 7.2.3 when navigating within a boundary described as an inverted obstacle with distance function $\Gamma(\boldsymbol{\xi})$ as defined in* (7.12)*, and reference vector $\boldsymbol{r}(\boldsymbol{\xi})$ as given in* (7.13).

Proof. Since the distance function $\Gamma(\boldsymbol{\xi})$, average vector $\boldsymbol{n}(\boldsymbol{\xi})$, and reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ possess all the necessary properties outlined in Theorem 7.2.3, the collision avoidance properties directly carry over.

The distance function $\Gamma(\boldsymbol{\xi})$ from (7.12) and normal direction (7.13) are not defined at the reference point $\boldsymbol{\xi}^r$. However, at this point the rotation weight $\lambda(\boldsymbol{\xi})$ reaches zeros:

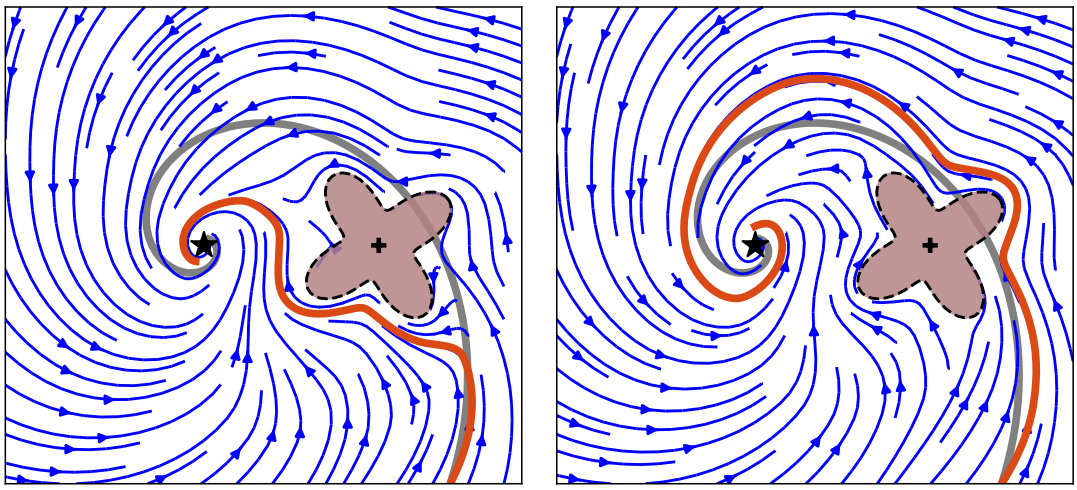
$$\lim_{\xi \to \xi^r} \Gamma(\boldsymbol{\xi}) \to \infty \underset{\text{using (7.5)}}{\Rightarrow} \lim_{\xi \to \xi^r} \lambda(\boldsymbol{\xi}) = 0 \;\Rightarrow\; \lim_{\xi \to \xi^r} \dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{7.14}$$

Thus, the rotation has no effect, and the system is smoothly defined. □

A more detailed analysis of inverted obstacles can be found in Chapter 5 . Further development applies to both standard and inverted obstacles if not stated otherwise.

7.3 General Nonlinear Motion

For systems characterized by small nonlinearities and a single attractor, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ exhibit global straightness, as defined in Definition 3.2.2, resulting in desirable behavior. However, in systems with high nonlinearities, this can lead to avoidance patterns that do not accurately reflect the global dynamics, as illustrated by the example shown in Figure 7.7a. To address this issue, this section introduces modified convergence dynamics that aim to maintain similarity with the original dynamics while being locally straight, as depicted in Figure 7.7b.



(a) Globally straight $\boldsymbol{c}(\boldsymbol{\xi})$ (b) Locally straight $\boldsymbol{c}(\boldsymbol{\xi})$

Figure 7.7: For motion with initial dynamics of $\boldsymbol{f}(\boldsymbol{\xi}) = \begin{bmatrix} -1 & 2 \\ -2 & -1 \end{bmatrix} \boldsymbol{\xi}$ (gray) and a single stationary point (black star), employing a globally straight convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ results in a trajectory (orange) that deviates significantly from the original motion (a). In contrast, by utilizing locally straight convergence dynamics, the trajectory (b) maintains similarity to the original motion throughout its course.

7.3.1 Nonlinear Motion without Stationary Point

Let us first focus on initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ which do not have any directional singularity point, i.e., $\nexists \boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0$. The convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ of such initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is constructed by evaluating as a weighted sum of the initial velocity at the reference point of the obstacle $\boldsymbol{\xi}^r$ and at position $\boldsymbol{\xi}$:

$$\boldsymbol{c}(\boldsymbol{\xi}) = w^c(\boldsymbol{\xi})\boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} (1 - w^c(\boldsymbol{\xi}))\boldsymbol{f}(\boldsymbol{\xi}) \tag{7.15}$$

The convergence weight is chosen such that the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ is straight on the surface of the obstacle:

$$w^c(\boldsymbol{\xi}) = \begin{cases} 1 & \text{if } \Gamma(\boldsymbol{\xi}) < 1 \\ 1/\Gamma(\boldsymbol{\xi}) & \text{otherwise} \end{cases} \tag{7.16}$$

where $\hat{+}$ describes the rotational summing described in Section 4.2. Note that the rotation summing from (4.15) is not defined for two anti-collinear vectors. Hence, we cannot apply this summing at a directional singularity point, as will be further discussed in the next subsection.

Lemma 7.3.1. *The convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi}) : \mathbb{R}^N \to \mathbb{R}^N$ as proposed in* (7.15) *for initial dynamics without directional singularity points, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \emptyset$, are straight according to Definition 3.2.2 on the surface and inside the obstacle, i.e, $\boldsymbol{c}(\boldsymbol{\xi}) \in \mathscr{F}^s$, $\boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i$.*

Proof. Inside and on the surface of the obstacle, we have:

$$\boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i \quad \Rightarrow \quad \Gamma(\boldsymbol{\xi}) \leq 1 \quad \Rightarrow \quad w^c(\boldsymbol{\xi}) = 1 \tag{7.17}$$

Hence the dynamics in this region are given as:

$$\boldsymbol{c}(\boldsymbol{\xi}) = 1\boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} 0\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}^r) \qquad \boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i \tag{7.18}$$

Thus, we have locally collinear dynamics. □

As the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ are straight on the surface of the obstacle, they can be used in the rotational obstacle avoidance method defined in Section 7.2, see Figure 7.8.

7.3.2 Nonlinear Motion in the Presence of a Stationary Point

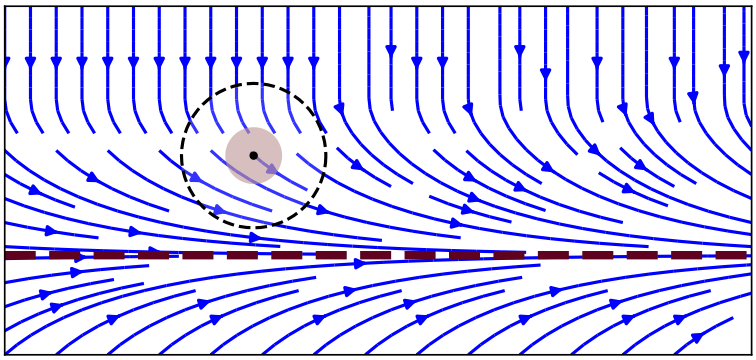
Many initial dynamics can be characterized by a motion with a single directional singularity point, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \{\boldsymbol{\xi}^a\}$. This singularity point may correspond to a desired attractor point or arise from a limit cycle where an unstable stationary point resides at its center. In the presence of such a singularity point, the directional summing approach employed in (7.15) cannot be directly applied. This is because we cannot smoothly define local rotations around $\boldsymbol{\xi}^a$. However, we can overcome this limitation by *unfolding* the space using a sequence of mappings (Fig. 7.9).

Shrinking and Inverse Mapping

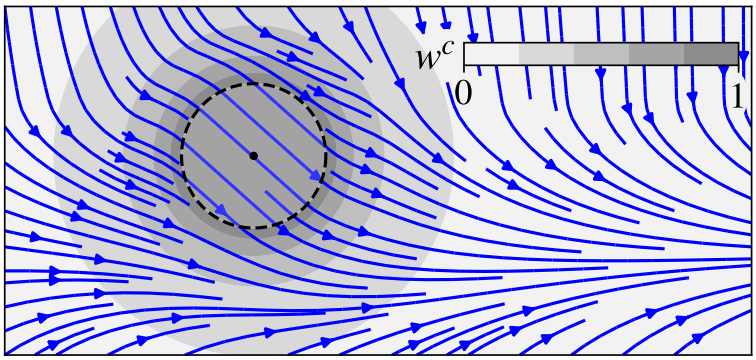
The first step is to shrink the obstacles to a single point, i.e., all boundary points of the obstacle $\mathscr{X}^b$ are mapped to the reference point $\boldsymbol{\xi}^r$.

Definition 7.3.1 (Shrinking Mapping)**.** The shrinking mapping $\boldsymbol{m}^s(\boldsymbol{\xi}) : \mathscr{X}^e \to \mathbb{R}^N \setminus \boldsymbol{\xi}^r$ defined as

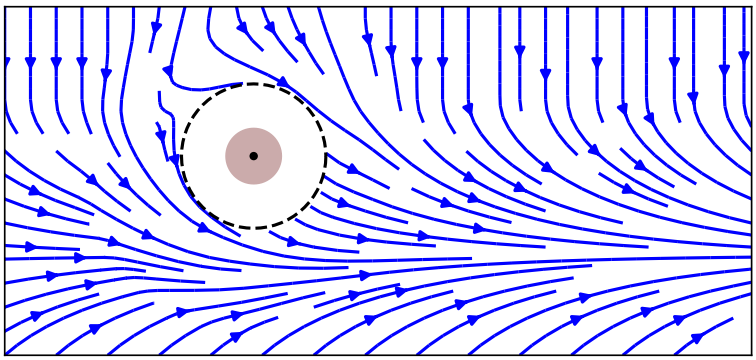
$$\boldsymbol{m}^s(\boldsymbol{\xi}) = \boldsymbol{\xi}^r + \boldsymbol{r}(\boldsymbol{\xi}) \left(\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| - \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \right) \ \forall \boldsymbol{\xi} \in \mathscr{X}^e \tag{7.19}$$



(a) Nonlinear dynamics to follow the line at $y = 0$ in red



(b) Convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ are locally straight in the gray subset



(c) Rotated avoidance without local minima

Figure 7.8: The line following dynamics $\boldsymbol{f}(\boldsymbol{\xi}) = [1 \quad -\boldsymbol{\xi}_2]^T$ shown in (a) uses locally straight convergence dynamics (b) to ensure the absence of local minima when avoiding the obstacle (c).

is a bijection, and it maps the point on the obstacle's surface to the obstacle's reference direction, i.e., $\lim_{\Gamma(\boldsymbol{\xi})\to 1} \boldsymbol{m}^s(\boldsymbol{\xi}) \to \boldsymbol{\xi}^r$.

Analogously, we define the inverse of this mapping:

Definition 7.3.2 (Inflating Mapping)**.** The inflating mapping $\boldsymbol{m}^i(\boldsymbol{\xi}) : \mathbb{R}^N \setminus \boldsymbol{\xi}^r \to \mathscr{X}^e$ defined as:

$$\boldsymbol{m}^i(\boldsymbol{\xi}) = \boldsymbol{\xi}^r + \boldsymbol{r}(\boldsymbol{\xi}) \left(\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| + \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \right) \;\; \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^r \tag{7.20}$$

is a bijection and the inverse function of the shrinking mapping defined in (7.19), i.e., $\boldsymbol{m}^i \circ \boldsymbol{m}^s(\boldsymbol{\xi}) = \boldsymbol{\xi} \;\; \forall \boldsymbol{\xi} \in \mathscr{X}^e$

The effect of shrinking and inflation mapping can be observed in Figure 7.9.

Folding Mapping

Let us introduce a *folding* mapping $\boldsymbol{m}^f(\boldsymbol{\xi})$, which moves the stationary point infinitely far away. Hence, the dynamics in the mapped space are without directional singularity point, which can be treated with the method introduced in Section 7.3.1. Conversely, the surface of the obstacle should not be affected by the mapping. The desired properties of this folding mapping are given as follows:

1. the stationary point gets mapped infinitely far away

$$\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}^a} \Gamma \left(\boldsymbol{m}^f(\boldsymbol{\xi}) \right) \to \infty \tag{7.21}$$

2. at the reference point (which represents the surface of the obstacle after the shrinking mapping), the effect of the mapping vanishes

$$\boldsymbol{\xi}^r = \boldsymbol{m}^f(\boldsymbol{\xi}^r) \tag{7.22}$$

Let us first define the unit directions in the coordinate system, which has its center at the stationary point and the first axis points towards the reference direction of the obstacle:

$$\hat{\boldsymbol{\xi}} = \boldsymbol{B}^T (\boldsymbol{\xi} - \boldsymbol{\xi}^a) / \|\boldsymbol{\xi} - \boldsymbol{\xi}^a\| \tag{7.23}$$

where $\boldsymbol{B}$ is the orthonormal matrix of which the first row aligns with $\boldsymbol{\xi}^r - \boldsymbol{\xi}^a$

The desired constraints of the mapping from (7.21) and (7.22) can be achieved by uniformly

stretching along dimensions $i \in [2..N]$ of $\hat{\boldsymbol{\xi}}$ as:

$$\begin{aligned} \boldsymbol{s}_i = \left(2/(1+p)-1\right)^g \frac{\hat{\boldsymbol{\xi}}_i}{\|\hat{\boldsymbol{\xi}}_{[2:N]}\|} \quad p \in]-1,1],\ \boldsymbol{\xi} \neq \boldsymbol{\xi}^a \\ \text{with} \qquad p = \frac{\langle \boldsymbol{\xi}-\boldsymbol{\xi}^a, \boldsymbol{\xi}^r-\boldsymbol{\xi}^a \rangle}{\|\boldsymbol{\xi}-\boldsymbol{\xi}^a\|\,\|\boldsymbol{\xi}^r-\boldsymbol{\xi}^a\|} \end{aligned} \tag{7.24}$$

where $g \in \mathbb{R}_{>0}$ is the power factor, we choose $g = 2$. The above equation pushes points opposite the attractor relative to the obstacle infinitely far away, i.e., $\lim_{\xi_1 \to -1} \boldsymbol{s}_i \to \infty$, see the green line in Figure 7.9.

Finally, the stretching of the *folding* mapping along dimension $i = 1$ is constructed as follows:

$$\boldsymbol{s}_1 = \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\| \left(1 + \ln\left(\frac{\|\boldsymbol{\xi}-\boldsymbol{\xi}^a\|}{\|\boldsymbol{\xi}^r-\boldsymbol{\xi}^a\|}\right)\right) \tag{7.25}$$

Using this, the folding mapping $\boldsymbol{m}^f(\boldsymbol{\xi}) : \{\boldsymbol{\xi} \in \mathbb{R}^N : \hat{\boldsymbol{\xi}}_1 \neq -1, \boldsymbol{\xi} \neq \boldsymbol{\xi}^a\} \to \mathbb{R}^N$ is defined as:

$$\boldsymbol{m}^f(\boldsymbol{\xi}) = \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^T(\boldsymbol{\xi}-\boldsymbol{\xi}^a) + \boldsymbol{\xi}^a \quad \forall \boldsymbol{\xi} \in \mathbb{R} \setminus \boldsymbol{\xi}^a \tag{7.26}$$

Lemma 7.3.2. *The mapping $\boldsymbol{m}^f(\boldsymbol{\xi}) : \boldsymbol{m}^f(\boldsymbol{\xi}) : \{\boldsymbol{\xi} \in \mathbb{R}^N : \hat{\boldsymbol{\xi}}_1 \neq -1, \boldsymbol{\xi} \neq \boldsymbol{\xi}^a\} \to \mathbb{R}^N$ as given in* (7.26) *is a bijection, and smoothly defined, i.e., $\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}_j} \boldsymbol{m}^f(\boldsymbol{\xi}_i) = \boldsymbol{m}^f(\boldsymbol{\xi}_j)$. Furthermore, the mapping has no effect at the reference point, i.e., $\boldsymbol{m}^f(\boldsymbol{\xi}^r) = \boldsymbol{\xi}^r$, and the attractor is mapped infinitely far away, i.e., $\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}^a} \Gamma\left(\boldsymbol{m}^f(\boldsymbol{\xi})\right) \to \infty$.*

Proof. The system is made up of the two transformations, one along the $\boldsymbol{\xi}^r - \boldsymbol{\xi}^a$ as given in (7.25), and one perpendicular to, given in in (7.24). Since all the underlying functions involved are smooth and bijective, the resulting transformation is also smooth and bijective. However, it is important to note that there are discontinuities at $\boldsymbol{\xi} = \boldsymbol{\xi}^a$ and $\hat{\boldsymbol{\xi}}_1 = -1$. This points are excluded from the input set (in Eq. 7.30 we will additionally ensure that the corresponding weight goes to zero, for a smooth effect).

Furthermore for an input value of $\boldsymbol{\xi}^r$ in (7.25), we get that $\boldsymbol{s}_1 = \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\|$, and using (7.24) we get $\boldsymbol{s}_i = 0$ with $i \in [2..d]$. Hence, the transformation in (7.26) yields,

$$\begin{aligned} \boldsymbol{m}^f(\boldsymbol{\xi}^r) &= \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^T(\boldsymbol{\xi}^r - \boldsymbol{\xi}^a) + \boldsymbol{\xi}^a \\ &= \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s}) \begin{bmatrix} 1 & 0 & .. & 0 \end{bmatrix}^T + \boldsymbol{\xi}^a \\ &= \boldsymbol{B} \begin{bmatrix} \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\| & 0 & .. & 0 \end{bmatrix}^T + \boldsymbol{\xi}^a \\ &= \boldsymbol{\xi}^r - \boldsymbol{\xi}^a + \boldsymbol{\xi}^a = \boldsymbol{\xi}^r \end{aligned} \tag{7.27}$$

For the region $\hat{\boldsymbol{\xi}}_1 = -1$, we get for the stretching factors $s_i \to \infty$ for $d \in [2..d]$ from (7.24). □

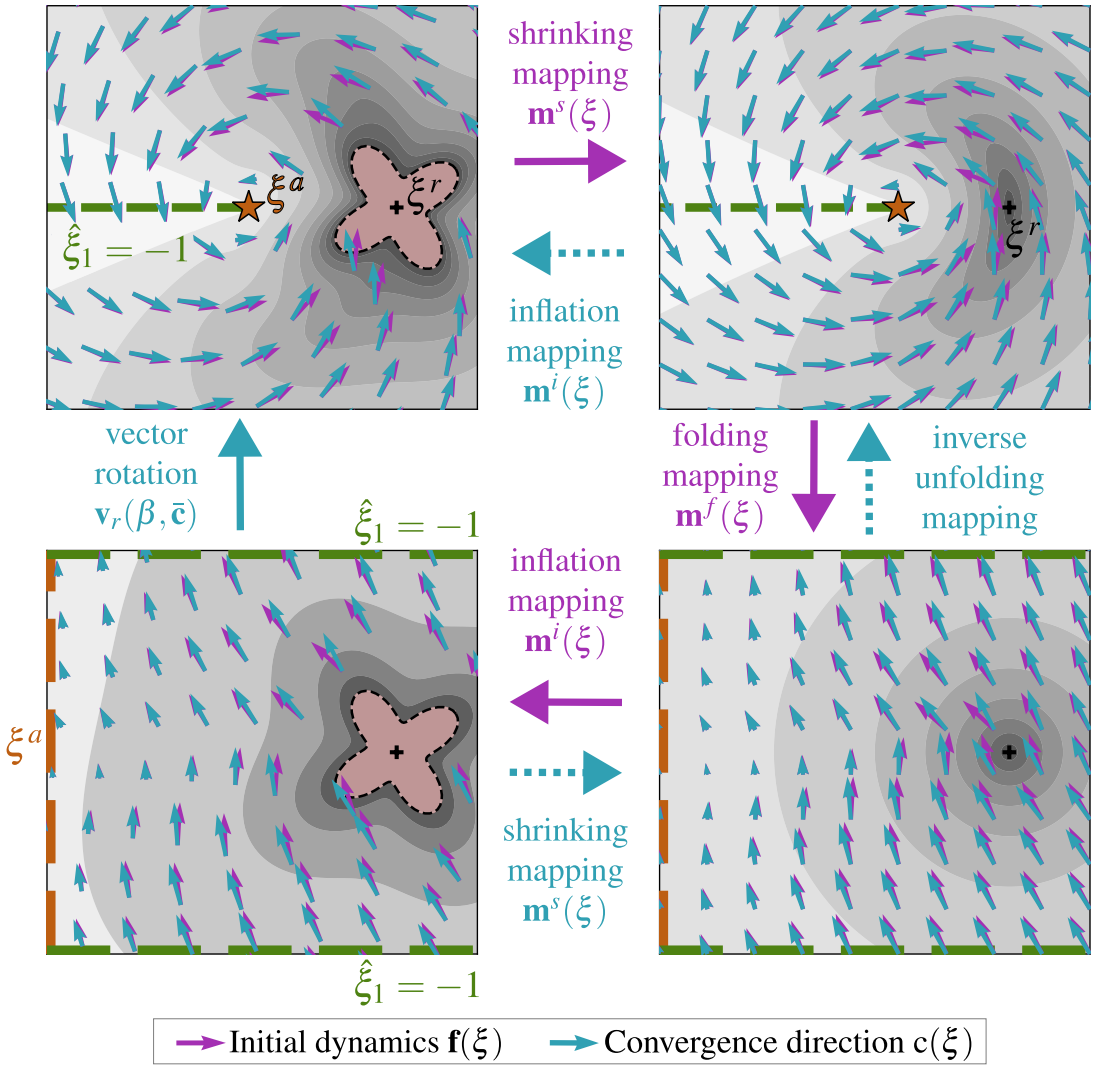


Figure 7.9: The initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ undergo a series of three consecutive mappings, depicted in a clockwise manner starting from the top left. The convergence weight $w^c(\boldsymbol{\xi})$ is evaluated in the transformed space, where its value is represented by the gray shading. Notably, the weight diminishes to zero at the attractor point $\boldsymbol{\xi}^a$. The directional summing operation from (7.15) is performed to obtain the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$. This step is carried out in the mapped space. To obtain the reverse mapping, the vector rotation $\boldsymbol{v}_r(\cdot)$ is simply evaluated in the original space. The folding mapping $\boldsymbol{m}^f(\boldsymbol{\xi})$ maps the attractor infinitely far away in the $-\hat{\boldsymbol{\xi}}_1$ direction, whereas the line with $\hat{\boldsymbol{\xi}}_1 = -1$ is folded out to be infinitely far in the directions of $\pm\hat{\boldsymbol{\xi}}_{[2..d]}$.

Evaluation of the Relative Rotation

The total mapping can be written as follows:

$$\boldsymbol{m}(\boldsymbol{\xi}) = \boldsymbol{m}^i \circ \boldsymbol{m}^f \circ \boldsymbol{m}^s(\boldsymbol{\xi}) \tag{7.28}$$

Since this mapping transforms the attractor infinitely far away, it produces a vector field without directional singularity, as Section 7.3.1 requires. Thereof, we can evaluate the convergence dynamics in the mapped space:

$$\bar{\boldsymbol{c}}(\boldsymbol{\xi}) = w^m \circ \boldsymbol{m}(\boldsymbol{\xi}) \boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} \big(1 - w^m \circ \boldsymbol{m}(\boldsymbol{\xi})\big) \boldsymbol{f}(\boldsymbol{\xi}) \tag{7.29}$$

the symbol $\hat{+}$ implies the directional summing as defined in Section 4.2.

The mapping weight is defined as:

$$w^m(\boldsymbol{\xi}) = 1/\sqrt{\big(\Gamma(\boldsymbol{\xi}) - 1\big)\big(\Gamma(m(\boldsymbol{\xi})) - 1\big) + 1} \tag{7.30}$$

This weight function is designed to ensure a decreasing influence as the distance from the obstacle increases, as well as a decreasing influence when the position is opposite to the singularity point relative to the obstacle (green line in Figure 7.9). Additionally, it ensures that the weight approaches one when the position lies on the surface of the obstacle.

Finally, the mapping into the original space can be made through a vector rotation

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{v}_r \left(\beta, \bar{\boldsymbol{c}}\right) \tag{7.31}$$

where the vector rotation $\boldsymbol{v}^r$ and angle $beta$ are obtained according to Eq. (4.15), using input vector $\boldsymbol{v}_i = \boldsymbol{m}(\boldsymbol{\xi}) - \boldsymbol{\xi}^a$ and output vector $\boldsymbol{v}_o = \boldsymbol{\xi} - \boldsymbol{\xi}^a$.

Theorem 7.3.3. *The smoothly defined convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ as given in* (7.31) *for obstacles in a vector field with a directional singularity point $\boldsymbol{\xi}^a$, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \{\boldsymbol{\xi}^a\}$ is ensured to be straight according to Definition 3.2.2 on the surface and inside the obstacle, i.e., $\boldsymbol{c}(\boldsymbol{\xi}) \in \mathcal{F}^s$, $\boldsymbol{\xi} \in \mathcal{X}^b \cup \mathcal{X}^i$.*

Proof. Lemma 7.3.2 states that the unfolding has no effect at $\boldsymbol{\xi}^r$ in the shrunk space, which is the surface of the obstacle in the original space as stated in Definition 7.3.1. Further, the inflating mapping is the inverse of the shrinking as stated in Definition 7.3.2. Thus, when approaching the surface, i.e., $\Gamma(\boldsymbol{\xi}) \to 1$, Lemma 7.3.1 transfers to Theorem 7.3.3. □

Obstacles Across the Attractor

When an obstacle spans across the attractor $\boldsymbol{\xi}^a$, i.e., $\Gamma(\boldsymbol{\xi}^a) \leq 1$, the unfolding mapping is not defined. Hence, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ must approach globally straight dynamics

according to Definition 3.2.2. To account for this, the initial dynamics are updated as follows:

$$\boldsymbol{f}(\boldsymbol{\xi}^r) \leftarrow w^\Gamma \operatorname{sign}\left(\nabla \boldsymbol{f}(\boldsymbol{\xi})|_{\boldsymbol{\xi}=\boldsymbol{\xi}^a}\right)(\boldsymbol{\xi}^r - \boldsymbol{\xi}^a) + (1 - w^\Gamma)\boldsymbol{f}(\boldsymbol{\xi}^r) \tag{7.32}$$

The weight is designed to reach one when the obstacle reaches the attractor, e.g., $w^\Gamma = 1/\Gamma(\boldsymbol{\xi}^a)$.

7.4 Multi-Obstacle Environments

In the presence of multiple obstacles, the rotational obstacle avoidance method for a single obstacle can be extended using a weighted rotational summing. First, the weighted convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ as introduced in (7.15) is averaged as follows:

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o \boldsymbol{c}_o(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o w_o^c \boldsymbol{f}(\boldsymbol{\xi}_o^r) \tag{7.33}$$

where $\hat{+}$ denotes the rotational summing as defined in Section 4.2.

The obstacle weights have been proposed in Chapter 3 as:

$$w_o(\boldsymbol{\xi}) = \frac{\tilde{w}_o(\boldsymbol{\xi})}{\sum_{i=1}^{N^{\text{obs}}} \tilde{w}_i(\boldsymbol{\xi})} \text{ with } w_o(\boldsymbol{\xi}) = \frac{1}{\Gamma_o(\boldsymbol{\xi})} \quad \forall\, \boldsymbol{\xi} \in \mathcal{X}^e \tag{7.34}$$

The weights associated with each obstacle ensure that their sum is at most one and converge to zero as the distance from the respective obstacle increases. Furthermore, on the surface of an obstacle o, the corresponding weight equals 1. Moreover, the weighting allows for overlapping of the influence regions of the obstacles.

These convergence dynamics are used to evaluate the preferred pseudo tangent for each obstacle $\boldsymbol{e}_o(\xi)$ as defined in (7.2). Finally, the rotation of the initial dynamics from (7.5) can be restated for multi-obstacle scenarios as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o \dot{\boldsymbol{\xi}}_o = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} \lambda_o w_o m(\boldsymbol{\xi}) \boldsymbol{e}_o(\boldsymbol{\xi}) \tag{7.35}$$

The method is summarized in Algorithm 3 and handles star-shaped obstacles and boundaries, as shown in Figure 7.10.

Lemma 7.4.1. *The weighted rotation of the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ from* (7.33) *and the final dynamics $\dot{\boldsymbol{\xi}}$ from* (7.35) *conserves impenetrability and saddle-point properties introduced in Theorem 7.2.3.*

Proof. When approaching an obstacle o, the weight defined in (7.34) results in $\lim_{\Gamma_o(\boldsymbol{\xi})\to 1} w_o(\boldsymbol{\xi}) = 1$. Hence the evaluation of the multi-obstacle avoidance converges to the single obstacle scenario of obstacle o. It follows that the properties of the single obstacle case are conserved locally. □

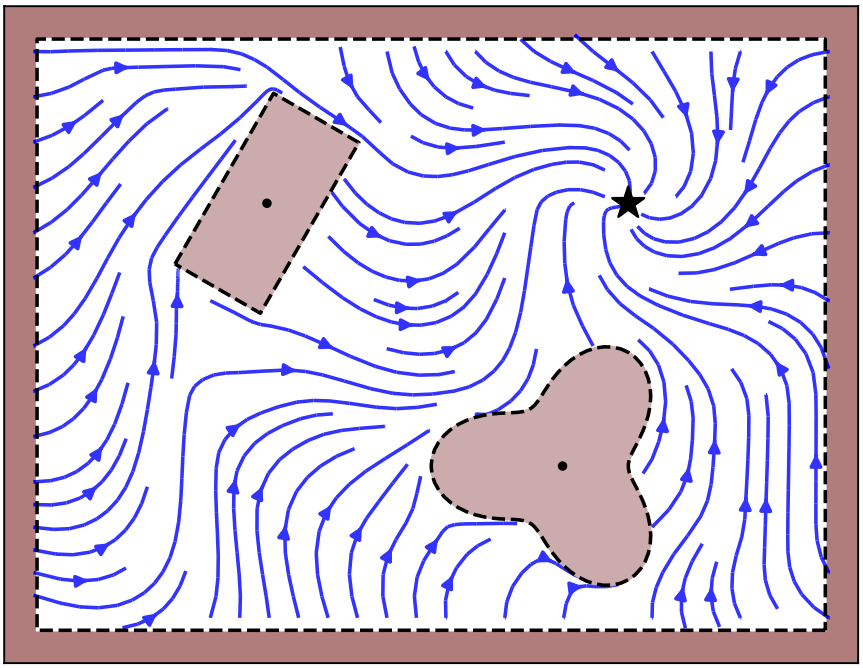


Figure 7.10: The rotational obstacle avoidance with respect to wavy dynamics as used in Figure 7.6a moves towards the attractor (black star) while avoiding collisions inside a rectangular hull with two obstacles.

7.4.1 Dynamic Environments

The closed-form description of the algorithm enables short computation time without the need for offline trajectory planning. Dynamic obstacles can be considered by transposing the avoided direction into the moving reference frame:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi})\boldsymbol{f}(\boldsymbol{\xi}) + \dot{\tilde{\boldsymbol{\xi}}} \quad \text{with} \quad \dot{\tilde{\boldsymbol{\xi}}} = \sum_{o=1}^{N^{\text{obs}}} w_o \dot{\tilde{\boldsymbol{\xi}}}_o \tag{7.36}$$

The method's application to dynamic obstacles is experimentally evaluated in Section 7.6.4.

7.4.2 Motion within Multiple Enclosing Hulls

For example, the outer boundary might not be star-shaped when a robot moves through a curvy corridor. However, such a general space can be divided into multiple stars-shapes (Lindemann and LaValle, 2009), which ROAM can use to guide a motion to stay within the boundary. This requires convergence dynamics, which transition between the obstacles as:

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{b=1}^{N^{bnd}} w_b(\boldsymbol{\xi})\boldsymbol{c}_b(\boldsymbol{\xi}) \tag{7.37}$$

Algorithm 3 Rotational Obstacle Avoidance Method (ROAM)

Input: $\boldsymbol{f}(\boldsymbol{\xi})$, N^{obs} obstacles
Output: $\dot{\boldsymbol{\xi}}$
for $o = 1$ **to** N^{obs} **do**
  $\tilde{w}_o(\boldsymbol{\xi}) = 1/\Gamma_o(\boldsymbol{\xi})$ {Evaluate weights (7.34)}
end for
$w_o(\boldsymbol{\xi}) = \tilde{w}_b(\boldsymbol{\xi}) / \sum_{i=1}^{N^{\text{obs}}} \tilde{w}_i(\boldsymbol{\xi})$ {Normalize weights (7.34)}
for $o = 1$ **to** N^{obs} **if** $w_o(\boldsymbol{\xi}) > 0$ **do**
  $\boldsymbol{f}(\boldsymbol{\xi}_o^r)$ {Compute DS at reference point}
end for
$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o w_o^c \boldsymbol{f}(\boldsymbol{\xi}_o^r)$ {Rotational average 7.33}
for $o = 1$ **to** N^{obs} **if** $w_o(\boldsymbol{\xi}) > 0$ **do**
  $\boldsymbol{e}_o(\boldsymbol{\xi})$ {Compute pseudo tangent (7.2)}
  $h(\boldsymbol{\xi})$ {Compute magnitude (7.9)}
end for
$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} \lambda_o w_o m(\boldsymbol{\xi}) \boldsymbol{e}_o(\boldsymbol{\xi})$ {Rot. average (7.35)}

where $N^{bnd} \in \mathbb{N}_{>0}$ is the number of boundaries. Furthermore, the weights of the multi-boundary environment are set to:

$$w_b(\boldsymbol{\xi}) = \frac{\max(\Gamma_b(\boldsymbol{\xi}), 1)) - 1}{\sum_{i=1}^{N^{bnd}} \max(\Gamma_i(\boldsymbol{\xi}), 1) - 1} \tag{7.38}$$

Where the local dynamics $\boldsymbol{c}_b(\boldsymbol{\xi})$ are locally straight according to Definition 3.2.2 in the subdomain of the surface of each boundary. The attractor of each hull $\boldsymbol{\xi}_b^a$ is placed such that it lies outside of the corresponding boundary b for all boundaries that do not contain the global attractor $\boldsymbol{\xi}^a$, i.e.,

$$\begin{cases} \boldsymbol{\xi}_b^a = \boldsymbol{\xi}^a & \text{if } \Gamma_b(\boldsymbol{\xi}^a) > 1 \\ \boldsymbol{\xi}_b^a \in \mathscr{X}_b^i \cup \mathscr{X}_b^b & \text{otherwise} \end{cases} \tag{7.39}$$

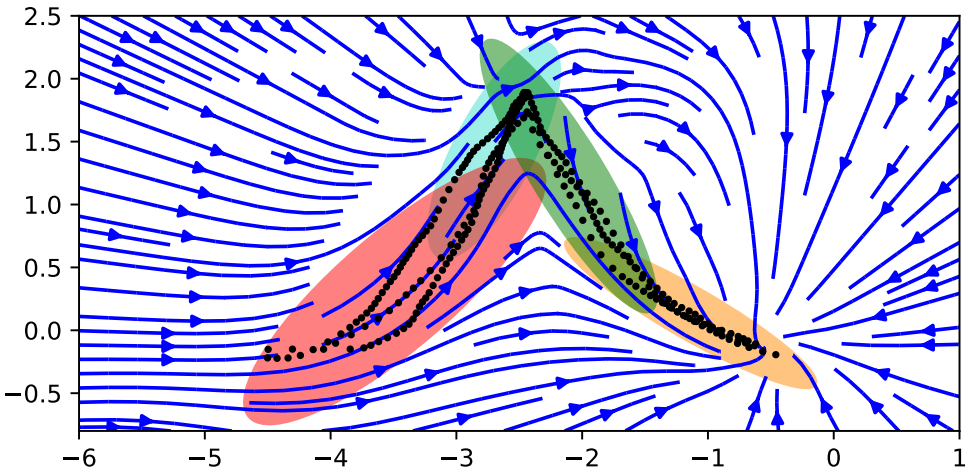
The boundary encapsulation can be seen in Figure 7.11.

Lemma 7.4.2. *A motion starting within multiple boundaries* $\{\boldsymbol{\xi}\}_0 \in \bigcup_{b \in N^{bnd}} \mathscr{X}_b^e$ *and evolving according to* $\dot{\boldsymbol{\xi}}$ *as described in* (7.35) *and with convergence dynamics* $\boldsymbol{c}(\boldsymbol{\xi})$ *defined in* (7.37) *will stay within the boundaries, i.e,* $\{\boldsymbol{\xi}\}_t \in \bigcup_{b \in N^{bnd}} \mathscr{X}_b^e \ \forall\ t \in \mathbb{N}_{>0}$.

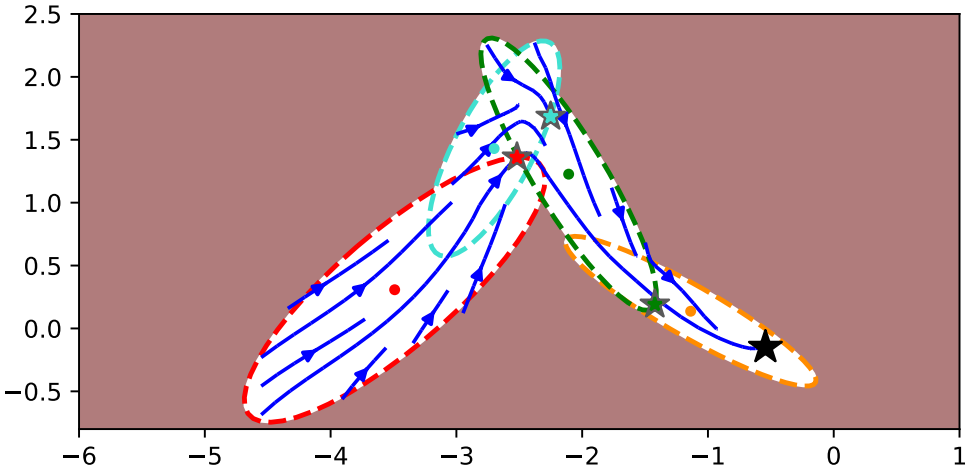
Proof. If $\boldsymbol{\xi}$ is within multiple boundaries, and it approaches boundary b, the weight given in in (7.38) evaluates to:

$$\lim_{\Gamma_b(\xi) \to 1} w_b(\boldsymbol{\xi}) = \frac{0}{0 + \sum_{i \neq b}^{N^{bnd}} \max(\Gamma_b(\boldsymbol{\xi}), 1)) - 1} = 0 \tag{7.40}$$

Hence, boundary b has no effect, and the motion can traverse any boundary $b = 1..N^{\text{bnd}}$. Until, $\boldsymbol{\xi}$



(a) The data (position and velocity) are fit using Gaussian Mixture Model (GMM) with four components (Pedregosa et al., 2011), which is used to predict the velocity.



(b) The GMM components are used as a multi-hull environment to bind the dynamics and enforce leaving the boundary in the direction of the local attractors $\boldsymbol{\xi}_b^a$ (colored stars).

Figure 7.11: Approaches for learning nonlinear motion from demonstration can guarantee stability by ensuring that the system asymptotically converges towards the attractor $\boldsymbol{\xi}^a$ (Perrin and Schlehuber-Caissier, 2016). However, they do not prohibit the dynamics of taking an undesired shortcut or moving away from the data, which is the known region (a). Convex hulls can be obtained through trajectory learning methods, by interpreting the Gaussian Mixture Model applied to the data as ellipse-shaped obstacles (Figueroa and Billard, 2018). ROAM can enforce the final dynamics to stay close to the data (b).

is within only one boundary b. In this case, the corresponding boundary weights simplify to:

$$w_b(\boldsymbol{\xi}) = \frac{\max(\Gamma_b(\boldsymbol{\xi}), 1)) - 1}{\max(\Gamma_b(\boldsymbol{\xi}), 1)) - 1 + \sum_{i \neq b}^{N^{bnd}} 0} = 1 \tag{7.41}$$

given that $\Gamma_b(\boldsymbol{\xi}) > 1$. Thus, the algorithm evolves according to the single-boundary case, i.e., collision avoidance with the boundary b is ensured. □

7.5 General Concave Obstacles

A general obstacle can be described as a union of multiple star-shaped obstacles (Rimon and D. E. Koditschek, 1991), also referred to as trees of stars. Extending the algorithm to such shapes enables navigating in many more environments. Let us for this introduce a general obstacle using nomenclature from graph theory (Knuth, 1997):

Definition 7.5.1 (Tree of Obstacles)**.** A tree of star-shaped obstacles represents a shape without holes. Each obstacle (node) in the tree can have multiple children (successors), but have exactly one single parent (predecessor), except for the root obstacle which does not have a parent. All obstacles have a non-zero intersection with their parent and children. Obstacles are assigned a level l in the tree, starting from $l = 0$ at the root.

7.5.1 Velocity Propagation through Obstacle Tree

The avoidance velocity $\dot{\boldsymbol{\xi}}$ in the presence of trees-of-obstacles is obtained through the summed average of a rotation-tree as described in Section 4.2. The tree is constructed as described in Algorithm 4, and the individual steps are detailed below.

Surface Point Propagation

The rotational avoidance of each obstacle $o = 1..N^{\text{com}}$ of the tree is obtained at the corresponding surface point $\boldsymbol{s}_o$. The surface points are obtained by propagating the position $\boldsymbol{\xi}$ through the obstacle tree, starting from the an obstacle o down to the root r:

$$\boldsymbol{s}_p = b(\boldsymbol{\xi}_c^r - \boldsymbol{s}_c) + \boldsymbol{s}_c \quad \text{such that} \quad \Gamma_p(\boldsymbol{s}_p) = 1 \tag{7.42}$$

where p is the parent of the component c. The factor $b \in \mathbb{R}_{>0}$ is evaluated such that $\boldsymbol{s}_p$ lies on the parent's surface. Note, that the obstacles are intersecting, hence we have $\Gamma(\boldsymbol{s}_p) < 1$. See the surface points of a three-component tree in Figure 7.12.

Algorithm 4 Avoidance of Tree-of-Obstacles

Input: $\boldsymbol{f}(\cdot)$, N^{com} obstacle-components
Output: $\dot{\boldsymbol{\xi}}$

$\boldsymbol{t}^r(\cdot)$ {Create rotation tree, see Algo. 1}
for $o = 1$ **to** N^{com} **do**
  $\boldsymbol{s}_0 = \boldsymbol{r}(\boldsymbol{\xi})$ {Set initial surface point}
  $w_o^h \leftarrow \Gamma_o(\boldsymbol{s}_0)$ {Compute hiding-weight (7.46)}
  $w_o \leftarrow \Gamma_o(\boldsymbol{\xi})$ {Compute obstacle weights}
  $c = o$ {Initialize c to obstacle o}
  for $l = l(n)$ **to** 0 **do** {From node to root}
    $\boldsymbol{s}_{p(c)} = b(\boldsymbol{\xi}_c^r - \boldsymbol{s}_c) + \boldsymbol{s}_c$ {Propagate reference (7.42)}
    $c = p(c)$ {Set iterator c to parent p}
  end for
  $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{\xi}^r)$ {Set initial tangent}
  for $c = 1$ **to** $l(n)$ **do** {From root to node}
    $\boldsymbol{v}^r(\cdot), \beta \leftarrow (\boldsymbol{s}_c - \boldsymbol{\xi}_c^r), (\boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r)$ {Get rot. (4.15)}
    $\boldsymbol{f}_c = \boldsymbol{v}^r(\boldsymbol{f}_{p(c)}, \beta)$ {Propagate rotated velocity (4.16)}
    $\boldsymbol{f}_c \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree}
  end for
  $\dot{\boldsymbol{\xi}}_o \leftarrow \boldsymbol{f}_o$ {Avoidance with propagated velocity (7.45)}
end for
$w_f = 1 - \sum_o w_o^h w_o$ {Weight of initial velocity $\boldsymbol{f}(\boldsymbol{\xi})$}
$\dot{\boldsymbol{\xi}} \leftarrow \boldsymbol{t}^r(w_f, w_o^h(\boldsymbol{\xi}))$ {Evaluate rotation tree, Algo. 1}

Velocity Propagation

We can now propagate the velocity $\boldsymbol{f}(\boldsymbol{\xi}^r)$ iteratively from parent $p(c)$ to the component c until we reach the respective obstacle o. The iteration starts at the root r and is done for each obstacle, except the root. The propagated velocity $\boldsymbol{f}_c$ is obtained as follows:

$$\boldsymbol{f}_c = \boldsymbol{v}^r(\boldsymbol{f}_{p(c)}, \phi_c) \qquad o = 1..N^{\text{obs}} \setminus r \tag{7.43}$$

where $\boldsymbol{v}^r(\cdot)$ is the vector rotation as described in (4.16). The vector rotation is obtained with respect to the vectors $\boldsymbol{v}_i = (\boldsymbol{s}_c - \boldsymbol{\xi}_c^r)$ and $\boldsymbol{v}_o = (\boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r)$ as described in (4.15). When constructing the obstacle tree, the parent of an obstacle needs to be chosen such that $\boldsymbol{v}_i$ and $\boldsymbol{v}_o$ are not anti-collinear. This is ensured if the direction from the component to the parent is never opposing the direction from the parent to the *grand-parent* (parent of the parent):

$$\frac{\langle \boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r, \boldsymbol{\xi}_{p(p(c))}^r - \boldsymbol{\xi}_{p(c)}^r \rangle}{\|\boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r\| \, \|\boldsymbol{\xi}_{p(p(c))}^r - \boldsymbol{\xi}_{p(c)}^r\|} \neq -1 \tag{7.44}$$

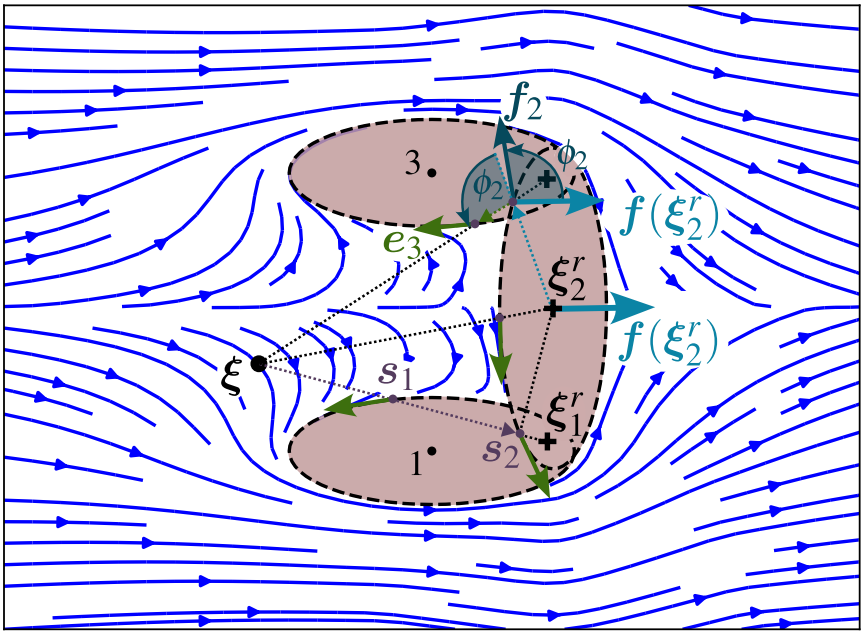


Figure 7.12: The surface points $\boldsymbol{s}_{(\cdot)}$ (purple) are evaluated for each obstacle up to the root, to then propagate the desired tangent velocity $\boldsymbol{e}$ (green) from the root to each obstacle.

Tangent Evaluation

Finally, after propagating the velocity to the obstacle o, the pseudo tangent is evaluated. For trees-of-stars, we want to enforce the pseudo tangent to be parallel to the surface at all times, hence we adopt (7.5) as follows:

$$
\begin{aligned}
&\text{if} \quad \frac{\langle -\boldsymbol{n}, \boldsymbol{f} \rangle}{\| -\boldsymbol{n} \| \, \| \boldsymbol{f} \|} > \cos(R^e) : \\
&\qquad \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}_o) = \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) + b\left(\boldsymbol{k}\left(-\boldsymbol{n}, \boldsymbol{e}_p\right) - \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})\right) \\
&\qquad \text{such that} \quad \| \boldsymbol{k}\left(-\boldsymbol{n}, \boldsymbol{e}_p\right) \| = R^e, \quad b \in \mathbb{R}_{>0} \\
&\text{otherwise} \\
&\qquad \boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e}_o) = \boldsymbol{k}(\boldsymbol{n}, -\boldsymbol{r}) + b\left(\boldsymbol{k}\left(\boldsymbol{n}, \boldsymbol{e}_p\right) - \boldsymbol{k}(\boldsymbol{n}, -\boldsymbol{r})\right) \\
&\qquad \text{such that} \quad \| \boldsymbol{k}\left(\boldsymbol{n}, \boldsymbol{e}_p\right) \| = \pi - R^e, \quad b \in \mathbb{R}_{>0}
\end{aligned}
\tag{7.45}
$$

Hiding Weights

An obstacle o which is occluded by its parent $p(o)$ should not influence the avoidance velocity. For this reason, we introduce the *hiding weight* which reaches zero at full occlusion:

$$w_o^h = \begin{cases} 1 & \text{if } \Gamma(\boldsymbol{s}_o) > 1 \\ \Gamma(\boldsymbol{s}_o)^{\frac{1}{1-b}} & \text{if } b < 1 \\ 0 & \text{otherwise} \end{cases} \qquad b = \frac{\langle \boldsymbol{\xi} - \boldsymbol{\xi}_o^r, \boldsymbol{\xi}_{p(o)}^r - \boldsymbol{\xi}_o^r \rangle}{\| \boldsymbol{\xi} - \boldsymbol{\xi}_o^r \| \, \| \boldsymbol{\xi}_{p(o)}^r - \boldsymbol{\xi}_o^r \|} \tag{7.46}$$

Lemma 7.5.1. *Let us assume a tree with obstacle components $o = 1..N^{\text{com}}$ and the corresponding reference points $\boldsymbol{\xi}_o^r$, which all lie within obstacle o and the corresponding parent p, i.e., $\boldsymbol{\xi}_o^r \in \mathcal{X}_o^i \cap \mathcal{X}_{p(o)}^i$. Conversely, the reference of each parent lies outside of the obstacle, i.e., $\boldsymbol{\xi}_{p(o)}^r \in \mathcal{X}_{p(o)}^i \setminus \mathcal{X}_o^i$. Moreover, the direction from obstacle to parent is never opposing the direction from the parent to the grandparent as described in* (7.44)*. Let us assume the dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ are locally straight in the surrounding of the obstacle according to Definition 3.2.2. The vector field $\dot{\boldsymbol{\xi}}$ obtained through the propagation of the velocity $\boldsymbol{f}(\boldsymbol{\xi})$ as described in* (7.43)*, with the rotation of the final velocity given by* (7.45)*, and vector tree summing using the weights w_o^h given in* (7.46) *ensures collision avoidance according to the boundary condition* (7.1) *with the absence of local minima in free-space.*

Proof. As shown in Lemma 7.2.1, the tangent is defined everywhere as long as the reference direction $\boldsymbol{r}_o(\boldsymbol{\xi})$ is not parallel to the propagated velocity $\boldsymbol{f}_o(\boldsymbol{\xi})$. Moreover, smoothness is ensured due to the adaptable rotation weight $\lambda(\boldsymbol{\xi})$ as shown in Theorem 7.2.3.

Furthermore, the rotation defined in (7.43) is smoothly defined for obstacle-parent-pair, due to the fact that the reference point of the parent is required to lay outside of the child obstacles, and the parent-opposing inequality from (7.44).

This is also ensured for the last surface point s_o, i.e., the projection of the position on the obstacle's surface since the hiding weight w_o^h goes to zero if the two vectors are opposing:

$$\frac{\langle \boldsymbol{s}_c - \boldsymbol{\xi}_c^r, \boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r \rangle}{\| \boldsymbol{s}_c - \boldsymbol{\xi}_c^r \| \, \| \boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r \|} \;\Rightarrow\; b = 1 \;\Rightarrow\; w^h = 0 \tag{7.47}$$

Hence, the corresponding tangent and the vector can be omitted.

Since all weights and corresponding vectors are smoothly defined, according to Lemma 4.2.1 the vector tree evaluation leads to continuous and minima-free dynamics.

Furthermore, from Lemma 7.4.1 we know that the impenetrability of the obstacles is preserved as a single component dominates when approaching the corresponding surface. □

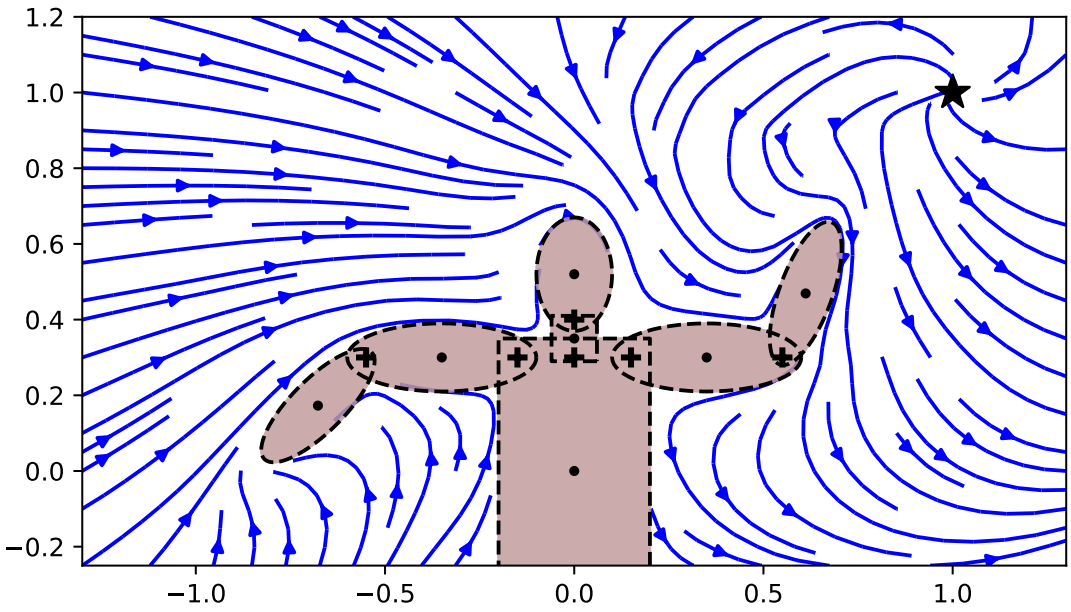


Figure 7.13: Obstacle avoidance of a two-dimensional human with a total of seven sub-obstacles (corresponding reference points as black crosses) of a circular motion with a single stationary point (black star).

7.5.2 Convergence Sequence

The obstacle avoidance algorithm is dependent on locally straight dynamics on the surface of the tree-of-obstacles. Since the general shapes described can be encapsulating directional singularity points, such as attractors, the design of convergence direction requires special care. The computation has to ensure smoothness when getting closer to the directional singularity point $\boldsymbol{\xi}^a$. It is detailed in Algorithm 5.

7.5.3 Mapping Weight Normalization

A mapping weight w_o^m as introduced in (7.48) approaching one indicates that the $\boldsymbol{\xi}$ is on the surface of the corresponding obstacle o. To ensure that this weight remains high while taking into account other obstacles, the normalized weights are defined as follows:

$$w_o^n = \begin{cases} \hat{w}_o^n / \sum_i^{N^{\text{com}}} \hat{w}_i^n & \text{if } \sum_i^{N^{\text{com}}} \hat{w}_i^n > 1 \\ \hat{w}_o^n & \text{otherwise} \end{cases}, \quad \hat{w}_o^n = 1/(1 - w_o) \tag{7.48}$$

for all obstacle $o = 1...N^{\text{obs}}$.

Algorithm 5 Convergence Direction for Tree-of-Obstacles. The abbreviation pred. refers to the predecessor of the direction tree.

Input: $\boldsymbol{f}(\boldsymbol{\xi})$, N^{com} obstacle-components
Output: $\boldsymbol{c}(\boldsymbol{\xi})$ Locally straight dynamics

$w_1^m(\boldsymbol{\xi})$ {Mapping weight for root component (7.30)}
$\boldsymbol{f}(\boldsymbol{\xi}), (\boldsymbol{\xi} - \boldsymbol{\xi}^a) \Rightarrow \boldsymbol{v}^{r,i}(\cdot), \beta_i$ {Initial rotation (4.15)}
$\boldsymbol{v}^{r,c}(\cdot), \beta_c \leftarrow \boldsymbol{f}(\boldsymbol{\xi}^r), (\boldsymbol{\xi}^r - \boldsymbol{\xi}^a)$ {Convergence rot. (4.15)}
$\boldsymbol{v}^{r,a}(\cdot), \beta_a \leftarrow (\boldsymbol{\xi} - \boldsymbol{\xi}^a), (\boldsymbol{\xi}^r - \boldsymbol{\xi}^a)$ {Get rotation (4.15)}
$\boldsymbol{t}_1^r(\cdot) \leftarrow \boldsymbol{v}^{r,i}(\cdot), \boldsymbol{v}^{r,a}(\cdot), \boldsymbol{v}^{r,c}(\cdot)$ {Create rotation tree}
{Tree reduction with Algo. 3 using vector-weight pairs:} $\boldsymbol{t}_1^r\left([\boldsymbol{f}(\boldsymbol{\xi}^r), w_1^m], [\boldsymbol{f}(\boldsymbol{\xi}), (1 - w_1^m)]\right) \Rightarrow \boldsymbol{t}^r(\cdot)$
for $n = 1$ **to** $N^{\text{com}}, i \neq r$ **do** {Iteration over components}
  $c \leftarrow n$ {Set initial node}
  $w_i^m(\boldsymbol{\xi})$ {Mapping weight (7.30)}
  for $l = l(n)$ **to** 0 **do** {From node to root}
    $(\boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r) \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree with pred. c}
    $c \leftarrow p(c)$ {Set current node iterator to its parent}
  end for
  $\boldsymbol{v}^{r,c}(\cdot) \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree with pred. $c = r$}
end for
$w_o^n \leftarrow w_o^m$ {Weight normalization (7.48)}
$\boldsymbol{c}(\boldsymbol{\xi}) \leftarrow \boldsymbol{t}^r(w_o^n)$ {Tree reduction, Algo. 1}

7.5.4 Practical Considerations

As for each obstacle, the algorithm needs to propagate the whole obstacle tree. The presence of multiple obstacles and long trees can lead to many computations. Therefore, it is advised to adapt the distance function Γ_o to approach infinity after a certain distance for outer leaves, as proposed in (3.12). However, the distance function of the root Γ_r should decrease slower, i.e.,

$$\left\| \frac{d}{d\boldsymbol{\xi}} \Gamma_o(\boldsymbol{\xi}) \right\| \geq \left\| \frac{d}{d\boldsymbol{\xi}} \Gamma_r(\boldsymbol{\xi}) \right\| \tag{7.49}$$

This has the effect that the algorithm only considers the obstacle core far away, while when approaching the obstacle, the more detailed leaf structure is considered. This leads to a sparse evaluation tree and enables real-time applicability.

7.6 Evaluation

7.6.1 Computational Cost

The most computationally intensive part of the ROAM algorithm is the matrix-vector multiplication in N dimensions for the stereographic projections and unfolding mappings. Therefore, the algorithm's complexity, given O obstacles, K components, and an obstacle tree of level L, can be

expressed as $\mathcal{O}(N^2OKL)$.

7.6.2 Obstacle Avoidance While Following a Stable Limit Cycle

Setup

We compare the three algorithms MuMo (see Chapter 5), VF-CAPF (Yao, B. Lin, et al., 2022), and ROAM (proposed approach). The chosen scenario uses initial dynamics and the obstacle distribution as proposed in (Yao, B. Lin, et al., 2022). The initial dynamics represent a circular limit cycle of the form:

$$\boldsymbol{f}(\boldsymbol{\xi}) = \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + 2(R_0 - \|\boldsymbol{\xi}\|)\boldsymbol{I} \right) \boldsymbol{\xi} \tag{7.50}$$

where the circle radius is $R_0 = 2$, and $\boldsymbol{I}$ is the two-dimensional identity matrix. The environment contains six convex obstacles, and the agent is aware of their location at all times (see Fig. 7.14). A grid of 10x10 evenly distributed starting points was constructed, of which 93 were in free space. The trajectory is evaluated for these starting points through Euler integration with a time step $dt = 0.01s$ and a maximum of 500 iterations.[2]

Metrics

The trajectories are compared by observing the local minima, the distance to the desired limit cycle, and the similarity between the velocity after obstacle avoidance and the initial velocity. Furthermore, we look at the evolution of the velocity over time, i.e., the discrete acceleration.

The root mean square error (RMSE), i.e., $\text{RMSE} = \sum_{m=1}^{M} \|\boldsymbol{v}_i - \boldsymbol{u}_i\|$ is used, as well as the normalized inverted cosine similarity (NICS)

$$\text{NICS} = \frac{1}{2}\left(1 - \frac{1}{M}\sum_{m=1}^{M} \frac{\langle \boldsymbol{v}_i, \boldsymbol{u}_i \rangle}{\|\boldsymbol{v}_i\| \, \|\boldsymbol{u}_i\|}\right) \tag{7.51}$$

with $\text{NICS} \in [0, 1]$. Note that the smaller NICS, the higher the similarity of the two vectors.

Results

In Table 7.2, it can be observed that MuMo is the only method among the three that results in trajectories ending up in local minima on the surface of an obstacle in almost half of the cases. These local minima lie on the limit cycle, suggesting that by increasing the step number, all MuMo trajectories would eventually converge to these local minima. Due to this behavior, MuMo is deemed inappropriate for the proposed scenario. Its limited applicability to nonlinear initial dynamics as pointed out in Table 7.1.

[2] Source code on https://github.com/hubernikus/nonlinear_obstacle_avoidance.git, 2022/02/31

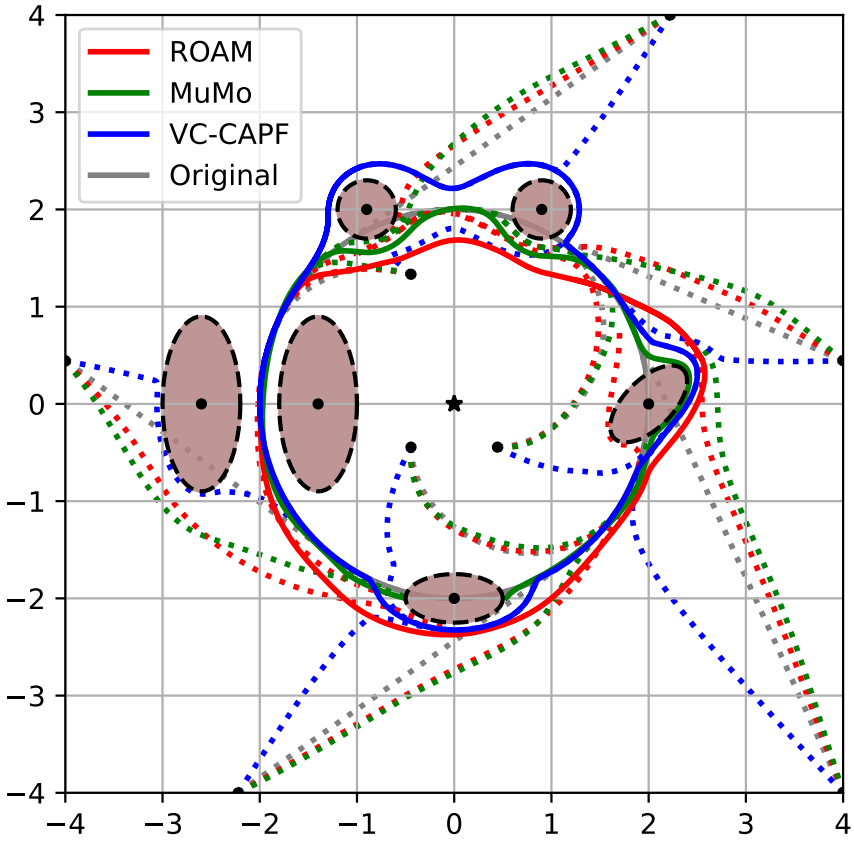


Figure 7.14: MuMo (see Chapter 3), VF-CAPF (Yao, B. Lin, et al., 2022) and ROAM are used to guide a nonlinear, limit-cycle-following vector field around six concave obstacles. The resulting limit cycles are visualized as a solid line in the respective color. Additionally, the trajectories from specific starting points (black circles) are visualized with a dashed line.

The focus of the comparison is thus between ROAM and VF-CAPF. ROAM demonstrates better path following, with trajectories maintaining a shorter distance to the reference circle throughout the motion. Moreover, ROAM exhibits lower acceleration along the path, indicating smoother motion with fewer abrupt changes. Additionally, ROAM shows higher similarity to the initial dynamics compared to VF-CAPF.

These findings can be qualitatively observed in Figure 7.14, where VF-CAPF closely follows the initial path (gray) outside the obstacle region and deviates only when in close proximity to the obstacle. The metrics indicate that this behavior leads to higher accelerations and more significant deviations from the initial trajectory overall. On the other hand, MuMO moves more directly towards the limit cycle compared to ROAM, but at the expense of creating local minima on the surface of the bottom obstacle.

	ROAM (proposed)	MuMo	VF-CAPF	Original dynamics
N^m	**0%**	48%	**0%**	0%
RMSE($\boldsymbol{\xi}, R_0$)	1.46 ± 1.11	**1.24 ± 1.15**	1.48 ± 0.87	0.93 ± 0.96
RMSE($\dot{\boldsymbol{\xi}}, \boldsymbol{f}_0$)	0.15 ± 0.07	**0.04 ± 0.09**	0.47 ± 0.43	0.00 ± 0.00
NICS($\dot{\boldsymbol{\xi}}, \boldsymbol{f}_0$)	0.04 ± 0.02	**0.01 ± 0.02**	0.12 ± 0.11	0.00 ± 0.00
RMSE($\dot{\boldsymbol{\xi}}_t, \dot{\boldsymbol{\xi}}_{t+1}$)	2.51 ± 2.66	**2.04 ± 5.98**	3.47 ± 2.14	0.27 ± 0.17
NICS($\dot{\boldsymbol{\xi}}_t, \dot{\boldsymbol{\xi}}_{t+1}$)[$10^{-4}$]	0.63 ± 0.66	**0.51 ± 1.49**	1.08 ± 0.82	0.07 ± 0.04

Table 7.2: The different trajectories are compared in (1) the ratio of trajectories which end up in a local minimum on the obstacle N^m, (2) the distance to the desired trajectory, i.e., the difference to radius R_0. Furthermore, (3) RMSE and NICS between all approaches to the original DS are evaluated, as well as (4) the change of the dynamics over time (corresponds to acceleration). The mean and standard deviation are evaluated over the 93 trajectories.

7.6.3 Nonlinear Path Following with Autonomous Wheelchair

Experimental Setup

In the second evaluation, the autonomous wheelchair QOLO was employed, and various initial vector fields combined with avoidance methods were tested. The control point of the wheelchair was positioned in front of the wheel axis to ensure effective maneuverability. To account for the shape and size of the QOLO wheelchair, a margin of 0.7 meters was added to the obstacles during the evaluation.

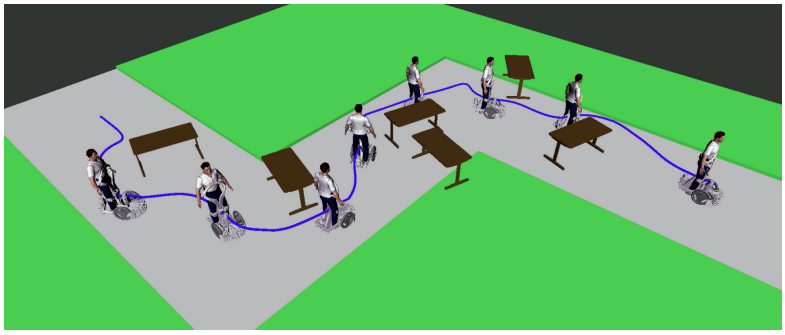
The path followed by the wheelchair consisted of three road segments, denoted as $s = 1$, the segment closest to the goal, up to $s = 3$. Although there was no strict constraint to remain within the road boundaries, the initial dynamics were specifically designed to guide the wheelchair towards the center of the road, promoting adherence to the desired path (see Fig. 7.15a).

Seven tables were randomly placed along the path, with their centers positioned on the path itself. The tables were positioned away from the starting point and the attractor. Additionally, no more than two tables intersected, including the margin, allowing all table shapes to be represented as star-shapes, as required by MuMo.

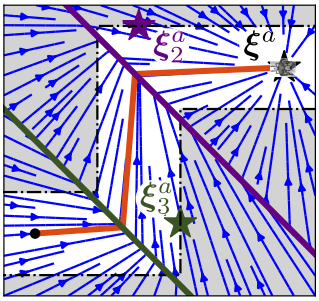
Navigation Algorithms

Three approaches are used to navigate in this environment.

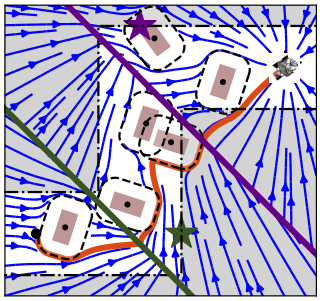
Local straight dynamics (3.3) are combined with MuMo. The initial dynamics consist of three separate vector fields with corresponding local attractors (star), see Fig. 7.15b. The attractor switches when transitioning from one region to the next (crossing the line).



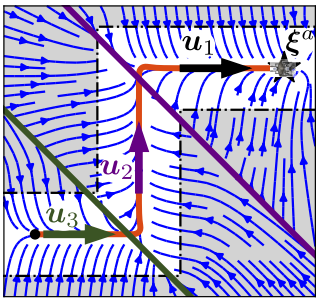
(a) QOLO-robot navigating along a wavy road using global PF



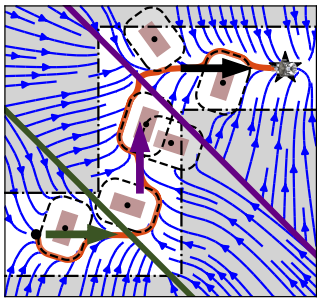
(b) Local straight - initial



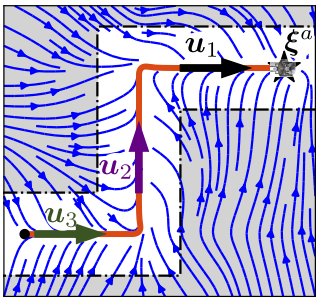
(c) Local straight - avoiding



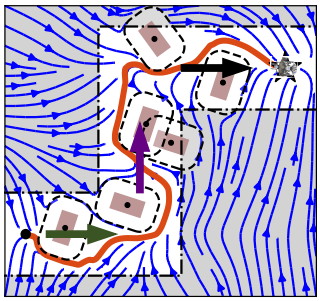
(d) Local PF - initial



(e) Local PF - avoiding



(f) Global PF - initial



(g) Global PF - avoiding

Figure 7.15: The robot is navigating between static tables on a wavy road (gray) through a grass field (green). The initial dynamics and the reference trajectory (orange) are on the left, with the corresponding obstacle dynamics on the right. The local attractors (colored starts) and switching regions (colored lines) are used to create global dynamics (a, c). The global path following (f) uses a single attractor only.

Local PF (path following) is combined with ROAM. The local PF dynamics are given as:

$$\boldsymbol{f}_s(\boldsymbol{\xi}) = \boldsymbol{u}_s + \langle \Delta\boldsymbol{\xi}, \boldsymbol{u}_s \rangle \, \boldsymbol{u}_s - \Delta\boldsymbol{\xi} \quad \text{with} \quad \Delta\boldsymbol{\xi} = (\boldsymbol{\xi} - \boldsymbol{\xi}_s^a) \tag{7.52}$$

for all segments $s = 1..3$, where $\boldsymbol{u}_s \in \mathbb{R}^2$ is the (local) nominal direction pointing along the road segment, and $\boldsymbol{\xi}_s^a \in \mathbb{R}^2$ is the local attractor.

Global PF dynamics is evaluated by using directional-tree averaging as described in Appendix 4.2. The root of the direction-tree is given as $\boldsymbol{v}_{0,1} = \boldsymbol{\xi} - \boldsymbol{\xi}^a$. The direction tree is populated iteratively:

- $\boldsymbol{v}_{s,1}(\boldsymbol{\xi}) = \boldsymbol{u}_s$ with respective parent direction $\boldsymbol{v}_{s-1,1}$

- $\boldsymbol{v}_{s,2}(\boldsymbol{\xi}) = \boldsymbol{f}_s(\boldsymbol{\xi})$ with respective parent direction $\boldsymbol{v}_{s,1}$

for all segments $s = 1..3$. The final dynamics are obtained through the weighted evaluation described Algorithm 1, using the following weights

$$w_{s,1}(\boldsymbol{\xi}) = 0, \;\; w_{s,2}(\boldsymbol{\xi}) = \frac{1}{d_s}(1 + \min(\langle \boldsymbol{v}_s, \boldsymbol{\xi}_s^a - \boldsymbol{\xi} \rangle, 0)) \tag{7.53}$$

where $d_s \in \mathbb{R}_{\geq 0}$ the distance to the line segment s. The segment weights $w_s(\boldsymbol{\xi}) \in [0, 1]$ are normalized if their sum exceeds one.

Results

Combining the global PF with ROAM ensures the convergence of all trajectories, as shown in Table 7.3. The other two methods achieve a convergence rate of approximately 92%. This can be attributed to using a high-level planner, specifically switching between the local dynamics. Since this conflicts with the guarantees of absences of local minima. While more sophisticated switching or transitioning methods may exist, to the best of our knowledge, there is no global path sequencer that can guarantee these convergence properties within a finite time. Furthermore, when using the local potential field (PF) with ROAM, the robot spends less time on the desired path. It has a greater average distance to the path boundaries compared to the local straight algorithm combined with MuMo. However, the average distance traveled remains approximately the same across all methods.

	Local straight	Local PF	Global PF
Converged []	92%	92%	**100%**
Off-track [%]	1.49 ± 0.00	**0.79 ± 0.00**	1.89 ± 0.00
Δd [m]	1.54 ± 0.04	**1.73 ± 0.02**	1.65 ± 0.04
Distance [m]	**20.2 ± 3.4**	21.0 ± 3.8	20.8 ± 2.8

Table 7.3: The three approaches for following the local path are compared based on the following metrics: the convergence ratio to the attractor, the ratio of trajectories deviating from the path, the distance to the road border Δd, and the total length of the trajectory. The reported values represent the mean and standard deviation calculated from 100 runs with randomly distributed furniture while keeping the start and endpoints consistent. The local straight approach use the obstacle avoidance as proposed in Chapter 5, while the other methods use the approach developed here.

7.6.4 Obstacle Avoidance in Three Dimensions

Spiraling Motion Around Human in Simulation

Inspired by (7.50), we propose spiraling dynamics as:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{\xi}) &= \boldsymbol{B}^T \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + 2\left(R_0 - \|\boldsymbol{B}\tilde{\boldsymbol{\xi}}\|\right) \boldsymbol{I} \right) \boldsymbol{B}\tilde{\boldsymbol{\xi}} + \boldsymbol{p}(\boldsymbol{\xi}) \\ \text{with } \boldsymbol{B} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{p}(\boldsymbol{\xi}) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \end{aligned} \tag{7.54}$$

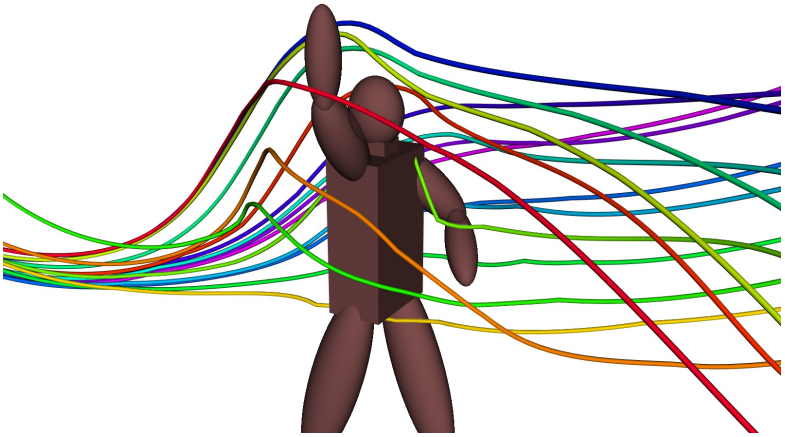
with the spiraling radius of $R_0 = 0.1$ m and $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}^a$ the relative position with respect to the center $\boldsymbol{\xi}^a$.

The obstacle tree representing the human in ROAM consists of components corresponding to the limbs and main body, with the core serving as the tree's root. When applying ROAM for collision avoidance, see Figure 7.16, it can be observed that all trajectories successfully avoid the human without becoming trapped in local minima. Furthermore, the system's dynamics return to the initial state, far away from the obstacle, both at the beginning and after successfully avoiding the collision. This behavior highlights the effectiveness of ROAM in generating smooth and convergent trajectories while maintaining the desired dynamics of the system.

Qualitative Evaluation on Robot Arm

Experiments were performed using the 7 DoF Panda robot by Franka Emika on a fixed base (see Fig. 7.17). The scenario chosen is the automated disinfecting of a running conveyor belt, which transports various parcels. The initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ are similar to the spiral motion in (7.54),

(a) Initial spiraling motion $\boldsymbol{f}(\boldsymbol{\xi})$



(b) Rotated motion $\dot{\boldsymbol{\xi}}$ around human

Figure 7.16: ROAM guides trajectories from 16 different initial positions and ensures that all trajectories successfully avoid the static human in simulation.

but the basis $\boldsymbol{B}$ and perpendicular dynamics $\boldsymbol{p}(\boldsymbol{\xi})$ given by:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{p}(\boldsymbol{\xi}) = \begin{bmatrix} 0 & 0 & (\boldsymbol{\xi}^a(t) - \boldsymbol{\xi}) \end{bmatrix}^T \tag{7.55}$$

Moreover, the attractor is dynamic and moves back and forth the conveyor belt:

$$\boldsymbol{\xi}^a(t) = \begin{bmatrix} 0.5 & 0.6 + 0.1 \sin\left(\frac{\pi}{10} t\right) & 0.3 \end{bmatrix}^T \tag{7.56}$$

As a result, the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi}, t)$ are time-varying, too.

It is assumed that the robot has complete knowledge of the relative position and shape of the conveyor belt. The position and velocity of the parcel are determined using reflective markers (Optitrack). The analytical shape of the objects is known analytically. Additionally, an operator

handles parcels on the conveyor belt, but the robot is not aware of its presence. However, the robot can adapt to physical interactions since an impedance controller is employed (Kronander and Billard, 2015).

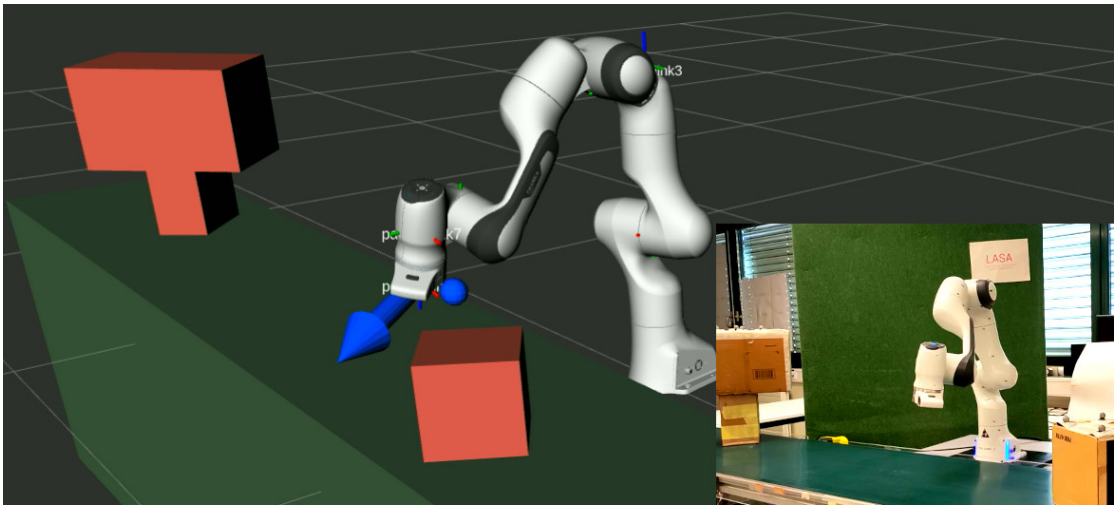
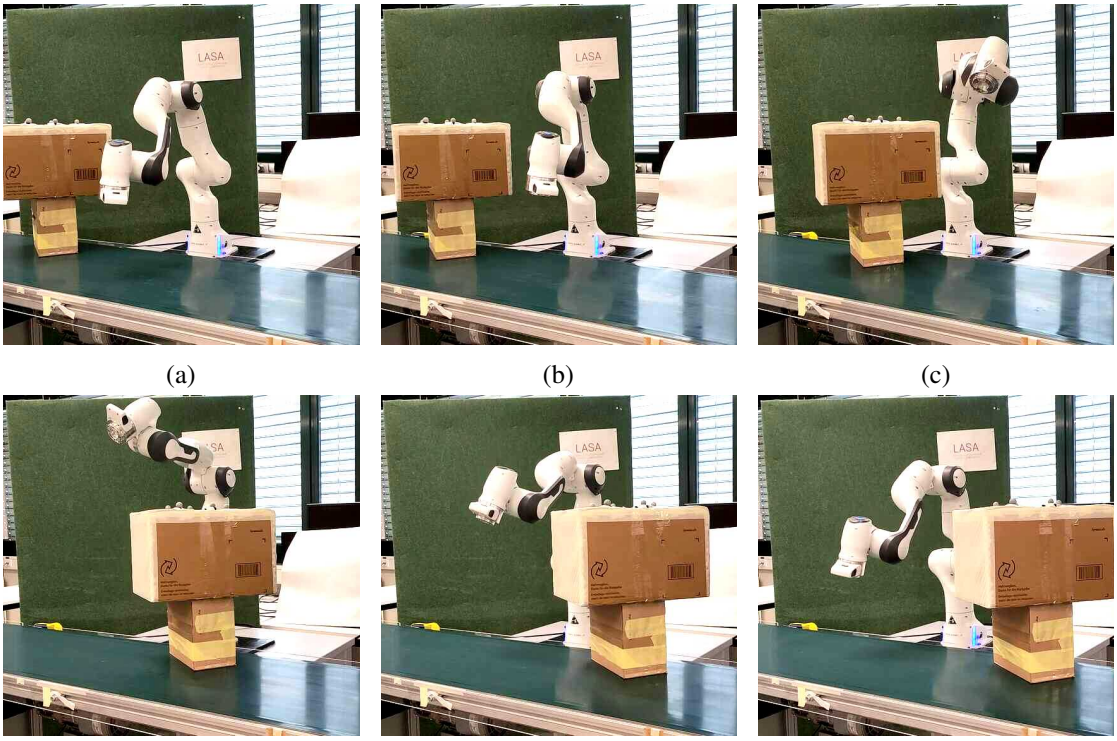


Figure 7.17: The robot is aware of the two obstacles (brown shapes) as well as the conveyor belt (green block) to obtain the avoidance dynamics (blue arrow). The center of the initial dynamics (blue dot) is moving across the conveyor belt.

The robot successfully avoids both the parcels and the conveyor belt while maintaining adherence to the initial dynamics whenever feasible. By utilizing trees-of-stars to represent the concave obstacle and positioning the reference point of the root component on the conveyor belt, the robot effectively avoids the parcel from above (see Fig. 7.18). Notably, comparable methods such as MuMo , see Chapter 5, or VF-CAPF (Yao, B. Lin, et al., 2022) do theoretically not permit the placement of a reference point in trees of obstacles that facilitates collision avoidance in such environments.

7.7 Discussion

The proposed rotational obstacle avoidance method (ROAM) has successfully addressed the challenge of avoiding collisions with initially nonlinear dynamics around general concave obstacles without holes. To the best of our knowledge, it is the first state-dependent solution for trees-of-stars obstacles free from local minima regardless of hyperparameter choice. Furthermore, ROAM enables obstacle avoidance while attempting to maintain nonlinear dynamics, which is a significant advantage. The algorithm has demonstrated improved convergence and motion similarity compared to the baseline methods in experimental evaluations. Moreover, its low computational cost has allowed its application to dynamic obstacle avoidance scenarios involving a robotic arm.



(a) (b) (c)

(d) (e) (f)

Figure 7.18: The 7 DoF robot arm adeptly avoids the star-shaped parcel that is being transported on the conveyor belt.

7.7.1 Stationary Points

ROAM introduces a new stationary point for each obstacle (or obstacle tree). However, due to the topological properties of smooth vector fields, at least one fixed point is created for each obstacle (a hole in space). These fixed points are observed to be saddle points and the probability of reaching them is effectively zero. Additionally, any noise or perturbation in the system pushes the motion away from these unstable points. It is worth noting that while ROAM is used for dynamic scenarios, the trajectories of these saddle points should be preserved as they reflect the smoothness of the velocity. Smoothness is crucial as it ensures that even with uncertainties in perception or unexpected disturbance, there are no discontinuity in the desired velocity command $\dot{\boldsymbol{\xi}}$. Nevertheless, the removal of saddle points could be achieved by setting the smoothness factor q to 1 in (7.5) and selecting any tangent direction for $\boldsymbol{e}(\boldsymbol{\xi})$ when $\boldsymbol{c}(\boldsymbol{\xi})$ and $\boldsymbol{r}(\boldsymbol{\xi})$ are collinear.

7.7.2 Trees of Stars

ROAM relies on a star-shaped (or trees-of-stars) obstacle environment. While in certain real-world scenarios, such division can be achieved based on the rigid-body features of the surroundings (e.g., dividing a human into limbs and core or a table into plate and legs), it is often challenging

to determine such divisions for more complex obstacles or in higher-dimensional spaces, such as joint space collision avoidance. Some propositions for extracting star-shaped environments exist (Dahlin and Karayiannidis, 2023a); however, further research is needed to extend these approaches for real-time applications.

However, algorithms exist to simplify general shapes by approximating them as a union of overlapping spheres (Hubbard, 1996), which could serve as the basis for constructing the obstacle tree in future work. Additionally, using a circular shape reduces the computational complexity of tasks such as checking for component intersections and evaluating the normal direction and distance to the obstacles.

7.7.3 Higher Dimensional Applications: Joint Limits

During the experiments with the robotic arm, there were instances where obstacle-free motions led the arm to reach its joint limits or encounter singularities. Future research will investigate a unified framework that combines task-space collision avoidance with joint-space constraint avoidance. ROAM is well-suited for evaluating the desired motion in both representations due to its low computational cost and applicability in higher dimensions.

7.7.4 Application to Robotic Systems

The proposed algorithm assumes that the robot is a point mass. However, it can be extended to real robots by introducing a margin around the obstacles or using multiple control points, as proposed in Chapter 5 . It is important to note that the latter method does not guarantee full convergence in general scenarios. Alternative methods employ a full body collision model for task planning involving sampling or optimization (Mainprice et al., 2020; Koptev, Figueroa, and Billard, 2022). However, they cannot guarantee finding a feasible path in a finite time. Therefore, future work should focus on extending ROAM to handle analytic avoidance scenarios where trees of stars represent both the agent and the obstacle.

7.7.5 Tangent Following on the Surface

Each tree of stars has exactly two saddle points, which lie on the convergence direction line passing through the tree's root. At any other surface position, the velocity is tangent to the surface. This can lead to extensive wall-following behaviors for obstacles with multiple levels. Moreover, for obstacle-tree with high concavities, this can lead to a fast change of the desired velocity. As on each side of the concavity, the vector field might point in the opposite direction. While theoretically still smooth, in practice this can lead to jittery behavior. To address this, future work should focus on optimizing the selection of saddle points and allowing the rotated velocity to deviate from the obstacle's tangent plane. However, this would require a combination of ROAM with high-level planning.

7.7.6 Region of Influence during Task Obstruction

In the experiment on the robot arm, the size of the limit cycle radius was small compared to the obstacle. Hence, a small region of influence is chosen, in which the obstacle affects the environmental dynamics. As the obstacle passes through the center of the limit cycle, the robot is guided by the global vector field which tries to approach the limit cycle, additionally ROAM repulses from the obstacle. The algorithm ensures a smooth and minima-free transition between these opposing dynamics. Yet, integrated into the control loop such regions of high directional change can exert jerky behavior, as observed during the experiment. This limitation opens up extensions of ROAM in future work which are twofold. On the one hand, the region of influence of the obstacles should be adapted based on the local curvature of space. As such, in regions with fast change, the transition between two opposing dynamics should be slower and smoother. On the other hand, a single, temporarily smooth, vector field cannot represent all the intricacies of a dynamically changing environment. Imagine, an operator trying to wipe the conveyor belt at all costs as obstacles pass. This is expected to lead to jerky behavior. However, as humans, we use a logic similar to a state machine. The global task is chosen based on the environment, hence there is a switching between polishing the conveyor belt and waiting for passing obstacles. Even though ROAM exerts remarkable adapting capabilities, for applications it should be coupled with a high-level logical planner.

8 Obstacle Aware Passive Control for Dynamical Systems

Publication Note

The material in this chapter is adapted from:
Huber, L., T. Trinca, J.-J. Slotine, and A. Billard (2023). “Passive Obstacle Aware Control to Follow Desired Velocities”. In: *IEEE Robotics and Automation Letters* (submitted).

Authorship Note: The algorithm presented in this Chapter was first investigated by Thibaud Trinca during a semester project supervised by Lukas Huber. Lukas Huber then iterated on the algorithm, developed the theoretical proofs, and performed the experimental evaluation on the robot arm.
Source Code:

- Algorithm: https://github.com/hubernikus/obstacle_aware_damping.git
- Implementation on robot arm:
  https://github.com/hubernikus/franka_obstacle_avoidance.git

Multimedia: Supplementary video can be found on:

https://youtu.be/WKso-wu68v8

Evaluating and updating the obstacle avoidance velocity for an autonomous robot in real-time ensures robustness against noise and disturbances. A passive damping controller can be used to obtain the desired motion with a torque-controlled robot, which remains compliant and ensures a safe response to external perturbations. This chapter proposes a novel approach for designing the passive control policy. Our algorithm complies with obstacle-free zones while transitioning to increased damping near obstacles to ensure collision avoidance. This approach ensures stability across diverse scenarios, effectively mitigating disturbances. Validation on a 7DoF robot arm demonstrates superior collision rejection capabilities compared to the baseline, underlining its practicality for real-world applications. Our obstacle-aware damping controller represents a substantial advancement in secure robot control within complex and uncertain environments.

8.1 Introduction

Robots operating in unstructured, dynamic environments must balance between adapting the path to the surrounding and following a desired motion. In human-robot collaboration, they must efficiently comple their tasks while ensuring safe copmliance in the presence of potential collisions.

In complex and dynamic environments, velocity adjustments based on real-time sensory information are essential. Achieving this necessitates algorithms that are easily configurable and capable of rapid evaluation. As such, closed-form control laws eliminate the need for frequent replanning. Dynamical Systems (DS) are a valuable framework for representing such desired motion L. Huber, Billard, and J.-J. Slotine, 2019. Where the behavior of the desired first order DS is approximated through suitable controllers.

Most widely used robotic systems consist of rigid materials. Consequently, interaction of these robotic systems with the surrounding leads to abrupt energy transfers, posing the risk of damage to the robot and its environment. However, modern robotic platforms equipped with force and torque sensing capabilities which enable precise control over the forces exerted by the robot. However, integrating these sensors make the feedback-controller more complex. The robot must achieve its designated position, by follwoing a desired velocity profile while remaining compliant with interaction forces. The control problem of balancing position, velocity, and force constraints, is addressed by *impedance controllers* (Takegaki and Arimoto, 1981; Hogan, 1984) .

Obstacle avoidance is fundamental to motion control, with reactive approaches capable of handling dynamic and intricate environments (L. Huber, Billard, and J.-J. Slotine, 2019). The controller should remain compliant in free space while adhering to the desired motion when getting close to the obstacle. Furthermore, when encountering surfaces like fragile glass on a table, the controller must adopt stiffness to prevent a collision, yet it should be compliant when interacting with an operator (Fig. 8.1).

This work introduces a novel approach incorporating dynamic obstacle avoidance using DS and

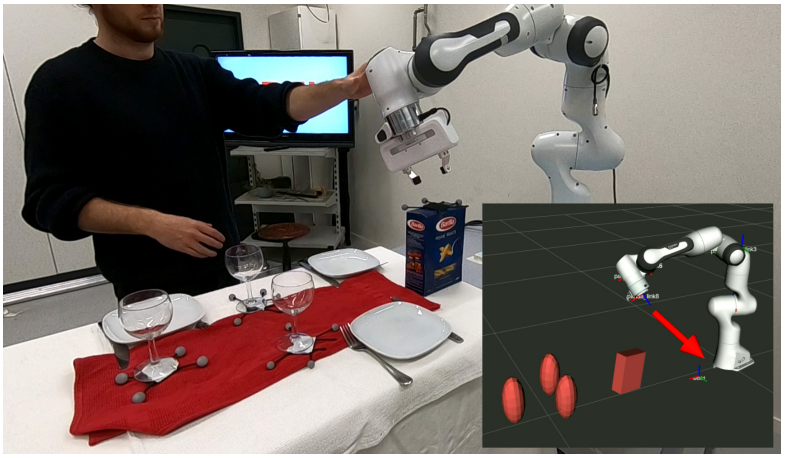


Figure 8.1: The proposed passive obstacle-aware controller lets the robot absorb external disturbances while ensuring collision avoidance. While tipping over the closed pasta box on this dinner table setup might be acceptable. Yet, the delicate wine glasses demand careful handling to prevent breakage.

variable impedance control, enhancing adaptability, reactivity, and safety in robotic movements. It empowers robots to navigate complex environments, proactively avoiding collisions and rejecting disturbances. Our approach is evaluated through simulations and an implementation on a 7-degree-of-freedom (7DoF) robot arm, demonstrating robust and safe control in real-world scenarios.

8.1.1 Contribution

We introduce a passive controller that incorporates into the feedback control loop as visualized in Figure 8.2. In this work, we make the following contributions:

- A novel obstacle-aware passive controller (Sec. 8.2)
- A passivity guarantee (without storage tank) which applies to general damping controllers (Theorem 8.2.2)
- Discrete-time analysis to enable control parameter design which ensures stability (Section 8.3)
- Implementation and testing on 7DoF robot arm (Sec. 8.4)

8.1.2 Problem Statement

Following assumptions are made about the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$:

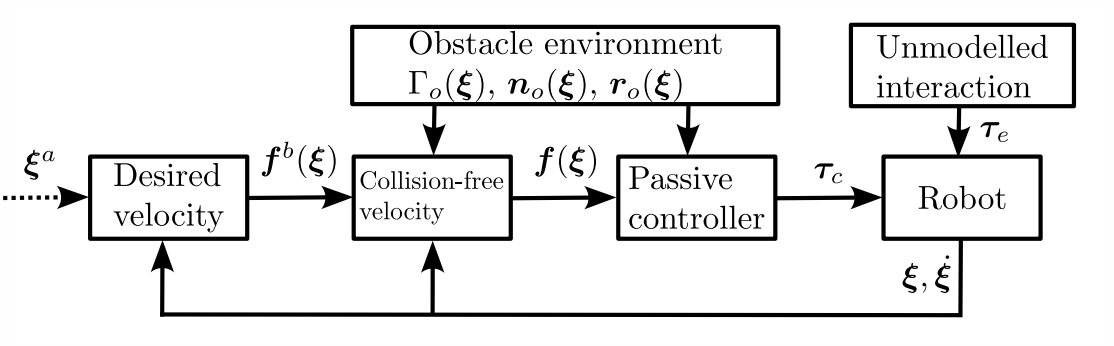


Figure 8.2: The desired velocity $\boldsymbol{f}^b(\boldsymbol{\xi})$ can result from a learned velocity field or pointing towards a desired attractor $\boldsymbol{\xi}^a$. The desired velocity is used to evaluate the obstacle avoidance velocity $\boldsymbol{f}(\boldsymbol{\xi})$, fed into the force controller to obtain the control force $\boldsymbol{\tau}_c$. In order to achieve collision avoidance, the distance function $\Gamma_o(\boldsymbol{\xi})$, the normal direction $\boldsymbol{n}_o(\boldsymbol{\xi})$, and the reference direction $\boldsymbol{r}_o(\boldsymbol{\xi})$ are evaluated for each obstacle $o = 1..N^{\text{obs}}$.

1. $\boldsymbol{f}(\boldsymbol{\xi})$ is continuous for all reachable states.
2. $\boldsymbol{f}(\boldsymbol{\xi})$ is bounded, i.e., there exists a constant $v^{\max} \in \mathbb{R}$ such that $\|\boldsymbol{f}(\boldsymbol{\xi})\| \leq v^{\max} \ \forall \boldsymbol{\xi} \in \mathbb{R}^N$
3. $\boldsymbol{f}(\boldsymbol{\xi})$ leads to a collision-free motion, i.e., $\boldsymbol{n_o}(\boldsymbol{\xi})^T \boldsymbol{f}(\boldsymbol{\xi}) \geq 0$ as $\Gamma_o(\boldsymbol{\xi}) \to 1 \quad \forall o = 1..N^{\text{o}}$ with the normal $\boldsymbol{n}_o$ and distance Γ_o of the o-th obstacle.

Note that velocity obtained using the obstacle avoidance method described in (3.13), fulfills these conditions if base velocity $\boldsymbol{f}^b(\boldsymbol{\xi})$ is continuous and bounded.

Note, that in this Chapter, the desired velocity $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^N$ is expected to be avoiding collisions, and, as such, might be obtained from a vector-field-based collision avoidance algorithm as described in the previous chapters.

Moreover, the velocity $\dot{\boldsymbol{\xi}}\mathbb{R}^N$ describes the actual velocity of the system, specifically the robot agent, and $\ddot{\boldsymbol{\xi}}\mathbb{R}^N$ is the acceleration.

8.2 Obstacle Aware Passivity

We propose a novel controller, which ensures passivity as defined in (3.31) but adapts the damping matrix given in (3.32) based on the desired velocity $\dot{\boldsymbol{\xi}}$ and obstacles in the surrounding. Far away from obstacles, the system is designed to follow the initial velocity, but approaching the obstacle increases the damping, decreasing the chance of a collision.

Hence, the damping matrix $\mathscr{D}(\boldsymbol{\xi}) \in \mathbb{R}^{N \times N}$ smoothly changes from being aligned with the direction of the velocity, as used in (Kronander and Billard, 2015), to be aligned with the averaged normal of the obstacles. The desired damping matrix transitions between velocity preserving and collision

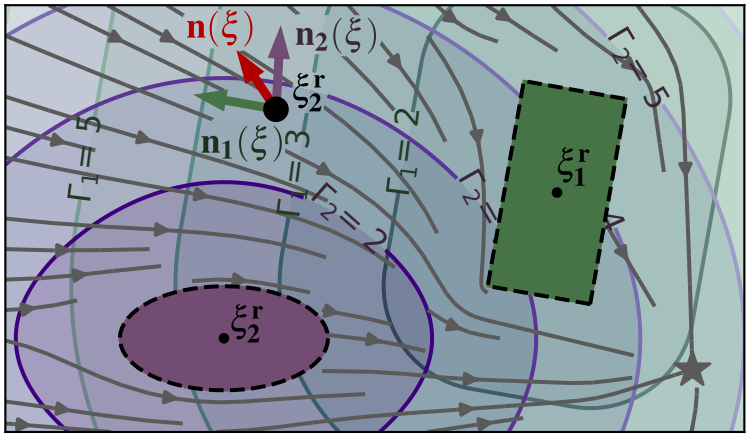


Figure 8.3: The Γ-field is defined individually for each of the obstacles. At each position $\boldsymbol{\xi}$, we can evaluate the surface normal $\boldsymbol{n}(\boldsymbol{\xi})$. The velocity $\boldsymbol{f}(\boldsymbol{\xi})$ (gray) avoids collision with the obstacles and converges towards the attractor (star).

avoidance using a smoothly defined linear combination:

$$\mathscr{D}(\boldsymbol{\xi}) = (1 - w(\boldsymbol{\xi}))\,\mathscr{D}^f(\boldsymbol{\xi}) + w(\boldsymbol{\xi})\mathscr{D}^o(\boldsymbol{\xi}) \tag{8.1}$$

The damping matrix is made up of two components: the velocity damping. $\mathscr{D}^f \in \mathbb{R}^{N\times N}$ which prioritizes following the desired velocity similar to (Kronander and Billard, 2015), and the obstacle damping $\mathscr{D}^o \in \mathbb{R}^{N\times N}$ which is designed to reject disturbances towards obstacles. The two damping matrices are positive definite and are smoothly summed using the danger weight $w(\boldsymbol{\xi}) \in [0,1]$. Far away from obstacles the weight reaches $w(\boldsymbol{\xi}) = 0$, whereas $w(\boldsymbol{\xi}) = 1$ when approaching a boundary:

$$\begin{aligned} w(\boldsymbol{\xi}) &= \max\left(0, \frac{\Gamma^{\text{crit}} - \Gamma(\boldsymbol{\xi})}{\Gamma^{\text{crit}} - 1}\right) \|\boldsymbol{n}(\boldsymbol{\xi})\| \\ \text{with} \quad \Gamma(\boldsymbol{\xi}) &= \min_{o=1..N^{\text{obs}}} \Gamma_o(\boldsymbol{\xi}) \end{aligned} \tag{8.2}$$

The critical distance $\Gamma^{\text{crit}} \in \mathbb{R}_{>0}$ defines the distance where the system has starts to increase the damping towards the obstacle. $\mathscr{D}^f(\boldsymbol{\xi})$ and $\mathscr{D}^{\text{obs}}(\boldsymbol{\xi})$ follow design given in (3.32) and are positive definite matrices, thus $\mathscr{D}(\boldsymbol{\xi})$ is positive definite, too.

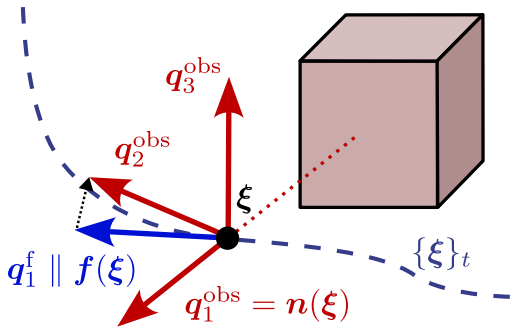


Figure 8.4: The damping matrix enforcing desired velocity following $\mathscr{D}^f$ has the first basis vector q_1^f which points along the avoidance velocity $f(\xi)$. The damping matrix to enforce collision avoidance $\mathscr{D}^{\text{obs}}$ uses the normal $n(\xi)$ to construct the first direction of the decomposition basis q_1^{obs}.

8.2.1 Damping for Collision Repulsion

Normal Direction

The damping matrix $\mathscr{D}^{\text{o}}(\xi)$ rejects velocities in the direction of the obstacles. To allow this, we introduce an averaged normal direction:

$$n(\xi) = \sum_{o=1}^{N^{\text{o}}} n_o(\xi) \frac{1/(\Gamma_o(\xi) - 1)}{\sum_{p=1}^{N^{\text{o}}} 1/(\Gamma_p(\xi) - 1)} \tag{8.3}$$

where the unit normals $n_o(\xi)$ pointing away from the obstacle $o = 1, .., N^{\text{o}}$ and are perpendicular to the surface, see Figure 8.3.

The averaged normal $n(\xi)$ is a weighted linear combination of the obstacles' normals, giving more importance to closer obstacles. Additionally, the averaged normal converges to an obstacle normal as we converge towards it, i.e., $\lim_{\Gamma_o(\xi)\to 1} n(\xi) = n_o(\xi)$. Note that the averaged normal is a zero-vector when two obstacles oppose each other.

Decomposition Matrix

The decomposition matrix $\mathscr{Q}^{\text{o}}(\xi)$ has its first vector aligned with the normal to the obstacle:

$$q_1^{\text{o}}(\xi) = n(\xi)/\|n(\xi)\| \quad \forall \xi : \|n(\xi)\| > 0 \tag{8.4}$$

In the case that $\|n(\xi)\| = 0$, the danger weight $w(\xi)$ from (8.2) reaches 0. Hence, the obstacle-aware damping in (8.1) has no effect and is not evaluated.

The second vector is set to be aligned with the desired velocity as much as possible, allowing

increased velocity following (Fig. 8.4) . However, it has to remain orthonormal to $\boldsymbol{q}_1^{\mathrm{o}}$

$$\boldsymbol{q}_2^{\mathrm{o}} = \frac{\hat{\boldsymbol{q}}_2^{\mathrm{o}}}{\|\hat{\boldsymbol{q}}_2^{\mathrm{o}}\|} \quad \hat{\boldsymbol{q}}_2^{\mathrm{o}} = \boldsymbol{q}_1^{f} - \boldsymbol{q}_1^{\mathrm{o}} p \quad \forall\, \boldsymbol{\xi} : |p| < 1 \tag{8.5}$$

where velocity unit vector $\boldsymbol{q}_1^{f}$ is defined in (3.33), and the object weight is evaluated as $p = \left\langle \boldsymbol{q}_1^{\mathrm{o}}, \boldsymbol{q}_1^{f} \right\rangle$. For the case that $|p| = 1$, the second basis $\boldsymbol{q}_2^{\mathrm{o}}$ is set to be any orthonormal vector. The remaining vectors $\boldsymbol{q}_d^{\mathrm{o}}, d = 3, .., N$ are constructed to form an orthonormal basis to the first two.

Damping Values

We define the values of the diagonal matrix $\mathscr{S}^{\mathrm{o}}(\boldsymbol{\xi})$ as

$$\mathscr{S}_d^{\mathrm{o}}(\boldsymbol{\xi}) = \begin{cases} s^{\mathrm{o}} & d = 1 \\ |p| s^{c} + (1 - |p|) s^{f} & d = 2 \\ s^{c} & d = 3..N \end{cases} \tag{8.6}$$

where the damping along the nominal direction $s^f \in \mathbb{R}_{>0}$, obstacle-damping $s^{\mathrm{o}} \in \mathbb{R}_{>0}$, and the compliant-damping $s^c \in \mathbb{R}_{>0}$ are user-defined parameters which determine the behavior of the passive-controller. The first entry of $\boldsymbol{S}^{\mathrm{o}}$ dictates the damping towards the obstacle, and the second entry the desired velocity following. Note how the second value approaches the compliant damping, as normal and velocity become parallel.

To ensure continuity across time of the control force as defined in (3.31), the values of the diagonal damping matrix $\mathscr{S}^{\mathrm{o}}(\boldsymbol{\xi})$ in the tangent directions are equal when the normal aligns with the velocity, i.e.:

$$\lim_{|p| \to 1} \mathscr{S}_d = \mathscr{S}_e, \quad d > 2..N,\, e > 2..N \tag{8.7}$$

Hence, the choice of orthonormal vectors $\boldsymbol{q}_d^{\mathrm{o}}, d > 2..N$ does not influence the control force as long as the matrix Q^{d} has full rank.

Damping Only Towards Obstacle

In the presence of an obstacle, the disturbances should be damped strongly when the agent is pushed against the obstacle. Conversely, the system can remain compliant if the motion is away from the obstacle. This is achieved by setting updating the first damping value $\mathscr{S}_1^{\mathrm{o}}$ if the robot is moving away from the surface:

$$\mathscr{S}_1^{\mathrm{o}}(\boldsymbol{\xi}) = \begin{cases} s^{\mathrm{o}} & \text{if } \left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}\right)^T \boldsymbol{n}(\boldsymbol{\xi}) > 0 \\ s^{\mathrm{c}} & \text{otherwise} \end{cases} \tag{8.8}$$

Due to the fact that $\boldsymbol{q}_1^o(\boldsymbol{\xi})$ given in (8.4) is pointing along the obstacle normal $\boldsymbol{n}(\boldsymbol{\xi})$, the first obstacle damping value $\mathscr{S}_1^{\mathrm{o}}(\boldsymbol{\xi})$ does not have any effect on the control force $\boldsymbol{\tau}^c$ given in (3.31) when $\left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}\right)^T \boldsymbol{n}(\boldsymbol{\xi}) = 0$. Hence, the damping value can be discontinuous across time, but the resulting control force remains continuous.

8.2.2 Damping for Velocity Preservation

The decomposition matrix $\mathscr{Q}^f$ is an orthonormal basis with the first vector being parallel to the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$, as proposed in Section 3.7.4. Hence, the values of the diagonal matrix $\mathscr{S}^f$ are high in the direction of the desired velocity (first component), but more compliant in the remaining directions. Moreover, the damping is set to ensure that when passing a narrow passage between two obstacles, where the normal vector cancels $\boldsymbol{n}(\boldsymbol{\xi}) \approx \boldsymbol{0}$, with additionally a low distance value $\Gamma(\boldsymbol{\xi}) \approx 1$, the damping perpendicular to the velocity direction is high. Hence, we set:

$$\begin{aligned} \mathscr{S}_d^f &= w^p s^{\mathrm{o}} + \begin{cases} (1-w^p)s^f & d = 1 \\ (1-w^p)s^s & d = 2..N \end{cases} \\ \text{with} \quad & w^p = \min\left(1, \|\boldsymbol{n}(\boldsymbol{\xi})\|^2 + \Delta\Gamma^2\right) \\ \text{and} \quad & \Delta\Gamma = \max\left(\frac{\Gamma^{\mathrm{crit}} - \Gamma(\boldsymbol{\xi})}{\Gamma^{\mathrm{crit}} - 1}, 0\right) \end{aligned} \tag{8.9}$$

8.2.3 Cluttered Environments

In a cluttered environment, the normal vectors of the individual obstacles can be opposing. And hence using (8.3) and (8.2), we get:

$$\|\boldsymbol{n}(\boldsymbol{\xi})\| \to 0 \quad \text{and} \quad \lim_{\Gamma(\boldsymbol{\xi})\to 1} w(\boldsymbol{\xi}) = 0 \tag{8.10}$$

Additionally using (8.1) and (8.9) we obtain:

$$\lim_{\Gamma\to 1, \|n\|\to 0} \mathscr{D}(\boldsymbol{\xi}) = \mathscr{D}^S(\boldsymbol{\xi}) + 0 = \mathscr{I} s^o \tag{8.11}$$

Hence, there is high damping in all directions to reject disturbances towards potential obstacles and ensure a collision-free motion even in cluttered environments.

8.2.4 Damping Parameter Design

Higher values for the damping parameters $s^{(\cdot)}$ generally result in improved velocity following and disturbance repulsion, whereas lower values allow more compliant behavior. The damping value in the direction of the obstacle is set high s^{o} to ensure obstacle avoidance even in the presence of

high disturbances. Conversely, the damping in the direction of the velocity s^f is set medium to high, as the system should follow the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$. However, it should remain compliant if needed. Finally, a low damping value s^c should be chosen for all other directions to facilitate interaction. This can be summarized as:

$$s^{\mathrm{o}} \geq s^f \gg s^c > 0 \tag{8.12}$$

8.2.5 Passivity Analysis

The stability analysis of the system gives information about the region of stability of the proposed controller. We analyze passivity by observing the evolution of the kinetic energy of the system, given as:

$$W(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) = \frac{1}{2} \dot{\boldsymbol{\xi}}^T \mathcal{M}(\boldsymbol{\xi}) \dot{\boldsymbol{\xi}} \tag{8.13}$$

Lemma 8.2.1. *Let us assume a robotic system as described in* (3.29) *is controlled using* (3.31) *using the damping matrix* $\mathcal{D}(\boldsymbol{\xi})$ *given in* (8.1) *with damping values* $s_d = 1, d = 1..N$*. The system is passive with respect to the input-output pair* $\boldsymbol{\xi}_e$, $\dot{\boldsymbol{\xi}}$ *when exceeding the desired velocity* $\boldsymbol{f}(\boldsymbol{\xi})$ *, i.e.,* $\dot{W} \leq \dot{\boldsymbol{\xi}}^T \boldsymbol{\tau}^e$, $\forall \boldsymbol{\xi} \in \mathbb{R}^N : \|\dot{\boldsymbol{\xi}}\| \geq \|\boldsymbol{f}(\boldsymbol{\xi})\|$ *and the storage function being the kinetic energy* $W \in \mathbb{R}_{>0}$ *given in* (8.13)

Proof. The time derivative of storage function W can be evaluated as:

$$\begin{aligned}
\dot{W}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) &= \dot{\boldsymbol{\xi}}^T \mathcal{M}(\boldsymbol{\xi}) \ddot{\boldsymbol{\xi}} + \frac{1}{2} \dot{\boldsymbol{\xi}}^T \dot{\mathcal{M}}(\boldsymbol{\xi}) \dot{\boldsymbol{\xi}} \\
&= \frac{1}{2} \dot{\boldsymbol{\xi}}^T \left(\dot{\mathcal{M}}(\boldsymbol{\xi}) - 2\mathcal{C}(\boldsymbol{\xi}) \right) \dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}^T \mathcal{D}(\boldsymbol{\xi}) \left(\dot{\boldsymbol{\xi}} - \boldsymbol{f}(\boldsymbol{\xi}) \right) + \dot{\boldsymbol{\xi}}^T \boldsymbol{\tau}^e \\
&= -\dot{\boldsymbol{\xi}}^T \mathcal{D}(\boldsymbol{\xi}) \left(\dot{\boldsymbol{\xi}} - \mathbf{f}(\boldsymbol{\xi}) \right) + \dot{\boldsymbol{\xi}}^T \boldsymbol{\tau}^e
\end{aligned} \tag{8.14}$$

where the second order dynamics $\ddot{\boldsymbol{\xi}}$ are evaluated according to the rigid body dynamics defined in (3.29). Furthermore, $\dot{\mathcal{M}} - 2\mathcal{C}$ is skew-symmetric for any physical system; hence, the corresponding summand is zero.

Let us investigate the region where the passivity holds. Since in the Lemma, we assumed all damping values being equal to one, we have:

$$\mathcal{D}(\boldsymbol{\xi}) = \mathcal{Q}(\boldsymbol{\xi}) \mathcal{S}(\boldsymbol{\xi}) \mathcal{Q}(\boldsymbol{\xi})^{-1} = \mathcal{Q}(\boldsymbol{\xi}) \mathcal{I} \mathcal{Q}(\boldsymbol{\xi})^{-1} = \mathcal{I} \tag{8.15}$$

where $\mathcal{I} \in \mathbb{R}^{N \times N}$ is the identity matrix.

It follows that the system is passive with respect to the input, the external force τ^e, and the output, the velocity $\dot{\boldsymbol{\xi}}$, if:

$$\dot{\boldsymbol{\xi}}^T \mathcal{D}(\boldsymbol{\xi}) \left(\dot{\boldsymbol{\xi}} - \boldsymbol{f}(\boldsymbol{\xi}) \right) = \dot{\boldsymbol{\xi}}^T \Delta \boldsymbol{f} \geq 0 \,, \quad \Delta \boldsymbol{f} = \dot{\boldsymbol{\xi}} - \boldsymbol{f}(\boldsymbol{\xi}) \tag{8.16}$$

On the border of this region, the two vectors $\Delta \boldsymbol{f}$ and $\dot{\boldsymbol{\xi}}$ are orthogonal. Hence, using Thale's

theorem, this region can be interpreted as a circle in velocity-space with radius $\|\boldsymbol{f}(\boldsymbol{\xi})\|/2$ and center $\boldsymbol{f}(\boldsymbol{\xi})/2$, see Figure 8.5.

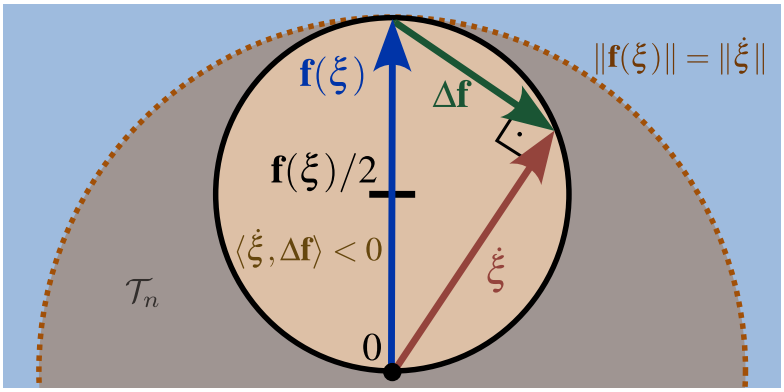


Figure 8.5: Analyzing the system in velocity-space, yields that the system is passive if it has a velocity $\dot{\boldsymbol{\xi}}$ larger than the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$, i.e., outside the dashed circle. However, the system can be non-passive for small velocities when $\langle \dot{\boldsymbol{\xi}}, \Delta \boldsymbol{f} \rangle < 0$ (yellow circle).

Moreover, the system is passive as long as the observed velocity $\dot{\boldsymbol{\xi}}$ is outside the circular-red region, which is a subset of the region where the magnitude of the observed velocity is smaller than the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$, i.e.,

$$\dot{W}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \leq \dot{\boldsymbol{\xi}}^T \boldsymbol{\tau}^e \quad \forall \boldsymbol{\xi} : \|\dot{\boldsymbol{\xi}}\| \geq \|\boldsymbol{f}(\boldsymbol{\xi})\| \tag{8.17}$$

□

As in the orange region, the system is not passive; the storage function W could increase, and hence, the velocity $\dot{\boldsymbol{\xi}}$ increases non-passively. This behavior is not unexpected, as the controller is designed to approach the desired dynamics $\boldsymbol{f}(\boldsymbol{\xi})$. Hence, as long as the desired velocity is not reached, the kinetic energy increases even with no force input $\boldsymbol{\tau}^e$. However, as soon as the system velocity $\dot{\boldsymbol{\xi}}$ exceeds the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$, the system behaves passively. We can use this to ensure the stability of the system:

Theorem 8.2.2. *Let $\boldsymbol{f}(\boldsymbol{\xi})$ is the desired velocity with bounded magnitude, i.e., $\|\boldsymbol{f}(\boldsymbol{\xi})\| < v^{\max}, \forall \boldsymbol{\xi} \in \mathbb{R}^N$. The closed loop system* (3.29) *using the controller from* (3.31) *and the damping matrix $\mathcal{D}(\boldsymbol{\xi})$ given in* (8.1) *is bounded-input, bounded-output (BIBO) stable with respect to the input disturbance force $\boldsymbol{\tau}^e$, and output the velocity $\dot{\boldsymbol{\xi}}$ for all times $T = 0, .., \infty$.*

Proof. To ensure BIBO stability, let us analyze the integral of the impulse of the response for the external force $\boldsymbol{\tau}^e$:

$$\begin{aligned} \int_0^T \|\dot{\boldsymbol{\xi}}\| \, dt &= \int_{t \notin \mathcal{T}_n} \|\dot{\boldsymbol{\xi}}\| \, dt + \int_{t \in \mathcal{T}_n} \|\dot{\boldsymbol{\xi}}\| \, dt \\ &\leq K_p + v^{\max} T_n \end{aligned} \tag{8.18}$$

where $\mathcal{T}_n$ denotes the set of time instances where the system is not shown to be passive (Fig. 8.5), specifically $\|\dot{\boldsymbol{\xi}}\| \leq \|\boldsymbol{f}(\boldsymbol{\xi})\|$, and $T_n \in \mathbb{R}_{\geq 0}$ is the total duration which the system spends in this region. Additionally, from passivity in the inner region, the system is bounded by a constant $K_p \in \mathbb{R}_{\geq 0}$. Hence, the impulse response is bounded, and the system is BIBO stable.

However, from (8.1), we know that a general damping matrix $\mathcal{S}(\boldsymbol{\xi})$ can have non-uniform diagonal values. This is analyzed by introducing the coordinate transfers:

$$\bar{\boldsymbol{v}} = \sqrt{\mathcal{S}(\boldsymbol{\xi})}\mathcal{Q}(\boldsymbol{\xi})^{-1}\dot{\boldsymbol{\xi}} \text{ and } \bar{\Delta\boldsymbol{f}} = \sqrt{\mathcal{S}(\boldsymbol{\xi})}\mathcal{Q}(\boldsymbol{\xi})^{-1}\Delta\boldsymbol{f} \tag{8.19}$$

where the square root of the diagonal matrix $\mathcal{S}(\boldsymbol{\xi})$ is taken element-wise. The transfer is then used to rewrite (8.16) as:

$$\dot{\boldsymbol{\xi}}^T\mathcal{D}(\boldsymbol{\xi})\Delta\boldsymbol{f} = \dot{\boldsymbol{\xi}}^T\mathcal{Q}(\boldsymbol{\xi})\mathcal{S}(\boldsymbol{\xi})\mathcal{Q}(\boldsymbol{\xi})^{-1}\Delta\boldsymbol{f} = \bar{\boldsymbol{v}}^T\bar{\Delta\boldsymbol{f}} \tag{8.20}$$

Hence, the BIBO analysis of (8.18) applied to the transformed system results as:

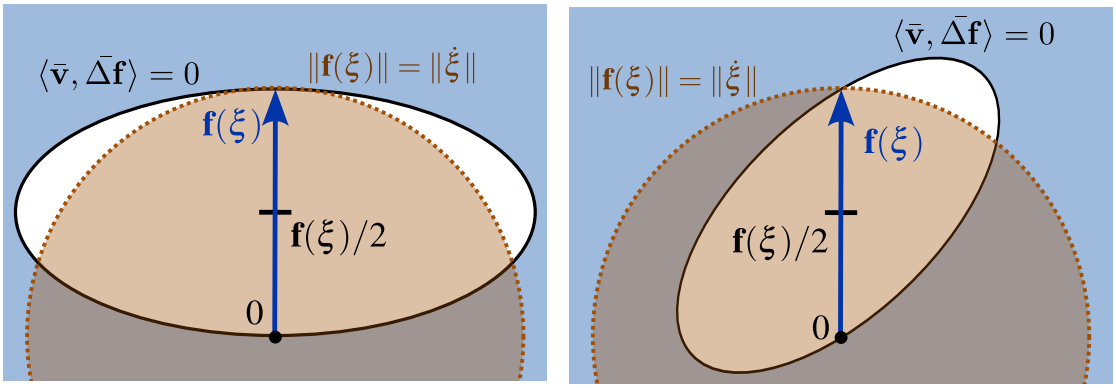
$$\begin{aligned}\int_0^T \|\bar{\boldsymbol{v}}\|\, dt &= \int_{t\notin\bar{\mathcal{T}}_n} \|\bar{\boldsymbol{v}}\|\, dt + \int_{t\in\bar{\mathcal{T}}_n} \|\bar{\boldsymbol{v}}\|\, dt \\ &< K_p + v^{\max}\bar{T}_n \max\Big(\mathrm{eig}\big(\mathcal{D}\big)\Big)/\min\Big(\mathrm{eig}\big(\mathcal{D}\big)\Big)\end{aligned} \tag{8.21}$$

where $\bar{\mathcal{T}}_n$ denotes the region where the transformed system $\bar{\boldsymbol{v}}$ is not shown to be passive, i.e. $\|\bar{\boldsymbol{v}}\| \leq \|\bar{\Delta\boldsymbol{f}}\|$, and $\bar{T}_n \in \mathbb{R}_{\geq 0}$ the corresponding time. Additionally, $\min(\mathrm{eig}(\mathcal{D}))$ and $\max(\mathrm{eig}(\mathcal{D}))$ are the smallest and largest eigenvalue of the damping matrix $\mathcal{D}$ respectively.

Hence, since the transformed system with velocity $\bar{\boldsymbol{v}}$ is BIBO stable, the original system with velocity $\dot{\boldsymbol{\xi}}$ is BIBO stable, too, as long as it is a continuous, finite transform. □

For an orthogonal decomposition matrix $\mathcal{Q}(\boldsymbol{\xi})$, the region of non-passivity is an ellipse where the direction of the axes points along column vectors of $\mathcal{Q}(\boldsymbol{\xi})$, and the corresponding axes lengths are the diagonal elements of $\|\mathbf{f}(\boldsymbol{\xi})^T\sqrt{\mathcal{S}(\boldsymbol{\xi})}\|/2\sqrt{\mathcal{S}(\boldsymbol{\xi})}^{(-1)}$. If the ratio of the first damping value to the other axis $i \geq 2$ is large, i.e., $s_1/s_i \gg 1$, it can lead to non-passivity even though the velocity $\dot{\boldsymbol{\xi}}$ is already larger (but not pointing in the correct direction) than the desired velocity. However, the non-passive region is still bounded (Fig.8.6b). This proof holds for any basis $\mathcal{Q}$ which is not singular. However, the controller must be carefully chosen to ensure that the speed up is limited when the basis is close to singular, for example, by limiting the relative difference of the stretching vectors. Furthermore, as stable behavior is ensured for a general shape of a damping matrix $\mathcal{D}(\boldsymbol{\xi})$, the global stability proof extends to any positive definite damping matrix matrices.

Since the damping matrix $\mathcal{D}(\boldsymbol{\xi})$ changes dynamically, a change in the environment can move the system outside of the passive region. However, there exists always a finite maximum velocity, at which the system is ensured to be passive.



(a) $\mathscr{S}_1^f > \mathscr{S}_2^f$, $w \approx 0$ (b) $\langle \boldsymbol{f}(\boldsymbol{\xi}), \boldsymbol{q}_1 \rangle \neq \|\boldsymbol{f}(\boldsymbol{\xi})\| \, \|\boldsymbol{q}_2\|$

Figure 8.6: The stability is ensured even if the controller can temporarily accelerate the system to reach a velocity $\dot{\boldsymbol{\xi}}$, which is faster than the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$ (white region). This happens when the eigenvalues of the damping matrix $\mathscr{D}(\boldsymbol{\xi})$ are not uniform (a) or the stretching vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are not orthogonal (b). In both cases, the region of non-passivity is elliptical (black circle).

8.3 Discrete Time Behavior

So far, we've considered that the system is continuous time. This is a reasonable assumption, if we have a high sampling time Δt, compared to the dynamics, i.e., $\|\Delta t \dot{\boldsymbol{\xi}}\| \ll 1$. However, any digital controller sends a discrete control signal. To reject the disturbances in the presence of obstacles, a high control force is exerted (while remaining within the control robot's limits), hence $\|\Delta t \boldsymbol{\tau}^c\| \gg 1$. In this case, it is crucial to analyze the discrete-time system to guarantee stability, as high damping can lead to unstable behavior, see Figure 8.7).

For the discrete-time system, the position and velocity of the agent evolve as follows:

$$\begin{bmatrix} \boldsymbol{\xi}_{t+1} \\ \dot{\boldsymbol{\xi}}_{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\xi}_t \\ \dot{\boldsymbol{\xi}}_t \end{bmatrix} + \Delta t \begin{bmatrix} \dot{\boldsymbol{\xi}}_t \\ \ddot{\boldsymbol{\xi}}_t \end{bmatrix} \tag{8.22}$$

Lemma 8.3.1. *Let us consider a discrete-time system with the state as given in Eq.* (8.22)*, and is governed by the controller from Eq.* (3.31) *and damping matrix $\mathscr{D}$ defined in Eq.* (8.1)*. The system is BIBO (bounded-input, bounded-stable) with respect input the desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$ the velocity, and as output the agent's velocity $\dot{\boldsymbol{\xi}}$ such that $\lim_{t\to\infty} \|\dot{\boldsymbol{\xi}}\| < \infty$, if all damping values are limited as $s_d \leq 2\min\left(eig\left(\mathscr{M}\right)\right)/\Delta t$ with $d = 1..N$.*

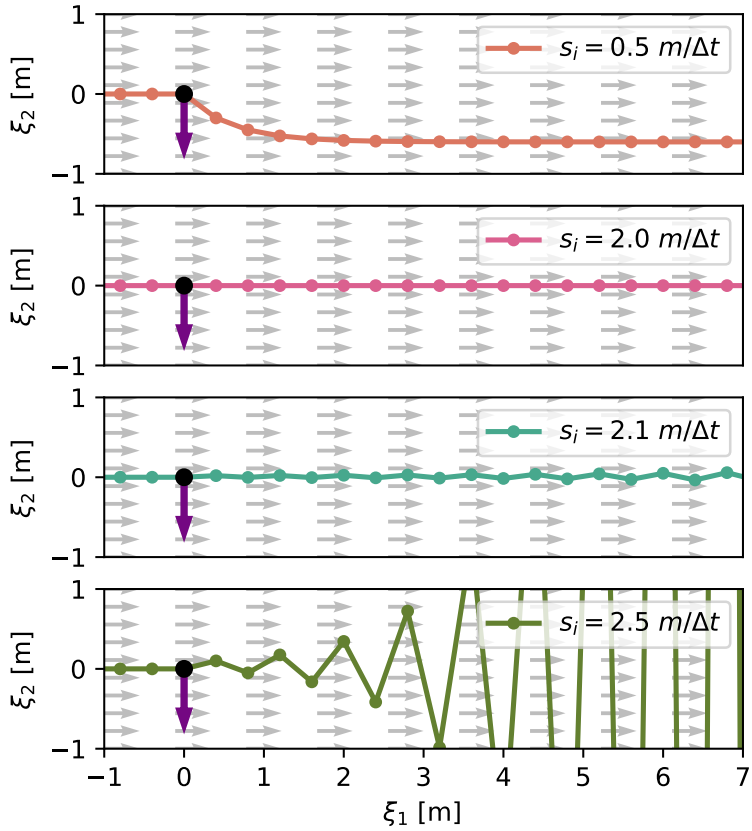


Figure 8.7: An agent has the desired velocity $\boldsymbol{f}(\boldsymbol{\xi}) = [1,0]^T$ (gray arrow), and a control matrix $\mathscr{D}$ with damping values equal in all directions and the smallest eigenvalue of the inertia matrix m. The agent is disturbed (purple arrow) position $\boldsymbol{\xi}_0 = [0,0]^T$ and has a velocity of $\boldsymbol{\xi}_1 = [1,1]^T$ after the impact.
High damping leads to unstable behavior with increasingly high oscillations. Conversely, the lowest value leads to more deviation from the initial straight line resulting from the higher impedance. The critical value of $s_i = 2.0m/\Delta t$ results in stable behavior with immediate correction to the desired velocity.

Proof. The evolution of the discrete-time feedback system is given as:

$$\begin{aligned}\begin{bmatrix}\boldsymbol{\xi}_{t+1}\\ \dot{\boldsymbol{\xi}}_{t+1}\end{bmatrix} &= \begin{bmatrix}\boldsymbol{\xi}_t + \Delta t\, \dot{\boldsymbol{\xi}}_t \\ \dot{\boldsymbol{\xi}}_t + \Delta t\, \mathscr{M}^{-1}\left(\mathscr{D}\left(\boldsymbol{f}(\boldsymbol{\xi}_t) - \dot{\boldsymbol{\xi}}_t\right) - \mathscr{C}\right)\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{I} & \boldsymbol{I}\Delta t \\ \boldsymbol{0} & \boldsymbol{I} - \Delta t \mathscr{M}^{-1}\mathscr{D}\end{bmatrix}\begin{bmatrix}\boldsymbol{\xi}_t \\ \dot{\boldsymbol{\xi}}_t\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ \Delta t\, \mathscr{M}^{-1}\mathscr{D}\end{bmatrix}\hat{\boldsymbol{f}}(\boldsymbol{\xi}_t)\end{aligned} \tag{8.23}$$

where $\hat{\boldsymbol{f}}(\boldsymbol{\xi}_t) = \boldsymbol{f}(\boldsymbol{\xi}_t) - \mathscr{D}^{-1}C$. The dependency on the state $\boldsymbol{\xi}$ of the matrices $\mathscr{M}$, $\mathscr{D}$, and $\mathscr{C}$ are omitted for brevity. $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ denotes the identity matrix. As we look at global stability, we look at the updated velocity $\hat{\boldsymbol{f}}(\boldsymbol{\xi}_t) = \boldsymbol{f}(\boldsymbol{\xi}_t) - \mathscr{D}^{-1}\mathscr{C}$. Since the Coriolis force is bounded, it follows that if the system is BIBO stable for $\hat{\boldsymbol{f}}(\boldsymbol{\xi}_t)$, then it is also BIBO stable for $\boldsymbol{f}(\boldsymbol{\xi}_t)$

BIBO stability of a discrete-time system requires that all the eigenvalues of the feedback matrix are smaller or equal to one (Friedland, 2012). The eigenvalues of the above feedback system are given as:

$$\boldsymbol{\lambda}_1 = \mathrm{eig}(\boldsymbol{I}) \qquad \boldsymbol{\lambda}_2 = \mathrm{eig}\left(\boldsymbol{I} - \Delta t\, \mathscr{M}^{-1}\mathscr{D}\right) \tag{8.24}$$

where both $\boldsymbol{\lambda}_1 \in \mathbb{R}^N$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^N$ denote a vector containing N eigenvalues. All eigenvalues of $\boldsymbol{\lambda}_{1,i} = 1,\ i = 1...N$, and enable a stable system. The second set of eigenvalues $\boldsymbol{\lambda}_2$ depends on the passive control term. However, since $\mathscr{M}$ and $\mathscr{D}$ are both positive definite, the eigenvalues are positive:

$$\mathscr{M} > 0\,,\ \mathscr{D} > 0 \qquad \Rightarrow \qquad \boldsymbol{\lambda}_{2,i} > 0 \quad i = 1...N \tag{8.25}$$

Hence, we only have to consider the upper limit, and the system is stable if the maximum eigenvalue is limited to:

$$\max(\boldsymbol{\lambda}_2) \leq 1 \quad \Rightarrow \quad s_i \leq 2\min\Big(\mathrm{eig}\big(\mathscr{M}\big)\Big)/\Delta t \tag{8.26}$$

□

The stability guarantees provide BIBO stability, hence the boundedness of the output. Since the first eigenvalues equal one, there is no global asymptotic convergence. In fact, in the system from (8.23), it can be observed that when the input dynamics are zero, i.e., $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{0}$, the system immediately stops. But there is no convergence to a specific position. However, in most cases, the desired system should only reach zero at the attractor position, and hence, we expect convergence to such a point.

Yet, asymptotic stability is not guaranteed for general nonlinear input dynamics. Especially for dynamics with high curvature and low damping value, the final trajectory can end up in limit cycles ((Figure 8.8).

In practice, it is useful to use a large value for s^o as it rejects disturbances towards the obstacles, and lower values for the dynamics following s^f and damping in general direction s^c. Hence, we

propose $s^o = 2.0m/\Delta t$, $s^f = 1.0/\Delta t$, and $s^c = 0.1/\Delta t$.

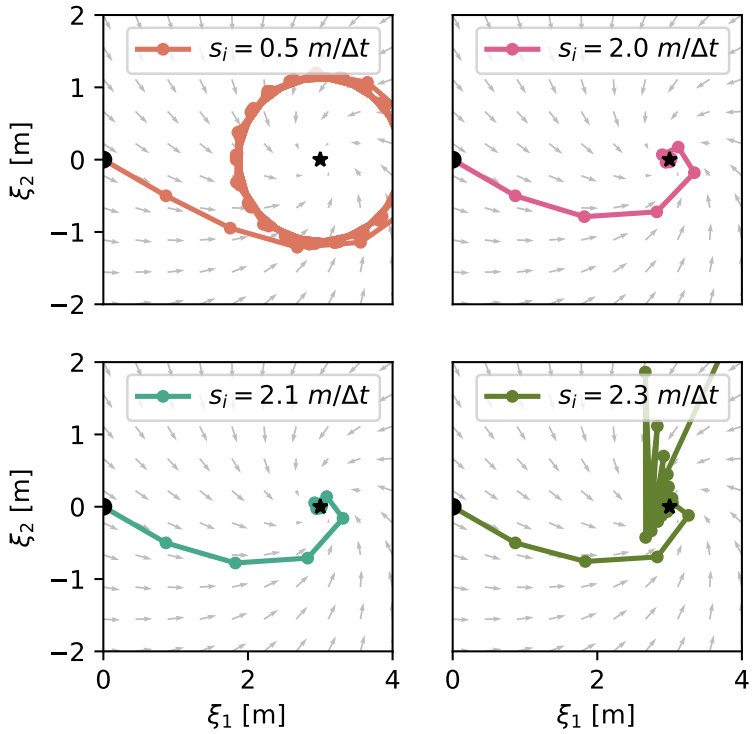


Figure 8.8: An agent with the desired dynamics of $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{R}(\pi/6)(\boldsymbol{\xi} - \boldsymbol{\xi}^a)$ where $\boldsymbol{R}(\cdot) \in \mathbb{R}^{N \times N}$ is the rotation matrix, and m is the mass of the agent. We assume equal damping values s_i. The controller with a critical stiffness of $2.0m/\Delta t$ leads to fast convergence and a stable system. With lower damping (top left), there is a large drift of the system, which converges to a limit cycle. The high damping of $2.3m/\Delta t$ leads to an unstable system. Interestingly, with damping of $2.1m/\Delta t$ in combination with the nonlinear dynamics, we obtain a visibly stable system.

8.4 Evaluation

The proposed obstacle aware controller[1] is compared to a baseline, the velocity preserving, passive controller (Kronander and Billard, 2015).

8.4.1 Qualitative Repulse Rejection

A qualitative analysis of the proposed controller's behavior in three scenarios, as depicted in Figure 8.9. In each scenario, the agent approaches multiple obstacles from distinct starting positions, a disturbance (indicated by an arrow) is applied. The simulation time step is $\Delta t = 0.01s$ seconds, and the agent's mass matrix is $\mathcal{M} = \boldsymbol{I}kg$. The controller is implemented using the following damping values: $s^{\text{obs}} = 200\,\text{s}^{-1}$, $s^{\text{f}} = 100\,\text{s}^{-1}$, and $s^{\text{c}} = 20\,\text{s}^{-1}$.

[1] Source code: https://github.com/hubernikus/obstacle_aware_damping

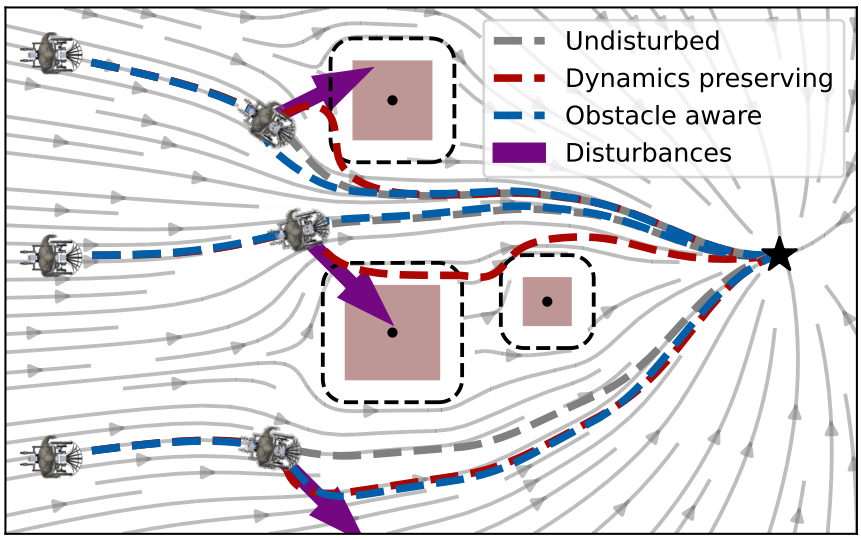


Figure 8.9: The desired velocity $\boldsymbol{f}(\boldsymbol{\xi})$ is represented in gray and serves as the input for the force controller. The mobile robot, initially positioned at three different locations, navigates safely towards the attractor (black star) even when confronted with external disturbances (purple arrows) while employing the obstacle-aware controller (blue trajectories). In contrast, the baseline controller (red) results in collisions when disturbances occur close to the robot.

In the top trajectory, the robot encounters a stand-alone obstacle and experiences a disturbance that pushes it toward the obstacle. With the obstacle-aware controller, the robot avoids collision and continues moving towards the attractor. In contrast, the baseline controller, which prioritizes velocity conservation, fails to respond effectively and is pushed into the obstacle, resulting in a colliding trajectory.

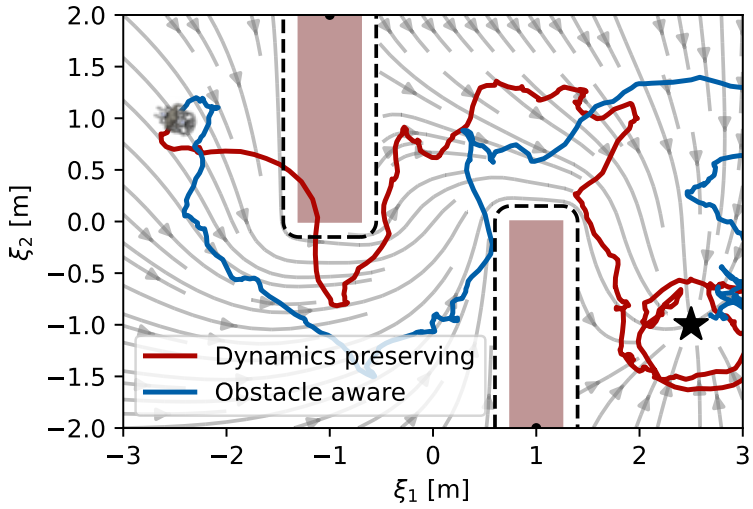
During the middle trajectory, a disturbance occurs when the agent is positioned between two obstacles. The obstacle-aware controller utilizes the normal vector $\boldsymbol{n}(\boldsymbol{\xi})$, as described in Section 8.2.1. However, due to its construction, the magnitude of $\boldsymbol{n}(\boldsymbol{\xi})$ diminishes in narrow passages (as seen in (8.3)), leading to increased damping in all directions, as described in (8.8). As a result, the agent successfully avoids the disturbance using the obstacle-aware controller, while the baseline controller follows a colliding trajectory.

In the bottom trajectory, the repulsive force points away from the obstacle. The obstacle-aware and the velocity-preserving controllers produce nearly identical trajectories due to equal compliance when moving away from an obstacle as defined in (8.8).

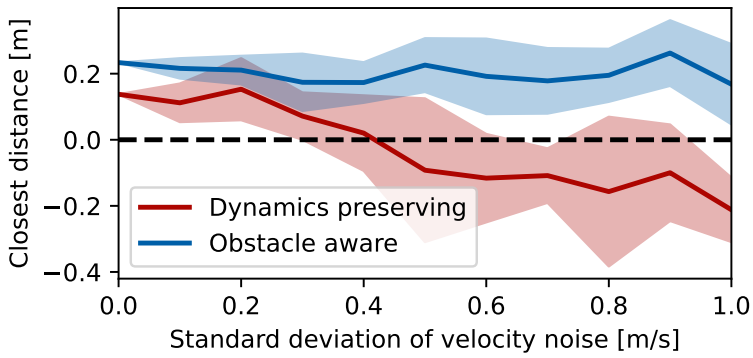
We observe the selective damping of disturbances towards the obstacle. The top and middle disturbances are highly damped, while the bottom disturbances are not. This feature imparts a natural behavior of moving away from obstacles.

8.4.2 Noise Analysis

In practical robotic applications, controllers are inevitably exposed to noise and unexpected disturbances, arising from sensor inaccuracies, environmental factors, or other external perturbations. A high-quality controller can effectively reject such disturbances while simultaneously achieving the control objectives, such as collision avoidance and trajectory tracking.



(a) Trajectories with a standard deviation of the velocity-noise of 1.0 m/s.



(b) The mean and variance (shaded) of the closest distances over 10 epochs.

Figure 8.10: The agent navigates between two elongated obstacles (a) towards the attractor (black star). The agent's velocity is subjected to white noise, with a mean of zero, and noise variances between 0.0 m/s and 1.0 m/s. The robot initiates its trajectory from the starting position $\boldsymbol{\xi}_0 = [-2.5, 1.0]^T$, with an initial velocity of zero. It aims to reach the attractor located at $\boldsymbol{\xi}^a = [2.5, -1.0]^T$.

This section investigates the impact of noise disturbances on a simulated agent with an identity mass matrix $\mathcal{M} = \boldsymbol{I}kg$. The discrete time step is set to $\Delta t = 0.2\,\mathrm{s}$. Additionally, the damping values are configured as follows: $s^{\mathrm{o}} = 50\,\mathrm{s}^{-1}$, $s^{\mathrm{f}} = 40\,\mathrm{s}^{-1}]$, and $s^{\mathrm{c}} = 5\,\mathrm{s}^{-1}$. The robot's objective is to follow a linear velocity field of the form $\boldsymbol{f}(\boldsymbol{\xi}) = (\boldsymbol{\xi}^a - \boldsymbol{\xi})$ with a velocity cap of 1 m/s. To

evaluate the controller's performance, a comparative analysis is conducted by assessing the minimal distance to the surface along the trajectory, denoted as $\min_t \|\boldsymbol{\xi}_t - \boldsymbol{\xi}^b\|$, with the boundary point $\boldsymbol{\xi}_b$ described in (6.15).

Velocity Noise Resistance

We first added a normally distributed noise with a zero mean to the agent's velocity $\dot{\boldsymbol{\xi}}$ at each time step before computing the control force. The agent has to navigate between two obstacles from the starting position to the attractor with different noise levels, as visualized in Figure 8.10)

The obstacle-aware controller effectively rejects the noise impacting the velocity, even as the noise variance increases. However, the closest distance during the trajectory diminishes with higher noise variance. On the contrary, the velocity-following controller's mean distance falls below zero already at a velocity noise variance of $0.5\,\mathrm{m/s}$, indicating that many trajectories collide with at least one obstacle.

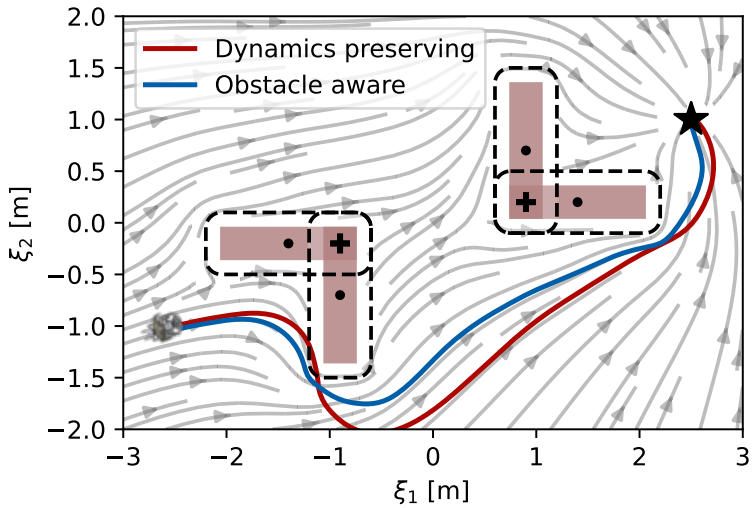
Furthermore, the obstacle-aware controller maintains a higher minimal distance along the trajectory without noise. This effect results from the higher damping applied towards the obstacle, enabling more precise tracking of the curvature guiding the velocity around the obstacle.

Position Noise Resistance

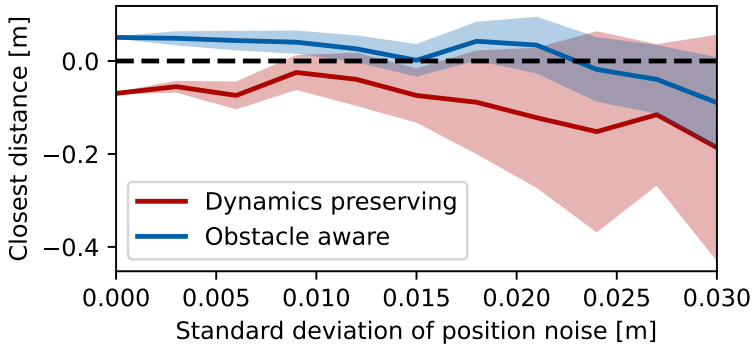
In the second experiment, normally distributed noise with a mean of zero is added to the position $\boldsymbol{\xi}$ (Fig. 8.11). The agent has to navigate around two star-shaped obstacles to reach the attractor with various noise levels being applied.

The obstacle-aware controller maintains a greater distance to the surface even when no noise is present, while the velocity-preserving controller already exhibits collisions. As a result, the obstacle-aware creates trajectories with the mean distance above zero for standard deviations of the position noise smaller than $0.023\,\mathrm{m}$. Conversely, the velocity-preserving controller's mean distance to the surface is below zero for all experiment runs, indicating collisions. This difference is attributed to the buffer mechanism inherent in the obstacle-aware controller. Despite a similar decrease in distance for both controllers, the obstacle-aware controller effectively prevents collisions due to its higher damping of velocities toward the obstacles.

Moreover, the velocity-preserving controller exhibits a higher distance variance, likely stemming from its lower damping, causing it to adapt more slowly to new velocities after being displaced by the noise. Consequently, this behavior leads to more random variations in velocity and trajectory.



(a) Trajectories with a standard deviation of the position-noise of 0.0 m.

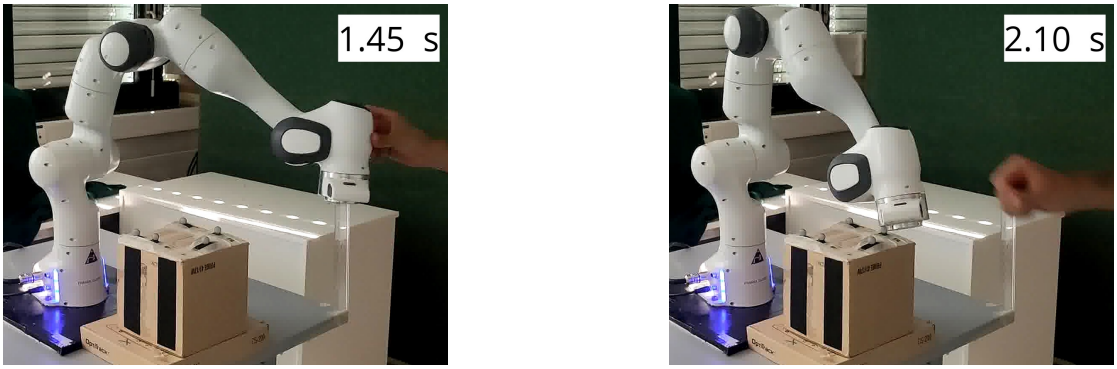


(b) Closest distances concerning different noise levels over ten epochs.

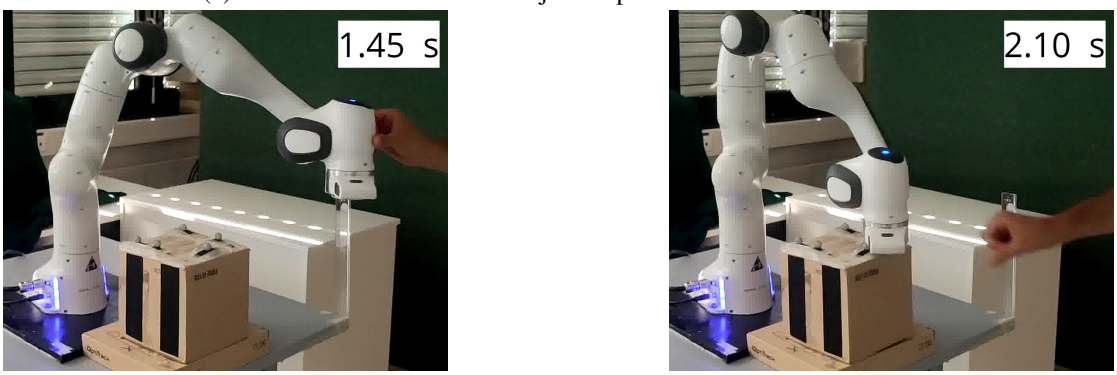
Figure 8.11: The agent is navigating towards the attractor (black star) between two concave obstacles (a) while being subjected to white noise in its position. The position noise has a mean of zero and various noise variances ranging between 0.0 m and 0.03 m. The robot starts at position $\boldsymbol{\xi}_0 = [-2.5, -1.0]^T$, and the attractor is set to $\boldsymbol{\xi}^a = [2.5, 1.0]^T$.

8.4.3 Obstacle Aware Passivity Using a Robot Arm

The obstacle-aware passivity controller was implemented to guide a 7-degree-of-freedom robot arm (Panda from Franka Emika) while moving around a cubic obstacle.



(a) Obstacle-aware controller rejects repulsion and avoids collision



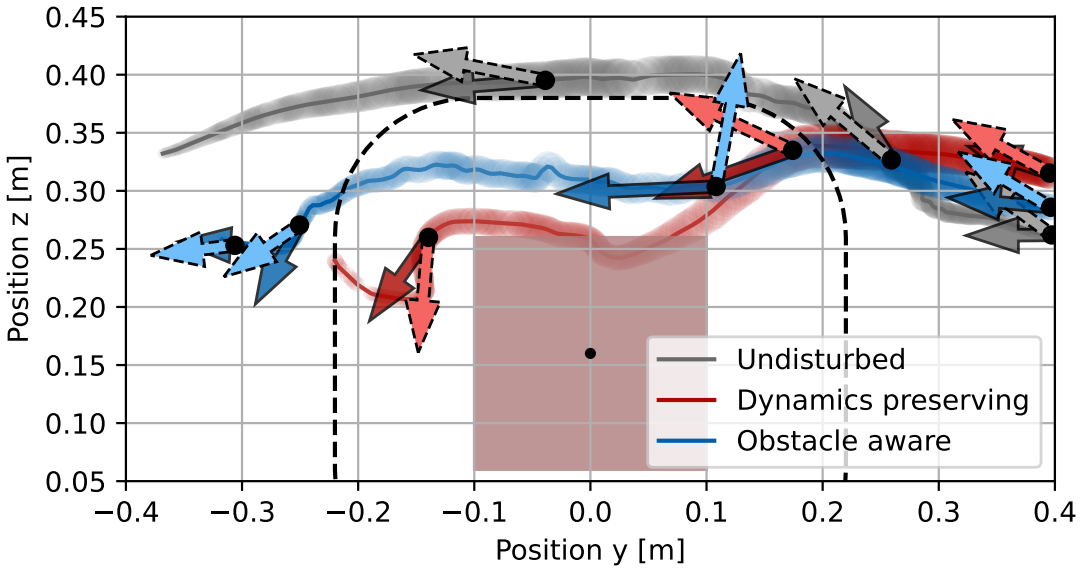
(b) Velocity preserving controller leads to collision with obstacle

Figure 8.12: The robot arm, guided by the obstacle-aware passive controller, effectively avoids the disturbance towards the obstacle while maintaining a margin of 0.16 m around the obstacle.

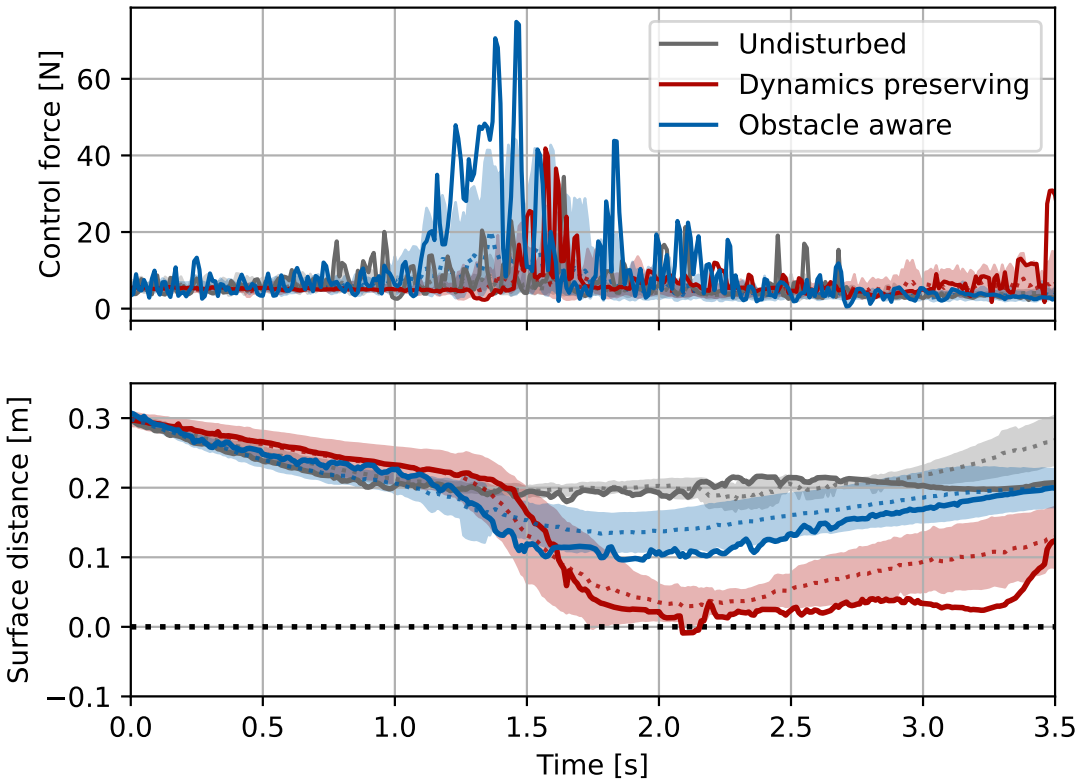
The joint torque is computed using inverse kinematics combined with the proposed passive controller for the position, but a proportional controller for the orientation:

$$\boldsymbol{\tau}_q = \boldsymbol{J}^{\dagger}(\boldsymbol{q}) \begin{bmatrix} \mathscr{D}(\boldsymbol{\xi})\left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}\right) \\ p^{\alpha}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{a}) \end{bmatrix} \tag{8.27}$$

where $\boldsymbol{J}^{\dagger}$ represents the Moore-Penrose pseudo inverse of the Jacobian matrix, and $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^{a}$ denote the end effector's orientation and the desired orientation, respectively. The angular damping parameter is chosen as $p^{\alpha} = 5.5$. The desired orientation $\boldsymbol{\alpha}^{a}$ is pointing downward with a quaternion value of $(w, x, y, z) = (0, 1, 0, 0)$. For the angle subtraction, we use quaternion representation to avoid singularities, but an angle-axis representation of the orientation is used to evaluate the torque from the angular offset.



(a) The two control methods compared with the undisturbed trajectory. The wider line indicates a higher x-value. The darker arrow is the actual, and the desired velocity is brighter arrow.



(b) The specific trajectory is represented by a full line, while the average (dashed line) and variance (shaded area) are evaluated over ten epochs. The control force's mean and variance are evaluated in the logarithmic space.

Figure 8.13: The experiment was repeated ten times with a similar disturbance applied to the robot arm in each run.

The angular damping is chosen as $p^{\alpha} = 5.5$. The damping values are set as $s^{\mathrm{o}} = 160\,\mathrm{s}^{-1}$, $s^{\mathrm{f}} = 64\,\mathrm{s}^{-1]}$, and $s^{\mathrm{c}} = 16\,\mathrm{s}^{-1}$.

The robot start position is approximately at $\boldsymbol{\xi}_0 = [0.3\mathrm{m}, 0.4\mathrm{m}, 0.3\mathrm{m},]^T$ and the attractor is at position $\boldsymbol{\xi}^a = [0.26\mathrm{m}, -0.53\mathrm{m}, 0.33\mathrm{m}]^T$. The controller operates at a frequency of 500 Hz. The robot encounters a single squared box with axes length 0.16 m and a margin of 0.12 m, placed in front of the robot base (Fig. 8.13). The precise location of the box is tracked in real-time using a marker-based vision system (Optitrack). When passsing the box, the robot is pushed with $\boldsymbol{t}^e$ towards the box. The experiment is repeated ten times for both controllers, and compared to the undisturbed motion.

The experimental results in Figure 8.12 and Figure 8.13 demonstrate that the obstacle-aware controller allows the robot to maintain an average distance of 0.15 m away from the obstacle surface. In contrast, the velocity-preserving controller results in a mean distance below 0.05 m, and numerous trajectories lead to collisions with the box. On average, the robot passes further away from the obstacle using the obstacle-aware controller and exhibits higher forces, Table **??**.

This outcome is attributed to the obstacle-aware controller's stronger control force, with a high peak occurring around 1.45 s when the disturbance is encountered. In contrast, the velocity-preserving controller only reacts when the robot is almost colliding, leading to a delayed response. Additionally, the obstacle-aware controller exhibits higher forces even before the disturbance, which contributes to improved tracking of the avoidance trajectory, as observed in Section 8.4.2. These findings affirm the superior collision avoidance capabilities and tracking performance of the obstacle-aware passivity controller in real-world robot arm scenarios.

8.5 Discussion

We introduced a novel passive obstacle-aware controller that takes as an input the desired, collision-free velocity and outputs a control torque. The stability proof enables the controller to be used with any bounded input velocity field, and this result extends to a general class of damping controllers. Furthermore, the controller is shown to reject disturbances, and the parameter-tuning for the discrete-time system has been analyzed. The controller was experimentally evaluated and compared to a baseline passive controller. The proposed approach shows increased resistance to noise, both in position and velocity and improved tracking. Applied to a real robot arm, the disturbance force was successfully rejected ensuring collision avoidance while following a reference motion.

8.5.1 Discrete Control Convergence

The proposed controller is only stable but does not ensure convergence. However, convergence could be achieved by modifying the controller by introducing an additional proportional term, as

is common in impedance controllers:

$$\boldsymbol{\tau_c} = \boldsymbol{g}(\boldsymbol{\xi}) + \mathscr{D}(\boldsymbol{\xi})\left(\boldsymbol{f}(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}\right) + \mathscr{K}(\boldsymbol{\xi})\left(\boldsymbol{\xi} - \boldsymbol{\xi}^a\right) \tag{8.28}$$

However, this design interferes with the basis passive controller, and the convergence towards the attractor is facilitated by (stable) velocity $\boldsymbol{f}(\boldsymbol{\xi})$. Moreover, introducing a variable damping matrix will result in a potentially unstable system, as the passivity proof no longer holds. Future work should further analyze how to include dynamics properties in the design of the damping parameters for improved system performance.

8.5.2 Applicability and Theoretical Analysis

The theoretical analysis from Theorem 8.2.2 indicates BIBO stability for a system with bounded desired velocity. Consequently, controllers like the damping-based approach in (Kronander and Billard, 2015) do not require an energy tank. Yet, introducing an adaptive proportional term $\mathscr{K}$ can lead to instabilities (Ferraguti, Secchi, and Fantuzzi, 2013; Kronander and Billard, 2016). Careful consideration of the controller design and stability analysis is necessary to ensure robust and safe performance in practical applications. Nevertheless, the passivity proof enables a broad range of time-varying damping controllers to be safely applied to robotic systems.

8.5.3 Point Mass Agents

While the controller is limited to a point mass representation of the robot, its computational efficiency would allow multiple evaluations along the robot arm. This could be used to ensure full body control of the robot, hence improving the system's safety.

8.5.4 Caution in Obstacle's Proximity

In this work, we assume the obstacles' position to be precisely known. However, in many scenarios, the robot might have limited perception as it approaches an obstacle. Hence, the robot should be more compliant to enable safe workspace exploration rather than increasing the damping. Future work should explore how to combine these two opposing paradigms: safe control for avoidance, yet cautious exploration.

9 Conclusion

This chapter highlights the primary contributions of this thesis while acknowledging the limitations and outlining future directions that arise from our work.

9.1 Contributions

Throughout this thesis, we have advanced the field of robotics by introducing novel concepts and methodologies to enhance dynamic and adaptive obstacle avoidance.

In Chapter 4, we introduced novel mathematical concepts and tools for developing and applying the obstacle avoidance algorithms presented in this thesis. Beyond the scope of our research, we believe these tools can be utilized for broader robotics and control applications.

Chapter 5 extended the modulation-based obstacle algorithms to human-inhabited environments. This was achieved through three concepts: inverted obstacles to describe complex indoor environments, the ability to navigate smoothly around non-smooth obstacles, and the adaptability to dynamic environments with deforming obstacles, such as an arrangement of people encapsulated in one obstacle. These advancements were demonstrated through successful applications in an autonomous wheelchair navigating indoor office-like spaces and through simulated and real crowds.

Chapter 6 addressed the substantial delay introduced by decoding image- or pointcloud-based environment recognition in many avoidance pipelines. We introduced a modulation-based, fast obstacle avoidance method that operates directly on unstructured sensor data. This method rapidly generates feasible avoidance velocities, similar to a reflex action of animals or humans, albeit without global convergence guarantees. The reactivity of this approach was validated during shared control of a semi-autonomous wheelchair across a busy marketplace.

Chapter 7 introduced a vector rotation to handle more complex obstacles, such as trees-of-stars, combined with nonlinear initial dynamics. This approach employed a stereographic projection to

ensure obstacle avoidance for both linear and nonlinear velocities The representation of various initial dynamics was standardized through a series of mappings. Additionally, introducing vector trees enabled handling highly concave environments divided into clusters of starshaped obstacles. These innovations facilitated obstacle avoidance in complex scenarios. They were validated with a robot arm following nonlinear limit cycles on a conveyor belt while avoiding incoming parcels represented by trees-of-stars.

Chapter 8 presented a method to ensure obstacle avoidance of a force-controlled robot while encountering external disturbances. Our adaptive damping controller was designed to reject disturbances in the direction of obstacles. Conversely, the controller remained compliant when the disturbance did not introduce any danger of colliding. The proposed method was applied to a collaborative robot arm, showcasing increased robustness against disturbances. Moreover, the controller was proven to maintain stability through the passivity theorem, which opens avenues for more flexible controller design.

Chapter A extends our research into the domain of autonomous volumetric multi-agent systems. The obstacle avoidance method using point-like agents with obstacles representing holes in the state-space is extended to volumetric agents by using multiple control points. The inherent computational efficiency allows multiple evaluations of the algorithm in real-time. This extension was applied to a simulated assisted living environment involving autonomous furniture agents, as it could be useful in hospitals and elderly care homes.

In summary, this thesis presents contributions to reactive collision avoidance. We expand the applicability of these algorithms, accelerate their execution time, ensure safety in human robots through force control, and even pave the way for multi-agent navigation.

9.2 Limitations and Future Work

Our work has made significant steps in addressing complex navigation in dynamic environments. This opens the door to numerous avenues for future research. On the one hand, this stems from the developed, novel tools. On the other hand, it originates from identifying limitations and assumptions limiting the scope of the presented work.

9.2.1 System Simplification

The obstacle avoidance algorithm proposed in this thesis relies on the assumption of star-shaped obstacles, which has been extended to trees of stars. However, further research is needed to investigate the pre-processing to enable the reduction of a general environment in real-time. Nevertheless, we were excited to see that since the beginning of this thesis, there have been promising developments in creating star-like representations from general obstacles and unstructured sensor data by Dahlin and Karayiannidis, 2023a. That work directly extends our first contribution in Chapter 5, and we believe this is a crucial step towards bringing the developed tools of this thesis

to everyday life.

9.2.2 Extension Beyond Obstacle Avoidance

Motion planning for robots can often be summarized as *following a path while reaching a desired configuration while avoiding specific regions in the state space*. The modulation method proposed in this work could create dynamics that guide away from undesired regions while converging to the attractor. Throughout this work, we introduced avoidance methods that can handle convex shapes, trees-of-stars, and surrounding walls. This can be used as a set of tools to constrain and guide the dynamics, where the obstacles represent virtual undesired regions. Such a constraining environment could be similar to unstructured cell decomposition, as introduced in Section 2.1.1. However, we believe to achieve this, further development is needed on how to represent soft obstacle boundaries, meaning boundaries that can be penetrated if necessary but should ideally be avoided. Additionally, future work needs to look into decomposing the environment into cells. Such approaches could potentially profit from learning from demonstration methods as briefly described in Section 7.4.2.

9.2.3 Combination of Perception and Control

In this work, we introduced new velocity and force control approaches for robots in the presence of obstacles. The research relied on existing perception systems, such as combining cameras with two-dimensional lasers on the mobile robot or infrared cameras and reflective markers with the robot arm. The obstacle position from these systems was considered absolute truth. However, it is crucial to acknowledge that most modern perception systems come with uncertainties. They often rely on neural network-based image detection (Sultana, Sufian, and Dutta, 2020), exhibiting non-negligible error rates that increase in unknown environments. To account for this, future research should consider the co-design of perception and control systems. A unified model integrating perception and control could significantly enhance safety in uncertain environments, encourage exploration of unknown regions, and enable adaptive motion in response to uncertainties of the obstacle's surfaces. This approach should involve adapting motion based on perceptual inputs and incorporating fundamental properties like convergence and collision guarantees, which have been central to our work. Furthermore, the design must address the inherent limitations of robots, including reactivity constraints due to acceleration limits and sensor precision, as well as kinematic constraints. Ideally, the controller should be adaptive and automatically update its parameters while maintaining stability, as is common in control theory (Landau et al., 2011). While such developments exist for MPC approaches (Xing et al., 2023), we believe many possibilities exist for integrating vision into closed-form control algorithms.

9.2.4 From Local Avoidance to Global Planning

As highlighted in Chapter 1, every path-planning algorithm involves a trade-off between computational cost and global convergence. The obstacle avoidance algorithms in this work have a local nature but excel in fast, short evaluation time. While we demonstrated to reduce computational costs while preserving specific convergence properties, this fundamental limitation persists. Even though the methods provide global convergence guarantees, they rely on the environments to meet certain conditions (e.g., star-shaped), which can only be guaranteed in local or simplified global environments. Even the algorithm developed in Chapter 7, which can handle complex trees-of-stars, has limited practical applicability in complex environments. This follows from the fact that constraining the vector field to be globally smooth leads too locally to high curvature of the dynamics around complex obstacles. We believe that further research could investigate how to extend and combine the proposed obstacle avoidance algorithms with alternative collision avoidance methods.

9.3 Final Thoughts

Finally, after an extensive dive into motion planning, obstacle avoidance, and robot control, I want to share some reflections on the field's challenges.

9.3.1 Extensive Field

The field of obstacle avoidance and path planning is vast, as evident from the extensive (in no means complete) body of work outlined in Chapter 2. This domain continues to expand rapidly. The primary focus of this thesis has been to extend closed-form control methods into faster and more complex environments. However, the development of new obstacle avoidance algorithms is subject to the following challenges:

1. **Algorithms Comparisons:** A core challenge in robotics lies in effectively operating within highly diverse environments, many of which are shared with humans. In such a scenario, comparing the algorithms' performance against well-established benchmarks is important. For instance, in Chapter 5, we utilized a crowd simulator originally developed as part of the benchmarking process presented in CrowdBot, 2023. This simulator allows for an in-depth analysis of algorithm performance within simulated crowd scenarios. However, any computational model carries limitations, and no simulation can perfectly replicate the complexities of a real crowd. This became evident during the real-world experiment with the robot in the marketplace, as parameters such as weather conditions, cultural norms, opening new shops nearby, and the time of day profoundly influence human behavior. The vast amount of known and unknown variables makes it challenging to replicate experiments involving human subjects (Camerer et al., 2018). Consequently, the field of robotics in human-shared environments often confronts a trade-off. On the one hand, there is the option

of relying on quantitative benchmarks within a controlled and simplified environment or simulator, which offers reproducibility but may lack real-world fidelity. On the other hand, there is the allure of qualitative real-world applications, which better encapsulate the complexities of human interaction but introduce a higher degree of uncertainty into the evaluation process.

2. **Variety of Applications:** Robots are designed to be versatile machines. Hence, a vast scope of work is produced under the research umbrella of robotics, from soft-robotics hardware to high-speed drone controllers to compliant but slow interactive motion planning. Moreover, the focus of academic research on the number of publications (Kiai, 2019) leads to many journal publications and larger conference proceedings. Due to the vast amount of work with great variety, it becomes hard to crystallize important contributions among recent works, even in a sub-field such as obstacle avoidance.

3. **Slow Adoption:** While a vast amount of work is being produced in research, the adoption of novel methods in the industry and research is relatively slow. While this is partially the result of the previous two points, it is enhanced because the real world is often more complicated than a robotics laboratory environment. What has worked in a fully controlled environment might fail if the number of free parameters increases. As a result, most robots in the industry are still working in almost fully controlled environments. Understandably, the leap to have robots exit such a safe zone is a big step, as it requires full assurance that the robot performs safely in a human-shared environment.

9.3.2 Problems Become Harder

Primary novel algorithms were developed in the 20th century in closed-form obstacle avoidance. Recent advancements in motion planning tend to be more incremental. Many recent publications combine and slightly adapt existing algorithms rather than introducing entirely new ones. For example, the original Artificial Potential Field (APF), see Section 2.1.3, is a simple yet elegant solution. More recent approaches, such as navigation functions, see Section 2.1.3, are far more complex in their underlying mathematics, while methods like elastic trajectory deformation, see Section 2.1.2, entail more intricate algorithmic logic. Consequently, newer methods may solve some of the shortcomings of the original APF, but they are more specific and prone to failure in unforeseen environments. This progression seems logical, as the literal *low-hanging fruits* have likely been harvested.

I acknowledge that this might be a perceived bias, where important leaps of recent development will only crystallize in the future. Yet, this trend has been confirmed across many research areas, as recent analysis has shown a decrease in groundbreaking findings despite the exponential growth of publications (M. Park, Leahey, and Funk, 2023). For example, artificial intelligence models require increasing data and computational power to improve their performance (Narayanan et al., 2021), or computer chip manufacturing requires progressively greater resources (Bloom et al., 2020). It seems logical to assume that algorithm development in robotics is not excepted from

this trend.

To address the slowing pace of research, we propose the following areas of focus:

1. **Embrace Novel Algorithms:** Researchers should be encouraged to explore and develop novel algorithms, even if they initially underperform existing solutions. For instance, a novel obstacle avoidance algorithm may be constrained to specific environments or require more computational resources than established baselines. Embracing novelty can potentially lead to breakthroughs, and only subsequential, iterative development will adapt these algorithms to a wide range of scenarios, thereby realizing their full potential.
2. **Collaborative Research:** Collaboration among researchers should be actively promoted. Instead of celebrating the lone researcher who makes groundbreaking discoveries in isolation, we should prioritize collaborative efforts in algorithm development, implementation, and experimentation. The complexity of robotics, which merges diverse domains such as mechanical engineering, material science, electrical engineering, and computer science, demands a multidisciplinary approach. For instance, the field of motion planning encompasses various subtasks like environment recognition, environment understanding, global planning, local collision avoidance, and motor control, all of which may require specialized expertise. Acknowledging the limitations of individual researchers and advocating for collaborative endeavors can enhance problem-solving capabilities, enabling tackling challenges with high complexity.
3. **Benchmarking and Standardization:** The robotics community should work towards establishing benchmarks and standards that account for the diversity of real-world environments, facilitating more meaningful algorithm comparisons. However, as discussed earlier in this section, robotics as a multi-disciplinary brings many complex challenges, and defining a unifying benchmark is far from a trivial task.

9.3.3 Co-Design of Systems

Natural organisms developed during millions of years of evolution exhibit remarkable diversity, where the individual body parts are tailored to each organism's unique needs. From humans with their narrow but highly focused field of view adapted to hunting to herbivores with a panoramic vision to detect predators, nature's solutions are finely tuned to specific tasks. Conversely, insects are often limited to detecting rudimentary shapes and colliding with walls or windows. Nevertheless, each biological organism provides a sufficient vision system concerning their corresponding physic.

We believe a co-design can improve the performance of robots' vision, control, and hardware. The system can be more stable by creating algorithms tailored to the hardware. However, this is not a regression from a robot to a single-purpose machine but leads to a more robust and adapted robot.

9.3.4 Ethical Deployment of Robotics

The rise of robotics introduces transformative changes to the industry, the economy, and society. While it offers advantages such as addressing workforce shortages, particularly in an aging global population, and replacing *3D jobs* (dirty, dull, and dangerous), it also presents challenges. With the increased performance of robots, manual labor becomes less valuable. This might result in lower wages for manual workers or even the loss of jobs. This can lead to large corporations that generate huge amounts of wealth while only employing a minor workforce, similar to what has been observed with the corporate giants of the internet era. Additionally, robots combined with artificial intelligence in military use can lead to new weapons for mass destruction.

Research in this field must not turn a blind eye to the societal impacts of their development and leave the responsibility to politicians and regulators. Rather, they should acknowledge the potential consequences of these advancements. Researchers should actively advocate for safe and ethical deployment throughout the development process. Since these technologies have the potential to shape the future, we should aim for robots to help humanity and nature flourish.

A Dynamical System Modulation for Decentralized Collision-Free Coordination of a Swarms

Publication Note

The material in this chapter is adapted from:
(1) Conzelmann, F. M., L. Huber, D. Paez-Granados, A. Bolotnikova, A. Ijspeert, and A. Billard (2022). "A Dynamical System Approach to Decentralized Collision-free Autonomous Coordination of a Mobile Assistive Furniture Swarm". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*
(2) Douce, L.-N., A. Menichelli, L. Huber, A. Bolotnikova, D. Paez-Granados, A. Ijspeert, and A. Billard (2023). "Agent Prioritization and Virtual Drag Minimization in Dynamical System Modulation For Obstacle Avoidance of Decentralized Swarms". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Authorship Note: (1) and (2) were developed during the master projects by F. M. Conzelmann and L.-N. Douce, respectively. A. Menichelli finalized (2) during his work as a research assistant. The work was supervised by L. Huber, D. Paez-Granados, and A. Bolotnikova. L. Huber contributed additionally to the theoretical development and the software implementation.

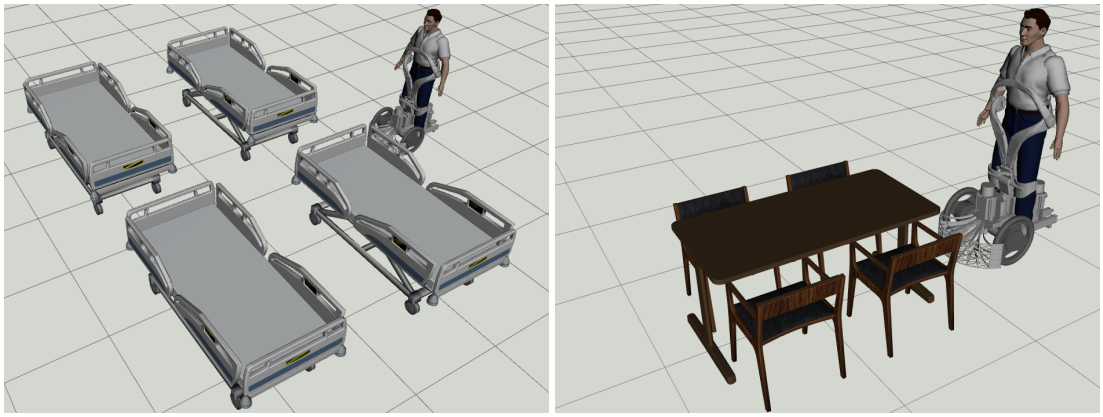
Source Code: https://github.com/epfl-lasa/autonomous_furniture/tree/main
Multimedia: Supplementary video can be found on:

https://youtu.be/F-u1U4Ugux4 https://youtu.be/D9_07xSKx_0

In order to facilitate and assist the indoor navigation of people limited mobility, the classically static objects in the environment, such as furniture, can be rendered mobile. The need for efficient and safe autonomous coordination of a mobile furniture swarm arises. In this Chapter, we augment the obstacle avoidance algorithm based on dynamical system modulation for a swarm of heterogeneous holonomic mobile agents. We present closed-form approach for mobile furniture obstacle avoidance and navigation within an indoor environment. The approach shows that each mobile furniture agent, defined by a polygonal surface, does not collide with any static or mobile obstacle (e.g., a person is moving around). All controllable mobile furniture converges towards a defined goal position and orientation. We further introduce a smooth prioritization to change the reactivity of the swarm towards the specific agents. A soft decoupling of the initial agent's kinematics is used to design an independent rotation control to ensure the agent reaches the desired position and orientation simultaneously. This decoupling allowed the introduction of a novel heuristic, the *virtual drag*. It minimizes the disturbance influence an agent has when moving through its surrounding. Additionally, the safety module adapts the velocity commands from the dynamical system modulation to avoid colliding trajectories between agents. We showcase the application of this algorithm in simulation on mobile furniture for smart environments. Results demonstrate that the proposed method can coordinate a swarm of mobile furniture to get out of the way of a mobile agent representing a person with limited mobility passing through the room while avoiding obstacles and converging towards a predefined target pose. Moreover, the prioritization successfully increased the minimum distance relative to a moving agent. The safety module is observed to create collision-free dynamics where alternative methods fail. Additionally, the repulsive nature of the safety module augments the convergence rate, thus making the proposed method better applicable to dense real-world scenarios.

A.1 Introdcution

Maneuvering around static objects in a non-adaptive and cluttered environment can be especially difficult for people with limited mobility, such as wheelchair users or elderly people. One approach to simplifying indoor mobility is to enable traditionally static furniture to become active agents. For instance, chairs could get out of the way and stack themselves automatically to clear the path. An armchair could also come close to an elderly to ease this person's comfort. Mobile furniture is a promising human-robot interaction area that has been gaining attention in recent years (Hauser et al., 2020; Stoddard, Giolando, and Knight, 2021). Additionally, autonomous furniture can be useful in workspaces where the displacement of furniture takes up an important part of the available time and effort, e.g., in health centers (Brooks et al., 2021; Fallatah et al., 2021), or hospitals (C. Wang et al., 2013). One of the main challenges of coordinating mobile agents while sharing space with humans is to guarantee safety and at the same time ensure to reach the assigned goal. It is paramount to ensure that no collisions occur despite the uncertainty of the environment. Furthermore, the mobile agents must coordinate to avoid blocking each other's path. Otherwise, the system will either be too dangerous for the involved people or useless for application.



(a) Hospital environment (b) Smart-home environment

Figure A.1: Sample of assistive environments, where the furniture are mobile agent facilitating a person's indoor mobility.

A general problem in mobile robotics is the navigation and coordination of multiple robotic agents. *Successful navigation* is defined as getting from a starting position to a goal position while safely avoiding all obstacles. Additionally, *successful coordination* is defined as multiple agents communicating to achieve a common goal without interfering with one another. This chapter explores a possible solution for coordinated motion within the framework of Dynamical Systems based on obstacle avoidance, a highly reactive approach to obstacle avoidance. We apply this concept to smart furniture to create assistive environments for people with limited mobility. Such environments would include hospitals with controllable beds, operating tables (Fig. A.1a), operating trays, or smart homes with mobile furniture (Fig. A.1b). The purpose is to develop new modular, distributed robotic systems to assist people with limited mobility in their daily lives, e.g., furniture moving out of the way of a person in a wheelchair, a table approaching a seated person.

This chapter aims to provide an approach to multi-robot coordination using obstacle avoidance based on dynamic system modulation (DSM). We reduce the disturbance in the environment each agent creates by introducing priority values, a soft decoupling of linear and angular velocity, the virtual rate, and active repulsion. The method eliminates observed collisions in dense environments and increases the convergence rate. The framework enables autonomous agents in an assistive living room environment, where the person does not have to adapt to the furniture, but it is the furniture that adapts to the person.

We enable multiple mobile agents to coordinate by augmenting control algorithms that allow dynamic collision avoidance for single agents (Figure A.2). This assures that agents can avoid the surroundings which we do not control, such as people in an assistive living environment. Furthermore, it allows full decentralization of decision-making and computing. The successful implementation of such an assistive environment relies on completing several projects (e.g., furniture and human localization, robust mobile furniture hardware and control) for the control

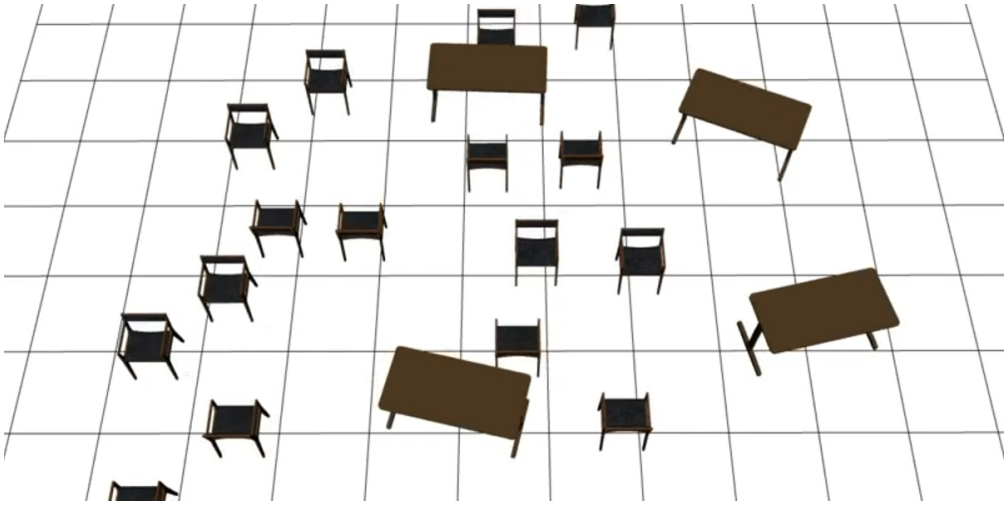


Figure A.2: Autonomous collision aware rearrangement of mobile chairs and tables in an assistive living environment.

solution to work. In this chapter, we assume that we can extract from the environment at any given time the position, speed, and heading of any furniture and person present in the environment. Furthermore, all agents are assumed to be holonomic and can follow a desired velocity exactly.

A.1.1 Contributions

In this Chapter, we use the dynamical system modulation (DSM) of Chapter 5 for point-agent obstacle avoidance to enable multi-robot navigation. Since the DSM has a short computation time and can navigate in environments with various shapes. This allows us to extend the algorithm in this work to swarm-navigation scenarios which reach beyond human crowds.

The contributions of this Chapter can be summarized as follows:

- Extension of the point-mass dynamical systems-based obstacle avoidance formulation to take into account the geometry of the agents with implicit orientation control (Sec. A.2).
- Creation of the dynamic attractor position for more reactive control and positioning (Sec. A.3).
- Prioritization of the agents to give each agent a specific importance (Sec. A.4)
- Soft decoupling of the agent's linear and angular velocity which allows for faster convergence and reduced environmental disturbance, referred to as minimization of the virtual drag (Sec. A.5 and Sec. A.6)
- Safety module ensure collision free motion.(Sec. A.7)

The long-term objective is to enable people with reduced mobility to take the shortest possible path, regardless of any obstacles present on their way. Hence, we propose a control solution in which the furniture stirs out of the way of an incoming person.

A.1.2 Notation

Due to the focus on mobile robot, this section specifically uses the vector $\boldsymbol{x} \in \mathbb{R}^N$ with $N = 2$ to describe the state of the system. The guiding dynamical system $\dot{\boldsymbol{x}}$ is used to describe the desired motion of an individual agent.

A.2 Agent Control Using Multiple Control Points

The algorithm in this section uses DSM to control holonomic agents by introducing multiple control points, it will be referred to as HDSM (Holonomic-DSM).

Smart furniture is defined as agents in the room: they do not move, unless re-arranging. However, they get pushed away when an incoming agent heads toward them. Furthermore, each furniture agent has an omnidirectional behavior and a corresponding shape, which is used to ensure safe collision avoidance. They are volumetric and often represented by multiple control points (Fig. A.3).

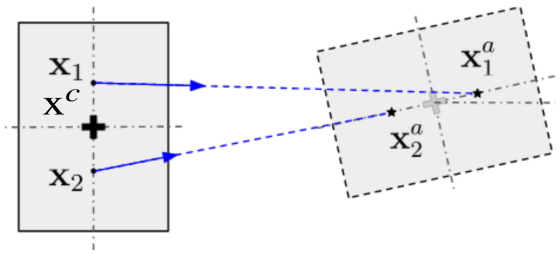


Figure A.3: Each control point of the furniture $\boldsymbol{x}_{(\cdot)}$ (black dots) has a velocity toward its attractor $\boldsymbol{x}^a_{(\cdot)}$ (black stars). The blue arrows are the initial dynamics $\boldsymbol{f}(\boldsymbol{x})$.

A.2.1 Optimal Control Points

Let us introduce *control points* which are placed within the obstacles. The control points are used as the positions where the dynamical system is evaluated. They have a margin m^c, tuned such that the control points cover the furniture. This allows the creation of a collision-free motion of the rigid bodies.

Many human-made furniture have rectangular shape with two major axis $\boldsymbol{a} = [a_1\ a_2]$. For this general shape, we want to place each control point $\boldsymbol{x}_i^c$ within the furniture, while ensuring that any boundary point is within the margin, i.e., $\min_{i=1}^{N^c} \|\boldsymbol{x}^b - \boldsymbol{x}_c^i\| \leq m^c\ \forall \boldsymbol{x}^b$. The optimal margin

which ensures this, can be written as

$$m^c = \sqrt{\sum_{i=1}^{N} \left(\frac{a_i}{1 + N_i^c} \right)^2} \tag{A.1}$$

where $N = 2$ is the number of dimensions, and $N_i^c \in \mathbb{R}_+$ is the number of control points of axis i. Approximating a rectangle with a finite number of circular shapes necessarily leads to *buffer areas* around the furniture (see Fig. A.4). This area represents an additional margin in specific directions. This buffer can be reduced by increasing the number of control points. However, the computational cost increases linearly with the number of control points.

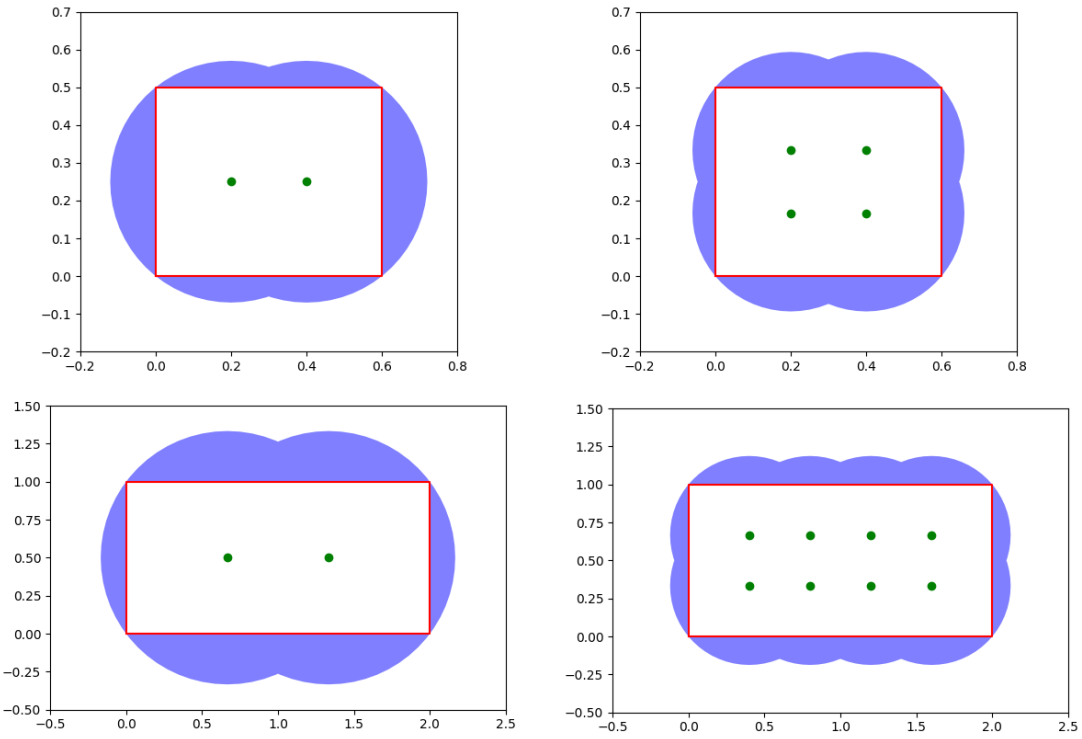


Figure A.4: The effect of increasing N^c on the margin size and buffer area. Green dots: control points. Red rectangle: size of mobile furniture. Blue area: buffer areas

Orientation Control

Multiple control points in the furniture can be coupled to obtain the linear velocity $\boldsymbol{v}_f \in \mathbb{R}^N$ and angular velocity $\omega_f \in \mathbb{R}$ of the furniture f. The coupling is computed as follows:

$$\boldsymbol{v}_f = \sum_{c=1}^{N^c} \mathrm{w}_c \boldsymbol{v}_c \quad \boldsymbol{\omega}_f = \sum_{c=1}^{N^c} \mathrm{w}_c (\boldsymbol{x}_f - \boldsymbol{x}_c) \times (\boldsymbol{v}_c - \boldsymbol{v}_f) \tag{A.2}$$

where $\boldsymbol{x}_c$ is the position of control point c, and $\boldsymbol{x}_f$ is the center of the furniture f. The desired velocity at each control point $\boldsymbol{v}_c$ is obtained using the DSM given in (3.13). The dynamics weights w_c are further described in Sec. A.2.2.

A.2.2 Dynamic Weights

The gamma function $\Gamma(\boldsymbol{x})$ as defined in (3.7) is used to estimate the proximity of each point to the surrounding obstacles. It is used to calculate the dynamic weights $\mathrm{w}_c \in [0, 1]$ of the control points:

$$\hat{\mathrm{w}}^c(\boldsymbol{x}) = \begin{cases} \left(\dfrac{\Gamma^c - \Gamma^{\min}}{\Gamma(\boldsymbol{x}) - \Gamma^{\min}} - 1 \right) \dfrac{\gamma^f}{1-\gamma^f} \dfrac{1}{N^c} & \text{if } \Gamma < \Gamma^c \\ \dfrac{1}{N^c} & \text{otherwise} \end{cases}$$

where $\Gamma^c > 1$ is the gamma cutoff value, which sets the distance at which an obstacle starts to influence a control point, $\Gamma^{\min} = 1$ is the lower bound of the gamma function, and γ^f is a scaling weight, which we set as $\gamma^f = 0.5$. The weights are further normalized as:

$$\mathrm{w}_c = \frac{\hat{\mathrm{w}}_c}{\sum_{c=1}^{N^c} \hat{\mathrm{w}}_c} \tag{A.3}$$

A simulation with two control points in one rectangular agent can be seen in Fig. A.5.

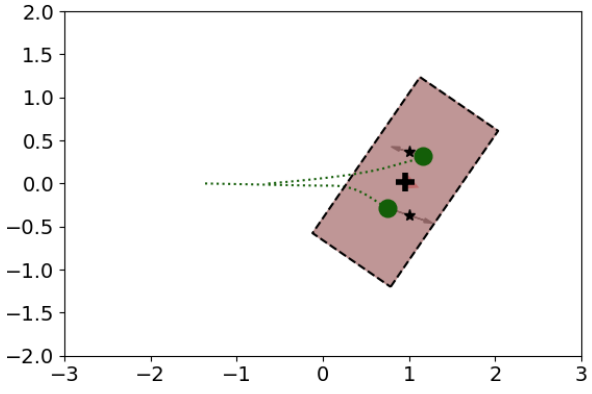
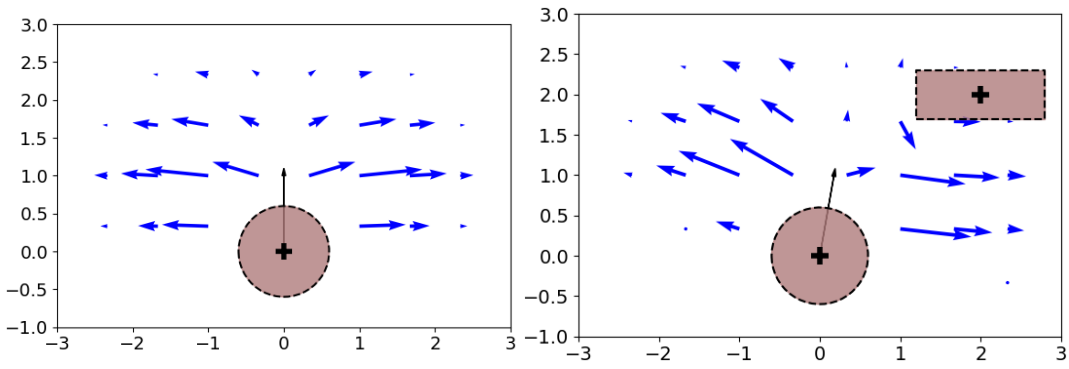


Figure A.5: The rotation is automatically induced on the rectangular furniture (brown) by using two separate control points (green). The desired position (black stars) is 90° rotated compared to the initial position. The black cross marks the reference point.

A.3 Dynamic Attractor Motion

The furniture only stops its movement when reaching the attractor. Hence, if a piece of furniture is being pushed away by a passing person, by default, it will move back to the initial position. In many cases, this moving back is not desired. Since for an autonomous environment, there is no need to move back to the initial condition. Furthermore, the next person might be passing the same trajectory; hence there is no advantage of going back to the initial position.

We introduce a dynamic displacement of the attractor, which guides and keeps the mobile furniture to new resting positions. The motion of the attractor is based on the DSM.



(a) $\boldsymbol{v}^a$ with no obstacle

(b) $\boldsymbol{v}^a$ with obstacle

Figure A.6: Visual representation of attractor dynamics $\boldsymbol{v}^a$ (blue arrows) in the presence of a person (brown circle) moving with a velocity $\boldsymbol{v}^p$ (black arrow) through the environment (a), in presence of an obstacle the dynamics are modulated to stay in free space (b).

A.3.1 Adaptive Repulsive Field

Let us consider the scenario where a person with velocity $\boldsymbol{v}^p$ at position $\boldsymbol{x}^p$ is passing the room. Firstly, the importance factor of the person evaluated as:

$$c^p = \langle \boldsymbol{d}^p(\boldsymbol{x}), \boldsymbol{v}^p \rangle \tag{A.4}$$

where $\boldsymbol{d}^p(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}^p$ is the relative position at point $\boldsymbol{x}$. The Gamma function $\Gamma(\boldsymbol{x})$ is then used to estimate the repulsion weight as follows

$$w^r(\boldsymbol{x}) = \begin{cases} -\Gamma^r \Gamma(\boldsymbol{x}) + c^p(1 - \Gamma^r) & \text{if } c^p(\boldsymbol{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.5}$$

where $\Gamma^r = c^p(\boldsymbol{x})/(\Gamma^c - 1)$ and $\Gamma^c > 1$ is the cutoff value, it indicates the Γ-value at which the repulsion dynamical system has no effect anymore. The repulsion weight $w^r(\boldsymbol{x})$ defines the repulsion of the attractor in the presence of a moving person.

We project it into the reference frame, to extract the corresponding speed value for the furniture's attractor:

$$\boldsymbol{v}^a(\boldsymbol{x}) = w^r(\boldsymbol{x})\boldsymbol{E}^p(\boldsymbol{v}^p)\boldsymbol{A}\left(\boldsymbol{E}^p(\boldsymbol{v}^p)\right)^T \frac{\boldsymbol{u}^d}{\| \boldsymbol{u}^d \|} \tag{A.6}$$

We construct the $\boldsymbol{E}^v$ and $\boldsymbol{A}$ matrices as follows:

$$\boldsymbol{E}^v(\boldsymbol{v}^p) = \begin{bmatrix} \boldsymbol{v}^0 & \boldsymbol{e}^p \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & \lambda^a \end{bmatrix} \tag{A.7}$$

where $\boldsymbol{v}^0 = \boldsymbol{v}^p / \| \boldsymbol{v}^p \|$ is the normalized velocity vector of the obstacle, and $\boldsymbol{e}^p$ is perpendicular to $\boldsymbol{v}^0$.

The repulsive force perpendicular to $\boldsymbol{u}^d$. is defined by $\lambda^a > 1$, the repulsion eigenvalue (we choose $\lambda^a = 10$). The resulting velocity of the furniture's attractor (Fig. A.6a).

Collision Free Motion of Virtual Obstacles

In a multi-furniture environment, as the dynamic attractor is repulsed from the incoming person, it might enter another furniture. Even though this is only a virtual collision, this would result in the furniture approaching each other closely before active avoidance happens. Hence, we introduce an approach to actively avoid other furniture by ensuring that the attractor stays in free space. For this, we use the modulation of the dynamical system as introduced in (3.13) to ensure collision-free motion of the attractor:

$$\dot{\boldsymbol{x}}^a = \boldsymbol{M}(\boldsymbol{x}^a)\boldsymbol{v}^a(\boldsymbol{x}^a) \tag{A.8}$$

An example of the attractor motion around an obstacle can be seen in Fig. A.6b.

A.3.2 Parking Zones

While the furniture is set to move out of the way when approaching a person, it should not stay unstructured in the room after the passing. For this, we introduce *parking zones*, which are dedicated regions and formations at which the furniture will stop.

The switch between the idle, avoiding, and reformatting behavior of a piece of furniture is generated as a cyclic state machine and controlled by the two parameters, as illustrated in (Fig. A.7. A switch from idle to avoiding happens when a person is moving towards the furniture piece, i.e., $c^p > 0$. The furniture piece or group thereof avoids the person and starts regrouping when $c^p < 0$. Each piece of furniture goes back to an idle state once the furniture has reached its attractor and has zero velocity, i.e., $\boldsymbol{v}_f \approx 0$.

We introduce two scenarios for the parking zone. The first maintains the original furniture

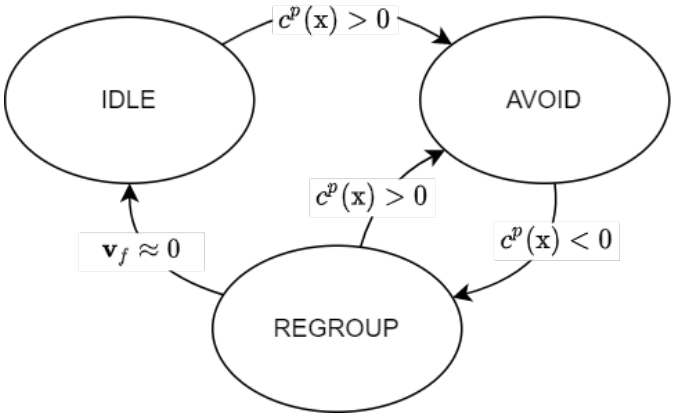


Figure A.7: The movement of the furniture is activated when the agent is moving towards it, i.e., $c^p(\boldsymbol{x}) > 0$ and goes back to idle once it's desired velocity reaches zero.

formation, such that the furniture is displaced as a group; e,g. a table and chairs move while preserving their arrangement. The second aims at reorganizing the furniture formation to be compact for space-efficient storage, see Fig. A.8

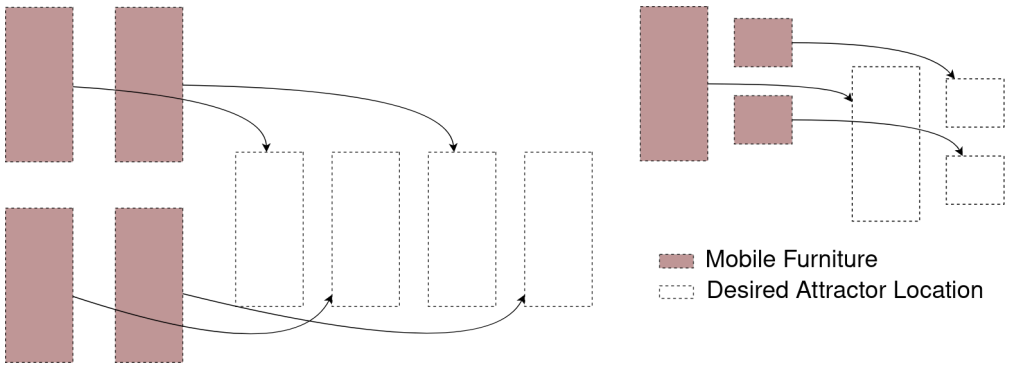


Figure A.8: The parking zones are designed to minimize occupation space (left) or stay in the original arrangement (right).

A.4 Prioritization of the Agent

An agent treating all obstacles in the surrounding equally might have undesired effects, e.g., an elderly person might get irritated when a robot gets close, while a chair would not be disturbed. Hence, we propose to use HDSM (see Section A.2) to adapt its trajectory based on the *priority* of the obstacles it encounters. In particular, we want the agents to prioritize trajectories staying further away from people to improve their safety.

The prioritization of an agent's safety distance (A.9) can be directly derived from the formula of the eigenvalues proposed in (3.18) by introducing $\gamma^s \in \mathbb{R}_+$, the priority of the agent (self), and $\gamma^o \in \mathbb{R}_+$, the priority given to an obstacle (other). The priority values are used in the calculation

of the eigenvalues as:

$$\lambda^r(\boldsymbol{x}) = 1 - \left(\frac{1}{\Gamma(\boldsymbol{x})}\right)^{\gamma^s/\gamma^o}, \quad \lambda^e(\boldsymbol{x}) = 1 + \left(\frac{1}{\Gamma(\boldsymbol{x})}\right)^{\gamma^s/\gamma^o} \tag{A.9}$$

We can verify that if $\gamma^s \gg \gamma^o$ then $\mathbf{D}(\boldsymbol{x}) \approx \mathbb{I}$, as defined in (6.7). Hence, if the self-priority of the agent is far greater than the obstacle's priority, its trajectory won't be modulated concerning this obstacle. Moreover, the new (A.9) does not break the necessary condition for impenetrability of the DSM given by (A.10), since we have:

$$\text{if} \quad \Gamma(\boldsymbol{x}) = 1 \quad \Rightarrow \quad \lambda^r = 0, \ \lambda^e = 1 \tag{A.10}$$

A.5 Soft Linear/Angular Control Decoupling

The coupling of the linear and angular velocity (A.2) results in the angular velocity being active only when close to the attractor $\boldsymbol{x}^a$, leading to delayed (angular) convergence. More flexibility of the angular control can be obtained by first computing the initial kinematics at the center of the agent, $\boldsymbol{x}^c$ (A.11) regarding the pose we want to reach (Fig. A.9).

$$\dot{\tilde{\boldsymbol{x}}}^c = -(\boldsymbol{x}^c - \boldsymbol{x}^a) \quad \text{and} \quad \omega = -(\phi - \phi^a) \tag{A.11}$$

The initial velocity, $\dot{\tilde{\boldsymbol{x}}}^c$, is modulated according to (3.13). The prior velocities $\boldsymbol{x}^c$ and ω are used to obtain the initial dynamics on the control points (A.12).

$$\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}}^c + \omega \times (\boldsymbol{x}_i - \boldsymbol{x}^c), \quad i \in \{1, .., N^c\} \tag{A.12}$$

where N^c is number of agent control points. The transformation is shown in Fig. A.9. Finally, the

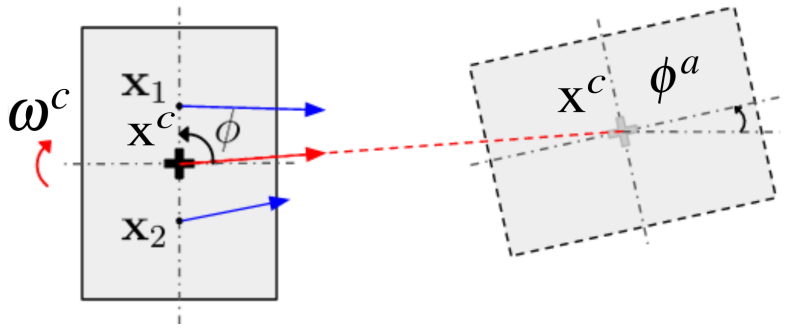


Figure A.9: The agent moves from its initial position (left) to the goal position (right). The prior linear and angular velocities (red arrows) are at the center of the agent $\mathbf{x}^c$ to reach the attractor position $\mathbf{x}^a$ and orientation ϕ^a. These can be transformed to get the control point velocities (blue arrows).

velocities at the control points are modulated by (3.13) and the agent's final linear and angular velocities are obtained by (A.2). This soft decoupling of kinematics brings more flexibility to the

orientation control and will be further exploited in the next section.

A.6 Minimization of The Virtual Drag

The drag of a moving object in a viscous fluid determines the disturbance of the surrounding particles. Similarly, the orientation of a swarm-agent moving through a clustered space affects the modulated velocity of its surrounding agents (Fig. A.10). Hence, we refer to the obstacle's effect on its neighboring agents as *virtual drag* and desire to minimize it to decrease the disturbance on its environment.

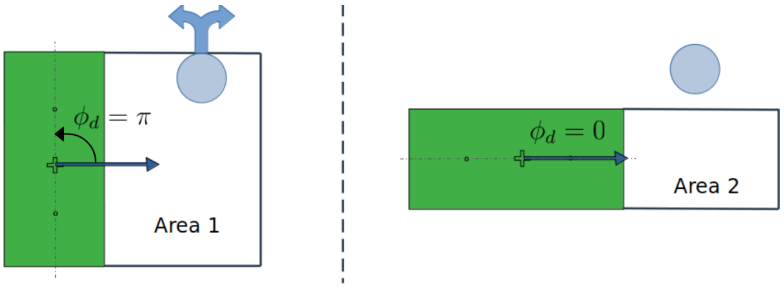


Figure A.10: The agent (green rectangle) needs to pass the blue circle (e.g., a person). If the agent moves perpendicular to its longest axis (left), the person must move out of the way. Whereas, if the agent has its longest axis parallel to the direction of the movement (right), the person does not get disturbed at all.

A.6.1 Drag Factor

The virtual drag produced by an agent depends on its shape and displacement direction. Circular agents have a constant virtual drag and are invariant to rotation. Depending on its orientation, a rectangular agent with a high length-to-width ratio produces a higher or smaller virtual drag. We introduce the drag factor μ in (A.13) to quantify this, where l and L are the shortest and longest axis lengths, respectively.

$$\mu = \frac{L}{l}, \qquad \mu \in [1, \infty) \tag{A.13}$$

A.6.2 Drag Angle

The drag angle $\phi^d \in [0, \pi]$ is defined as the angle between the direction of the linear velocity and the axis that minimizes the virtual drag produced by the agent. It is 0 for displacement which is in the direction parallel to the minimal drag (and thus has a minimal virtual drag), and π when the agent moves perpendicular to the minimal drag direction (and thus has maximal virtual drag), see Fig. A.10.

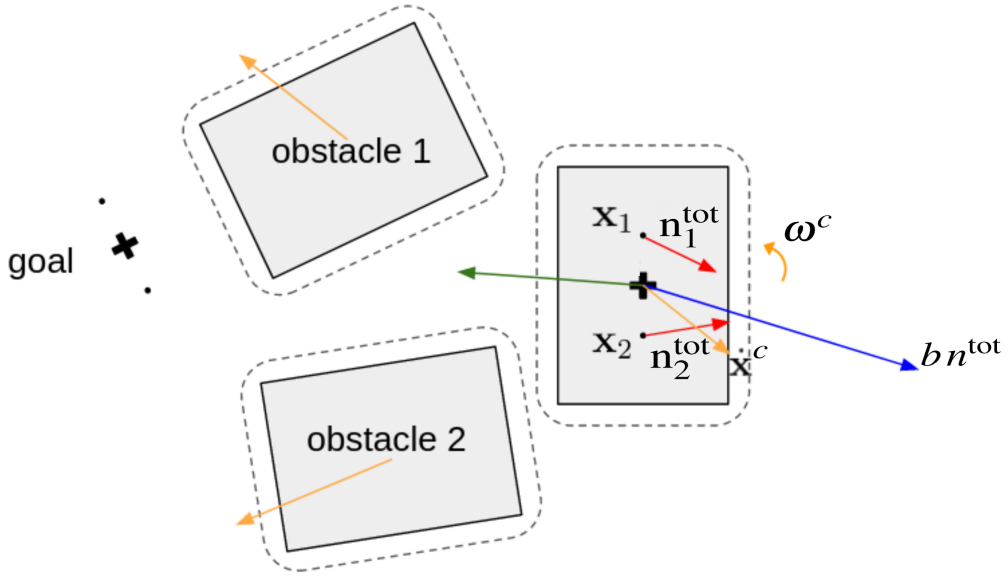


Figure A.11: Even though the planned trajectory (green arrow) is on a colliding path with the two obstacles, the safety module deviates the trajectory and avoids them safely (orange velocity). As all obstacles move away from each other, this leads to a *un-cluttering* of the space and has been observed to increase convergence.

A.6.3 Convergence to Attractor Orientation

Minimizing the drag is important when moving, but as we approach the attractor, the agent needs to approach its desired attractor orientation ϕ^a. This balancing can be quantified as:

$$\begin{aligned} \tilde{\phi} &= a_1\phi^d + a_2\phi^a \qquad \text{with} \\ a_1 &= \frac{1}{2}\frac{d}{d+k}(1+\tanh(\mu(d-\alpha)) \ , \quad a_2 = 1-a_1 \end{aligned} \tag{A.14}$$

where d is the distance from the agent to the goal position, $\mu \in \mathbb{R}_+$ is the drag factor, and $\alpha \in \mathbb{R}_+$ is a parameter that defines the distance to the goal at which $a_1 \leq a_2$, i.e., the agent starts rotating towards the goal orientation. The tuning parameter $k \in \mathbb{R}_+$ is used in $d/(d+k)$ and ensures that a_1 converges to zero at the goal position $d = 0$. The resulting desired angle $\tilde{\phi}$ is used in the initial dynamic (A.11) to obtain the initial angular velocity ω. This algorithm is referred to as drag-HDSM.

A.7 Safety Module

Highly dense scenes may lead to situations that cause non-convergence that drag-HDSM cannot resolve. For avoidance in cluttered environments with multiple overlapping obstacles, we introduce a *safety module* to ensure to deviate any colliding trajectory. The safety module becomes active in the *critical* region around the obstacle, where it calculates the avoidance

kinematics. This space is defined by the critical distance Γ^c and depends on the distance to the goal d:

$$\Gamma^c(d) = \begin{cases} \Gamma^{\max} & d > d^c \\ \Gamma^{\min} + \frac{d}{d^c}\left(\Gamma^{\max} - \Gamma^{\min}\right) & 0 \leq d \leq d^c \end{cases} \tag{A.15}$$

The lower bound is set as $\Gamma^{\min} > \Gamma(\boldsymbol{x}^a)$, i.e., the distance value at the attractor position. This ensures that the agent stays reactive at the goal position. Note that, if $d < d^c$, Γ^c shrinks linearly from $\Gamma^{\max}$ to $\Gamma^{\min}$.

The direction to avoid collisions, $\boldsymbol{n}_{\text{tot}}$, is computed as a weighted average of all the normal directions $\boldsymbol{n}_{\text{o}}$ of all surrounding obstacles $o \in \{1, .., N^{\text{obs}}\}$ for which we have $\Gamma^o(\boldsymbol{x}^c) < \Gamma^c$. The weighted normal is evaluated as:

$$\boldsymbol{n}^{\text{tot}} = \sum_{o=1}^{N^{\text{obs}}} w_o \boldsymbol{n}_{\text{o}} \quad \text{with} \quad w_o = \frac{1/\Gamma_{\text{o}}}{\sum_{o=1}^{N^{\text{obs}}} 1/\Gamma_{\text{o}}} \tag{A.16}$$

Elevating the basic velocity in (A.2), the safe linear velocity at the center $\dot{\boldsymbol{x}}^c$ at the center is defined as:

$$\dot{\boldsymbol{x}}^s = \sum_{i=1}^{N^{\text{c}}} w_i \dot{\boldsymbol{x}}_i + b\boldsymbol{n}^{\text{tot}} \tag{A.17}$$

where $b = (\Gamma^c - 1)\left(\min_o \gamma_o - 1\right)$ defines the strength of the safety response depending on the closest obstacle and the critical space.

Similarly, the angular velocity ω is updated using the parameter b and the correction term ω^c based on the avoidance direction $\boldsymbol{n}_i^{\text{tot}}$ for each of the control points $i \in \{1, .., N^c\}$ control and their position $\boldsymbol{x}_i$

$$\begin{aligned} \omega^s &= \sum_{i=1}^{N^{\text{c}}} w_i \left(\boldsymbol{x}^c - \boldsymbol{x}_i\right) \times \left(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}^c\right) + b\omega^c \\ \text{with} \quad \omega^c &= \sum_{i=1}^{N^c} w_i \|\boldsymbol{x}_i \times \boldsymbol{n}_i^{\text{tot}}\| \end{aligned} \tag{A.18}$$

where $\boldsymbol{n}_k^{\text{tot}}$ as described earlier in this section is taking only into account the obstacles in the critical space of the i-th control point. When approaching the limit of the avoidance parameter, i.e., $\min_o \gamma^o \to 1 \;\Rightarrow\; b \to \infty$ as defined in (A.18), Consequently, the linear velocity $\boldsymbol{x}^c$ and angular velocity ω need to be stretched to have a magnitude which is within the desired bounds. Figure A.11 shows the effect of the safety module. The safety module is combined with an emergency stop that sets $\dot{\boldsymbol{x}}^c = \boldsymbol{0}$ and $\omega = 0$ if any control point i we have $\Gamma(\boldsymbol{x}_i) \leq \Gamma^{\text{stop}}$. This algorithm will be referred to as safe-HDSM.

A.8 Results

A.8.1 Collision Avoidance with Human Disturbance

The algorithm was implemented in a simulation environment to explore its capabilities in different scenarios and environments. Robot Operating System (ROS)[1] was used for the simulation and visualization of the environment.[2] The setup is designed for future portability and experimental validation with real furniture and a human pilot.

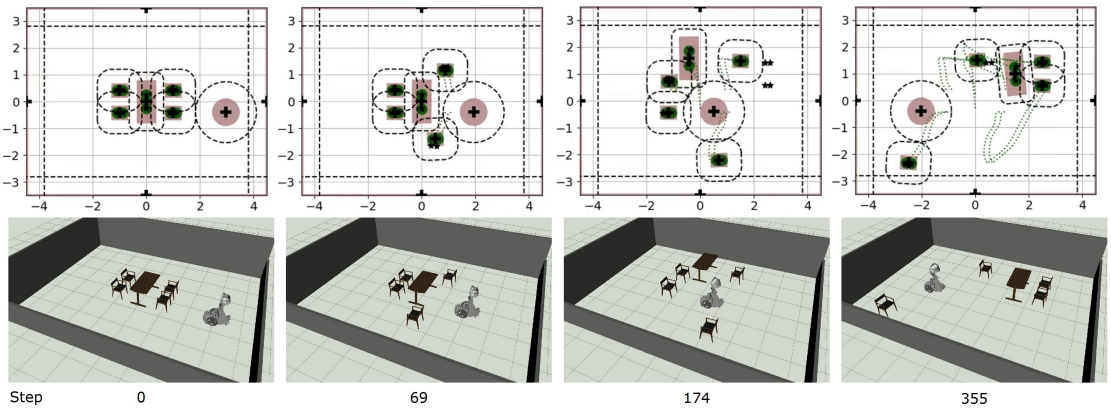


Figure A.12: A simulation run in a smart-home environment. Brown cuboid: mobile furniture. Brown disc: person. Brown frame: walls of the environment. Green point: control point of mobile furniture. Black star: attractor of mobile furniture. Dashed black line: margin

The environments include static obstacles (walls), a mobile furniture swarm (whose motion is regulated by our proposed method), and a non-controlled mobile agent representing a person passing through the room along a straight-line trajectory. The *main agent* in the simulation is Qolo Paez-Granados et al., 2022, a device for passive assistance for standing mobility.

Snapshots of two simulations are shown in Fig. A.12 and Fig. A.13. The mobile furniture successfully starts moving when the incoming agent gets close and safely avoids the agent and the furniture. Once the agent has passed through the immediate vicinity of the furniture, it moves towards the parking area and successfully regroups.

In Fig. A.14a, the distance of the furniture relative to the person is shown. We observe that the mobile furniture never gets close enough to the person for any possible collision, i.e., below the collision threshold.
In Fig. A.14b, the linear velocity increases continuously when the person is getting closer, and the mobile furniture starts to move away. As the velocity starts to slow down, the attractor location is moved to the desired parking area, and a spike in velocity is generated. In Fig. A.14c, The magnitude of the angular velocities are never greater than 0.16 rad/s. This indicates a smooth

[1] ROS2, https://docs.ros.org/

[2] The implementation can be found under https://github.com/epfl-lasa/autonomous_furniture.git

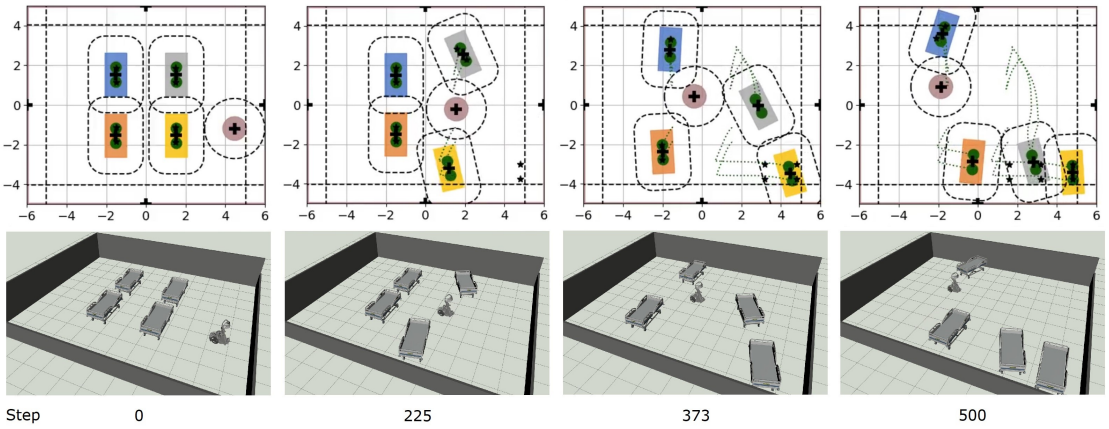


Figure A.13: A simulation run in a hospital environment. Multi-colored cuboid: mobile furniture. Brown disc: person. Brown frame: walls of the environment. Green point: control point of mobile furniture. Black star: attractor of mobile furniture. Dashed black line: margin

rotation resulting from an optimal control points placement. A spike of the angular velocity can also be detected as the attractor is moved to the parking zone.

A.8.2 Agent Priority

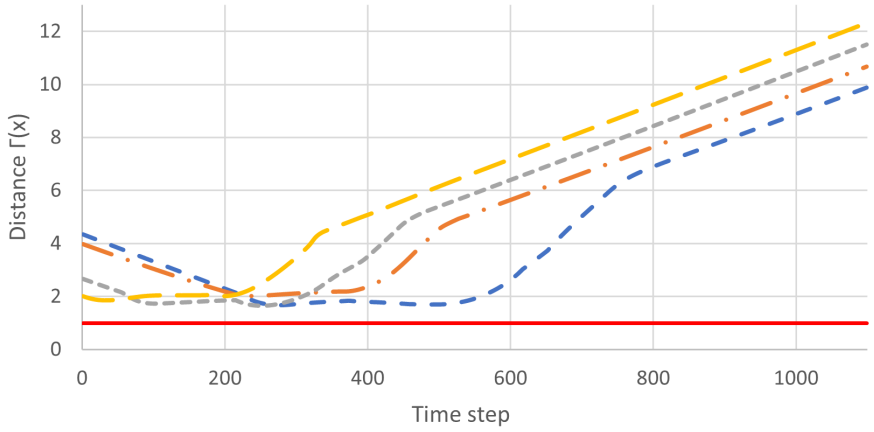
We first analyze the influence of the priority value γ^s in a scenario where a mobile agent moves between two static agents (Fig. A.15a). In the first setup, all agents have the same priority of $\gamma^s = 1$; in the second setup, we give the circular agent (interpreted as an elderly person) a higher priority, thus having $\gamma^s_{\text{circ.}} = 10^3$, $\gamma^s_{\text{rect.}} = \gamma^s_{\text{sqr.}} = 10^{-3}$.

Qualitatively, we can see that the blue agent is making a larger avoidance trajectory around the yellow circle (human) when the latter has a higher priority. Furthermore, the minimum distance between the human (yellow circle) and the mobile agent (blue) decreases by 20% when the human has a higher priority.

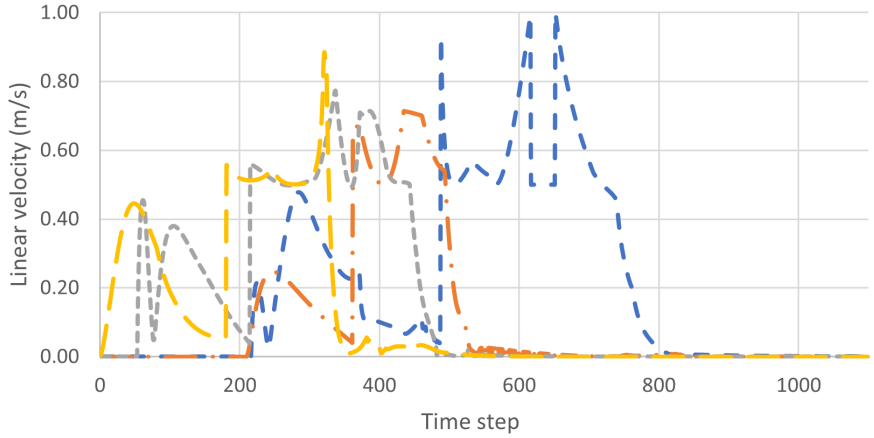
A.8.3 Drag Minimization

The second comparison scenario includes a mobile agent, which passes between two passive agents (Fig. A.16). In this case, the two passive agents can move out of the way if necessary, but they only do so if they are *disturbed*.

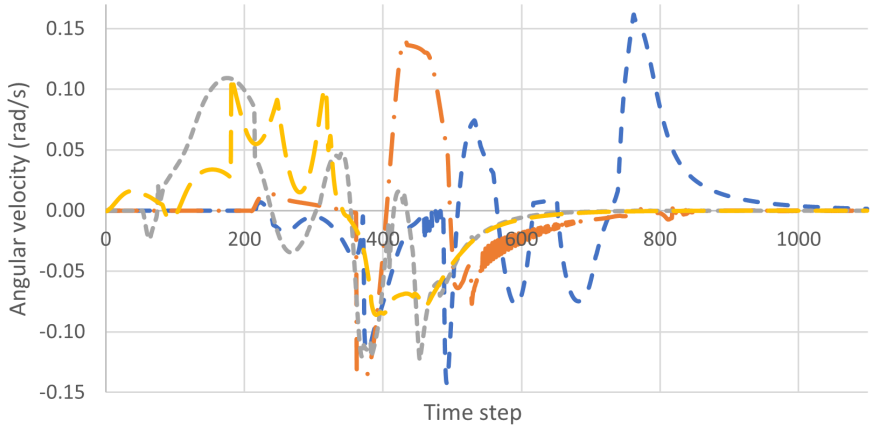
Using drag-minimization, the agent rotates before crossing the narrow passage. Hence, it has less disturbance to the person compared to the situation of the drag not being active. This can be interpreted as the furniture agent performing its trajectory with increased environmental awareness.



(a) Distance of furniture w.r.t. the person and collision threshold (solid red line)

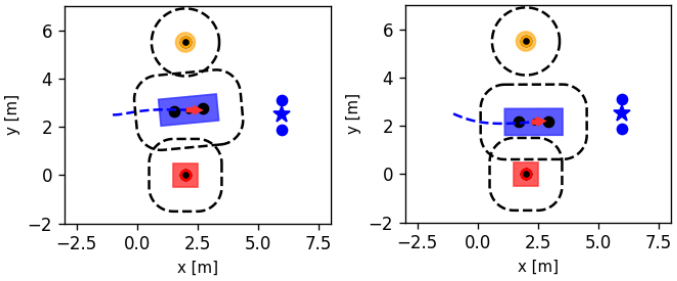


(b) Linear velocity of the furniture

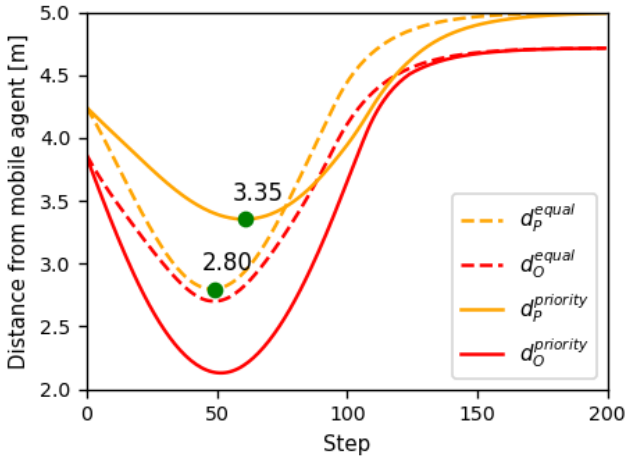


(c) Angular velocity of the furniture

Figure A.14: The distance, linear velocity and angular velocity of the experimental run seen in Fig. A.13. Each data line represents the same colored furniture. Dashed blue line: top left furniture. Short dashed grey line: top right furniture. Dash dotted orange line: bottom left furniture. Long dashed yellow line: bottom right furniture.



(a) Equal priority $\gamma^s = 1$ (b) $\gamma^s_{\text{circ.}} \gg \gamma^s_{\text{rect.}} = \gamma^s_{\text{sqr.}}$



(c) Distance from the mobile agent to the person (yellow) and the static obstacle (red) for the case with equal priority (dashed) and higher prioritization of the person (solid).

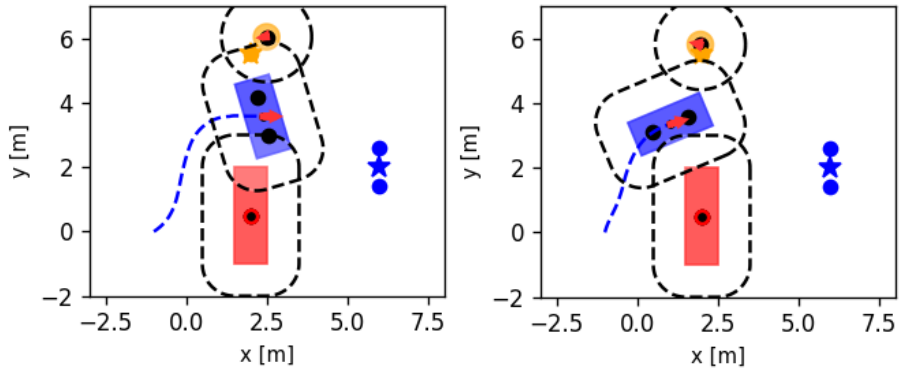
Figure A.15: The mobile agent (blue) passes between the red rectangle (e.g., a chair) and the yellow circle (e.g., a person) when controlled by drag-HDSM. When the priority is equal (a), the agent passes in the middle between the others, see (c). Conversely, when the person has higher priority (b), the mobile agent's trajectory stays further away from the person.

A.8.4 Quantitative Comparison

We set he hyper-parameters as $\Gamma^{\text{stop}} = 1.1$, $\Gamma^{\text{min}} = 1.2$, $\Gamma^{\text{max}} = 2$, $d^c = 1$, $\alpha = 1.5$ and $k = 0.01$.

Metrics

Distance Traveled, $\overline{\mathscr{D}}$: Ideally, the trajectories are closest possible to the shortest distance to the goal. To assess the method performance in terms of how close the final trajectories are to the ideal case, we use the mean relative distance metric. The relative distance is the ratio of the total distance d_i traveled by an agent to the shortest distance D_i. $\overline{\mathscr{D}}$ is the mean relative distance made



(a) HDSM (b) drag-HDSM

Figure A.16: The agent (blue) moves to its goal on the right. The red rectangle (mobile furniture) and the yellow circle (e.g., human) can move but desire to stay in their current position. With the HDSM (baseline), the person (yellow circle) has to move further out of the way than when using drag-HDSM.

by all the N^a agents during a scenario.

$$\overline{\mathscr{D}} = \frac{1}{N^a} \sum_{i=1}^{N^a} \frac{d_i}{D_i} \tag{A.19}$$

Virtual Collisions Rate, $\mathscr{C}$ A virtual collision is registered when an agent's control point breaches another agent's margin. Here, agents are stopped before a collision happens.

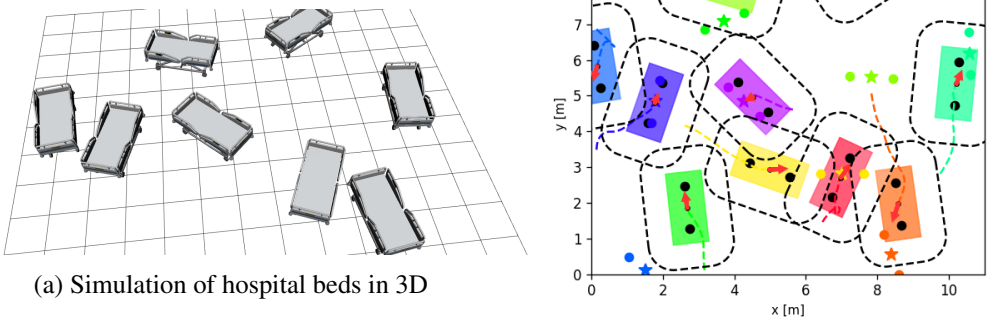
Setup

The proposed method and the baseline were compared quantitatively by generating 100 random scenarios for 3 to 10 agents. In each scenario, the agent's initial and goal positions are randomly chosen inside the spawn area without overlapping. The size of the spawn area of 11m × 9m, the agents are hospital beds which need to be rearranged of size 2m × 1m (see Fig. A.17). No walls constrain the area, and the agents can move outside this area if needed. However, such behavior negatively affects the distance traveled score.[3]

The swarm using the algorithm with virtual drag minimization (drag-HDSM) shows a reduction in the mean relative distance traveled by all the agents compared to the baseline (HDSM). This effect increases with the density of environments (Fig. A.18 top plot). In scenarios with 10 agents, the median reduction is 10%. HDSM is expected to use a longer path because the agents disturb each other on the way.

The convergence rate decreases for all algorithms with an increasing number of agents (Fig. A.18 middle plot). The virtual drag (drag-HDSM) improves convergence compared to the baseline

[3]Source code on https://github.com/epfl-lasa/autonomous_furniture



(a) Simulation of hospital beds in 3D

(b) Algorithmic representation of beds

Figure A.17: 100 random scenarios for different numbers of hospital beds were simulated (here, ten beds).

(HDSM). Since minimizing the drag makes the obstacles align with their direction of motion, they present less surface to get stuck with each other. The safety module (safe-HDSM) shows the biggest improvement in convergence. When obstacles get close to each other, the safety module pushes them away so that they get untangled and converge toward their attractors. For safe-HDSM, even with 10 agents, almost 50% of the scenarios converge towards the attractors, compared to 0% for the other two algorithms.

The baseline method (HDSM) often makes agents collide when trying to reach the goal, and already with three agents, an average of around one-sixth of the simulations collide (Fig. A.18 bottom plot). This increases drastically with the number of agents: with nine agents, almost 100% of the simulations result in collisions. The virtual drag (drag-HDSM) decreases the number of colliding scenarios. The effect is more pronounced with fewer agents, and we observe almost no collision for simulations with three obstacles. The greatest contribution comes from the safety module (safe-HDSM), which reduces the ratio of virtual collisions to 1% or less for all simulations.

A.9 Conclusion

We have presented a comprehensive algorithmic solution for decentralized multi-agent control: the Holonomic Dynamical System Modulation (HDSM). Our algorithm calculates the real-time avoidance velocities based on the state surrounding obstacles and agent. The algorithm inherits the flexibility of the Dynamic System Modulation (DSM) framework which renders it suitable for environments with constant change. By utilizing multiple control points volumetric agent, the algorithm can avoid collisions with the environment independently of the agents shape.

The behavior of HDSM was improved by introducing features such as agent prioritization, soft decoupling of agent kinematics, minimization of virtual drag, and a safety module. These aug-

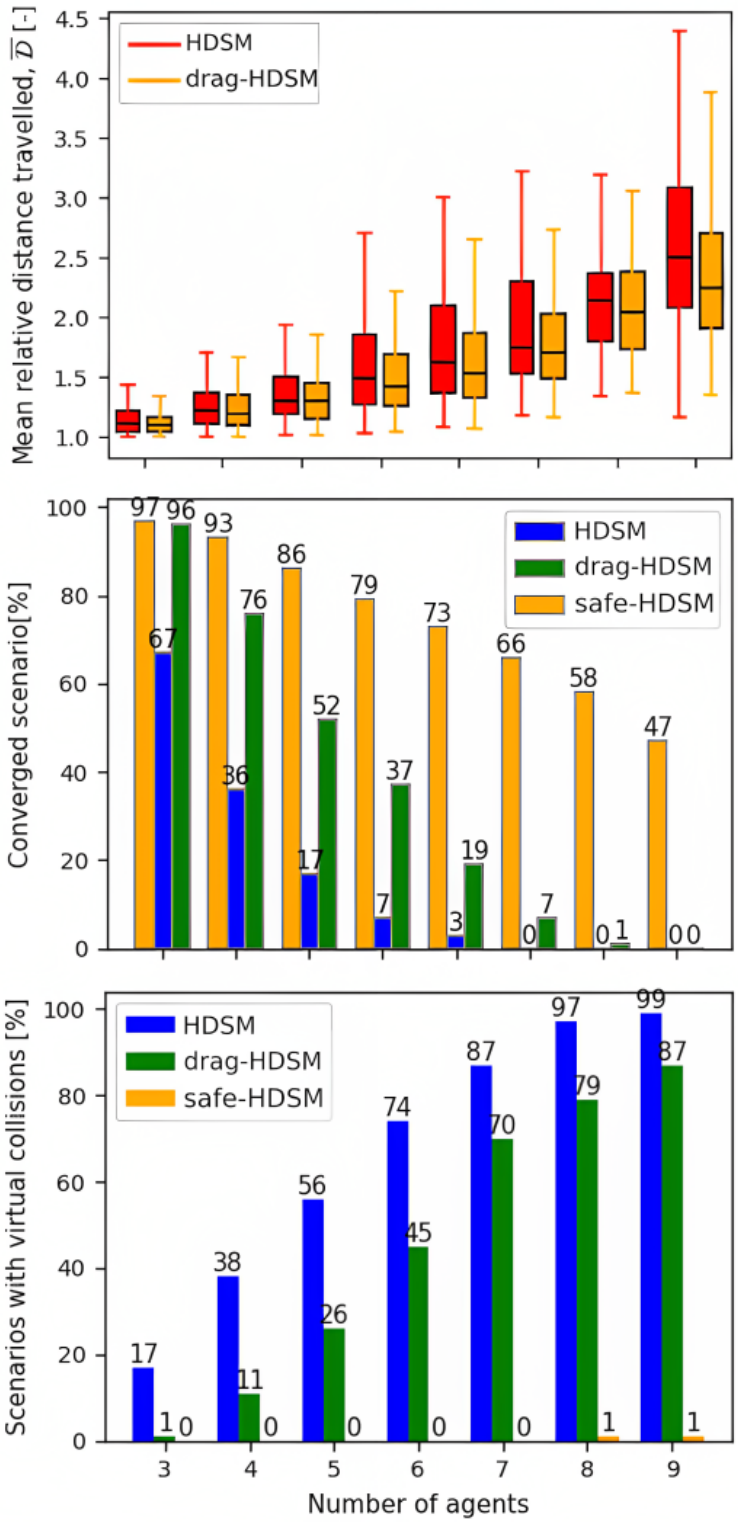


Figure A.18: The proposed algorithm performance comparison to the baseline (HDSM). Only no-collision scenarios were considered for the metrics in the top and middle plots.

mentations collectively enhance each swarm agent's environmental awareness. The prioritization mechanism, achieved through augmented eigenvalues in obstacle avoidance, increases safety around higher-priority agents by maintaining greater distances. Soft decoupling of linear and angular velocities, coupled with the minimization of virtual drag, leads to faster angular convergence and reduced disturbance within the environment. This becomes especially significant in densely populated spaces, where collision probability is notably reduced.

Finally, the algorithm was applied to simulated mobile furniture, defined for mobile used for assisted living environments. Such an environment offers a great potential for individuals with limited mobility. During qualitative evaluations, our control system demonstrated the ability to navigate safely around incoming individuals while successfully maneuvering between other mobile furniture agents to designated parking areas. The experimentation indicated that the incorporation of multiple control points within furniture not only averts collisions but also enables effective rotation in cluttered spaces. Quantitative results show, that the safety module decreases virtual collisions and greatly improves the agent's convergence toward goals. However, we acknowledge instances where agents can be blocked, necessitating further refinement, particularly in scenarios where one agent is significantly distant from its goal while its surrounding agents have already reached theirs.

In summary, our approach introduces advancements in decentralized multi-agent swarm navigation. The method presents itself for enhancing mobility and safety volumetric multi-agent navigation. The integration of these concepts holds promise not only in specific scenarios but also in broader human-operated settings.

A.9.1 Future Work

Our future work encompasses several key areas to further enhance the efficacy and practicality of the developed algorithms.

Firstly, while this work serves as a proof of concept, the ongoing hardware development for deploying a mobile furniture swarm is crucial. The subsequent phase involves evaluating the performance of this hardware in real-world scenarios.

The algorithm's current design focuses on collision avoidance within a two-dimensional representation of agents and their surroundings. Future endeavors will involve leveraging the three-dimensional geometrical shape of objects, enabling two-dimensional overlap within the context of collision-free three-dimensional bodies. Transitioning from 2D to 3D collision problems opens opportunities to better utilize available space, such as positioning lower furniture underneath higher ones. This advancement, while enabling more efficient space utilization, also presents challenges that will be addressed.

In relation to agent prioritization, systematic evaluations will be undertaken within assistive environments, involving potential users. This assessment will delve into the impact of prioritization

on agent disturbance, time taken to complete paths, and energy consumption.

The extension of DSM with multiple control points has demonstrated a reduction in convergence ratio, leading to a lower percentage of obstacles reaching their desired goals. To address this, we intend to explore a hybrid approach by combining obstacle avoidance methods with policy learning techniques, such as reinforcement learning. This fusion aims to capitalize on the strengths of both analytical and learning algorithms, enhancing overall system performance.

The algorithm has been evaluated qualitatively and quantitatively. Future work should investigate the algorithms potential theoretically. This will further allow to find the limitations of the algorithm, and hence open the door for improvements and iterations.

B Student Projects

Various student projects have directly or indirectly contributed to the work presented in this thesis. Some have led to publications in conference papers, whereas others have contributed to robotic implementations used for robotic experiments, or have been of an exploratory nature; extending the concepts to new applications. The summary of the projects are listed in chronological order.

Title: People Detection in Close-Proximity for Robot Navigation in Crowds based on 3D LiDAR Data
Author: Lara Brudermüller
Master Thesis (Spring, 2020)

LiDAR-based object detection is used in various applications, ranging from autonomous driving to service robotics and robot navigation. However, so far, the development of such detection pipelines has been focused on urban driving scenarios and less on human-centric indoor environments. Therefore, this work extends 3D end-to-end object detection pipelines to dynamic crowds in close-proximity to a mobile robot. Given as input 3D LiDAR data from humans moving in space, the detector, developed in this work, is able to detect humans in real-time. It is applicable to real-time collision avoidance algorithms on mobile robot platforms, as well as to real-time multi-object tracking tasks. To train and evaluate the detector, a new annotated dataset of individuals moving around a mobile robot in an indoor public space has been created using a 3D LiDAR sensor. The detection framework achieves an average precision (at IoU0.3) of 90.9% with an inference time of 0.01 seconds while the LiDAR sensor is running at 10Hz, showing a robust performance. Additional cross-validation on the JRDB benchmark (including indoor and outdoor scenes) resulted in an average precision of 27.99%. Yet, in a constrained setting to close-proximities and indoor-only, the detector is able to achieve an average precision of 63.6%.

Title: Swarm Furniture Obstacle Avoidance for Smart-Living Environment Assisting Wheelchair Navigation
Author: Federico M. Conzelmann
Master Thesis (Fall, 2021)

In order to facilitate and assist the indoor mobility of people with special needs, the classically static objects in the environment, such as furniture, can be rendered mobile. The need for efficient and safe autonomous coordination of a mobile furniture swarm arises. We present a closed-form approach for mobile furniture obstacle avoidance and navigation within an indoor environment. The approach shows that each mobile furniture agent, defined by a polygonal surface, does not collide with any static or mobile obstacle (e.g., a person is moving around). All controllable mobile furniture converges towards a defined goal position and orientation. We showcase the application of this algorithm in simulation on mobile furniture for smart environments. Results demonstrate that the proposed method can coordinate a swarm of mobile furniture to get out of the way of a mobile agent representing a person with limited mobility passing through the room while avoiding obstacles and converging towards a predefined target pose.

Title: A Combined Approach for Motion Learning and Obstacle Avoidance
Author: Christopher J. Stocker
Semester Project (Spring, 2022)

As human-robot interaction becomes increasingly prevalent through collaborative robots new safety requirements are needed to ensure the user's protection at all times. This means that robots need to adapt to human behavior, and not the other way around. To achieve this in human-inhabited environments they need to be robust to perturbances and be able to reevaluate in milliseconds its new path to avoid a collision.
This report analyzes the development of an experimental setup for full-body obstacle avoidance applied to a robot arm. The obstacle avoidance problem is adapted from a closed-form approach utilized to constrain flow within a given volume and around objects.
This approach ensures the flow converges and stops at a single fixed point. This technique can converge towards an attractor whilst avoiding both static and dynamic obstacles. This has never before been applied to a robot working in 3D space. For this reason, the experimental setup and simulation were tested on a virtual replica of the Franka Emika robot arm.

Title: Agent Prioritization and Virtual Drag Minimization in Obstacle Avoidance for Swarm Coordination
Author: Louis-Nicolas Douce
Master Thesis (Spring, 2022)

Efficient and safe multi-agent swarm coordination in environments where humans operate can serve many practical industrial and assistive purposes. In this paper, we extend the obstacle avoidance algorithm based on dynamic system modulation for swarm holonomic mobile agents by introducing a new feature that brings different levels of priority for the agents and a new heuristic in the displacement of the furniture, the minimization of the virtual drag. Prioritization of the agents has successfully displayed the adaptive behavior of the mobile agents regarding the priority of the other surrounding agents. It has enabled to enhance the safety of a selected agent by decreasing its proximity metric by 16%. Drag minimization has decreased both the mobile agents' proximity metrics and the number of scenarios where a virtual collision occurred.

Title: A Combined Approach for Obstacle Avoidance and Motion Learning
Author: Merih Ekin Özberk
Semester Project (Autumn, 2022)

In this project, we propose several approaches for incorporating a dynamic obstacle avoidance method into a dynamic system-based motion learning approach. The proposed approaches are tested in example trajectories in 2D with static and dynamic obstacles added to the environment. The resultant trajectories of the proposed methods are compared based on convergence, smoothness, and similarity to the reference trajectory. The final proposed approach, Trajectory Following with Local Rotations Around Obstacles, resulted in the most desirable trajectories from a qualitative analysis perspective in simulations in 2D. This approach applies an obstacle avoidance modulation on a velocity vector field that combines motion learning and local rotations around obstacles and successfully avoids obstacles while following the motion learning path. The final approach is evaluated in a simulation environment and on the 7DOF Franka Emika Panda robot for validation.

Title: Obstacle Aware Passive Control for Dynamical Systems
Author: Trinca Thibaud
Semester Project (Autumn, 2022)

Recent work on dynamical systems (DS) has enabled new ways of robot control. DS can be learned with a variety of methods and can often be modified such that all points in space converge to the attractor, making the system more robust to noise or disturbances. One can use DS modulation to consistently avoid obstacles, with solid guarantees of impenetrability. To transfer from DS velocity control to torque control, passive control theory is a powerful method to accurately track the desired speed, while having the freedom to be compliant. In this work, we present a method that extends the concept of passivity. The method maintains compliant behavior away from obstacles but increases the stiffness of the control when approaching one of them, allowing the robot to be aware of its environment. The proof of concept was tested in a simulation in Python. The robot shows great tracking performance and successfully rejects disturbances that would have led to a collision with an obstacle.

Title: Obstacle Aware Motion Learning for Human-Like Navigation in Crowds
Author: Bryan Kheirallah
Semester Project (Spring, 2023)

The goal of this project is to implement a new algorithm intended to help a robot navigate autonomously through moving crowds. For a robot agent intended to be used around human crowds, the navigation principle shall not only rely on obstacle avoidance, and it would be beneficial to mimic human behavior to allow the robot to blend in the crowd. To answer this challenges, the developed algorithm is based on an artificial neural network, which predicts the change in the motion angle for the following timestep of a moving agent. This network is trained on a dataset of moving crowds and relies on 42 features to achieve its prediction. Overall, this report presents the dataset used for all tests, and describes how the original modulation algorithm's performs against real human behaviors. After this primary analysis, we discuss the need for a new algorithm and explain our methodology used at implementing one. During this study, we design a functional model which combines a classic modulation algorithm and a new artificial neural network one, ready to improve the initial performances when replicating human-like behaviors. Also, performance metrics are described and implemented, giving insights on possible improvements and tests of the new model on a physical robot.

Bibliography

Abu-Dakka, F. J. and M. Saveriano (2020). “Variable impedance control and learning—a review”. In: *Frontiers in Robotics and AI* 7, p. 590681.

Acar, E. U. et al. (2002). “Morse decompositions for coverage tasks”. In: *The international journal of robotics research* 21.4, pp. 331–344.

Aguiar, A. P., J. P. Hespanha, and P. V. Kokotović (2008). “Performance limitations in reference tracking and path following for nonlinear systems”. In: *Automatica* 44.3, pp. 598–610.

Ajeil, F. H. et al. (2020). “Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments”. In: *Sensors* 20.7, p. 1880.

Albu-Schäffer, A., C. Ott, and G. Hirzinger (2007). “A unified passivity-based control framework for position, torque and impedance control of flexible joint robots”. In: *The international journal of robotics research* 26.1, pp. 23–39.

Amato, N. M. and Y. Wu (1996). “A randomized roadmap method for path and manipulation planning”. In: *Proceedings of IEEE international conference on robotics and automation*. Vol. 1. IEEE, pp. 113–120.

Ames, A. D., S. Coogan, et al. (2019). “Control barrier functions: Theory and applications”. In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 3420–3431.

Ames, A. D., J. W. Grizzle, and P. Tabuada (2014). “Control barrier function based quadratic programs with application to adaptive cruise control”. In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 6271–6278.

Ao, S. et al. (2021). “SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11753–11762.

Arslan, O. and D. E. Koditschek (2016a). “Exact robot navigation using power diagrams”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 1–8.

— (2016b). “Voronoi-based coverage control of heterogeneous disk-shaped robots”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 4259–4266.

— (2019). “Sensor-based reactive navigation in unknown convex sphere worlds”. In: *The International Journal of Robotics Research* 38.2-3, pp. 196–223.

Arslan, O., V. Pacelli, and D. E. Koditschek (2017). “Sensory steering for sampling-based motion planning”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3708–3715.

Artstein, Z. (1983). “Stabilization with relaxed controls”. In: *Nonlinear Analysis: Theory, Methods & Applications* 7.11, pp. 1163–1173.

Aurenhammer, F. (1991). “Voronoi diagrams—a survey of a fundamental geometric data structure”. In: *ACM Computing Surveys (CSUR)* 23.3, pp. 345–405.

Badue, C. et al. (2021). “Self-driving cars: A survey”. In: *Expert Systems with Applications* 165, p. 113816.

Bailey, T. and H. Durrant-Whyte (2006). “Simultaneous localization and mapping (SLAM): Part II”. In: *IEEE robotics & automation magazine* 13.3, pp. 108–117.

Bardaro, G. et al. (2018). “MPC-based control architecture of an autonomous wheelchair for indoor environments”. In: *Control Engineering Practice* 78, pp. 160–174.

Bauer, W. et al. (2016). “Lightweight robots in manual assembly–best to start simply”. In: *Frauenhofer-Institut für Arbeitswirtschaft und Organisation IAO, Stuttgart* 1.

Bayındır, L. (2016). “A review of swarm robotics tasks”. In: *Neurocomputing* 172, pp. 292–321.

Best, A., S. Narang, and D. Manocha (2016). “Real-time reciprocal collision avoidance with elliptical agents”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 298–305.

Bhardwaj, M. et al. (2022). “Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation”. In: *Conference on Robot Learning*. PMLR, pp. 750–759.

Bhattacharya, P. and M. L. Gavrilova (2008). “Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path”. In: *IEEE Robotics & Automation Magazine* 15.2, pp. 58–66.

Billard, A., S. Mirrazavi, and N. Figueroa (2022). *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Mit Press.

Birkhoff, G. D. (1927). *Dynamical systems*. Vol. 9. American Mathematical Soc.

Bischoff, R. et al. (2010). “The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing”. In: *ISR 2010 (41st international symposium on robotics) and ROBOTIK 2010 (6th German conference on robotics)*. VDE, pp. 1–8.

Bloom, N. et al. (2020). “Are ideas getting harder to find?” In: *American Economic Review* 110.4, pp. 1104–1144.

Bohlin, R. and L. E. Kavraki (2000). “Path planning using lazy PRM”. In: *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE, pp. 521–528.

Bony, J.-M. (1969). “Principe du maximum, inégalité de Harnack et unicité du probleme de Cauchy pour les opérateurs elliptiques dégénérés”. In: *Annales de l’institut Fourier*. Vol. 19. 1, pp. 277–304.

Borenstein, J. and Y. Koren (1989). “Real-time obstacle avoidance for fast mobile robots”. In: *IEEE Transactions on systems, Man, and Cybernetics* 19.5, pp. 1179–1187.

Borenstein, J., Y. Koren, et al. (1991). “The vector field histogram-fast obstacle avoidance for mobile robots”. In: *IEEE transactions on robotics and automation* 7.3, pp. 278–288.

Box, G. E. (1957). “Evolutionary operation: A method for increasing industrial productivity”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 6.2, pp. 81–101.

Bremermann, H. J. et al. (1962). "Optimization through evolution and recombination". In: *Self-organizing systems* 93, p. 106.

Brezis, H. (1970). "On a characterization of flow-invariant sets". In: *Communications on Pure and Applied Mathematics* 23.2, pp. 261–263.

Brock, O. and O. Khatib (1998). "Elastic strips: Real-time path modification for mobile manipulation". In: *Robotics Research: The Eighth International Symposium*. Springer, pp. 5–13.

— (1999). "High-speed navigation using the global dynamic window approach". In: *Proceedings 1999 ieee international conference on robotics and automation (Cat. No. 99CH36288C)*. Vol. 1. IEEE, pp. 341–346.

— (2002). "Elastic strips: A framework for motion generation in human environments". In: *The International Journal of Robotics Research* 21.12, pp. 1031–1052.

Brooks, J. et al. (2021). "Before coming home: The value of interaction studies with rehabilitation specialists using low-fidelity, physical prototypes prior to inserting novel assistive technologies into seniors' homes". In: *Smart Health* 22, p. 100248.

Brunke, L. et al. (2022). "Safe learning in robotics: From learning-based control to safe reinforcement learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5, pp. 411–444.

Brunn, H. (1913). "Über Kerneigebiete". In: *Mathematische Annalen* 73.3, pp. 436–440.

Bylard, A., R. Bonalli, and M. Pavone (2021). "Composable geometric motion policies using multi-task pullback bundle dynamical systems". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7464–7470.

Cai, M. et al. (2023). "Overcoming Exploration: Deep Reinforcement Learning for Continuous Control in Cluttered Environments from Temporal Logic Specifications". In: *IEEE Robotics and Automation Letters*.

Camerer, C. F. et al. (2018). "Evaluating the replicability of social science experiments in Nature and Science between 2010 and 2015". In: *Nature human behaviour* 2.9, pp. 637–644.

Canny, J. (1988). *The complexity of robot motion planning*. MIT press.

Canny, J. F. and M. C. Lin (1993). "An opportunistic global path planner". In: *Algorithmica* 10.2-4, pp. 102–120.

Chakraborty, A. and A. K. Kar (2017). "Swarm intelligence: A review of algorithms". In: *Nature-inspired computing and optimization*, pp. 475–494.

Chen, X., Y. Michel, and D. Lee (2021). "Closed-Loop Variable Stiffness Control of Dynamical Systems". In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 163–169.

Cheng, C.-A. et al. (2021). "Rmpflow: A geometric framework for generation of multitask motion policies". In: *IEEE Transactions on Automation Science and Engineering* 18.3, pp. 968–987.

Cheng, R. et al. (2019). "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 3387–3395.

Choset, H. and J. Burdick (2000). "Sensor-based exploration: The hierarchical generalized voronoi graph". In: *The International Journal of Robotics Research* 19.2, pp. 96–125.

Chung, S.-Y. and O. Khatib (2015). "Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6289–6294.

Connolly, C. I. (1997). "Harmonic functions and collision probabilities". In: *The International Journal of Robotics Research* 16.4, pp. 497–507.

Connolly, C. I., J. B. Burns, and R. Weiss (1990). "Path planning using Laplace's equation". In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, pp. 2102–2106.

Conzelmann, F. M. et al. (2022). "A Dynamical System Approach to Decentralized Collision-free Autonomous Coordination of a Mobile Assistive Furniture Swarm". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 7259–7265.

Corniani, G. and H. P. Saal (2020). "Tactile innervation densities across the whole body". In: *Journal of Neurophysiology* 124.4, pp. 1229–1240.

CrowdBot (2023). *CrowdBot Challenge*. URL: https://crowdbot.eu/crowdbot-challenge/ (visited on 10/12/2023).

Dahlin, A. and Y. Karayiannidis (2023a). "Creating star worlds: Reshaping the robot workspace for online motion planning". In: *IEEE Transactions on Robotics*.

— (2023b). "Obstacle Avoidance in Dynamic Environments via Tunnel-following MPC with Adaptive Guiding Vector Fields". In: *arXiv preprint arXiv:2303.15869*.

— (2023c). "Robotic Navigation with Convergence Guarantees in Complex Dynamic Environments". In: *arXiv preprint arXiv:2306.12333*.

Daily, R. and D. M. Bevly (2008). "Harmonic potential field path planning for high speed vehicles". In: *American Control Conference, 2008*. IEEE, pp. 4609–4614.

Douce, L.-N. et al. (2023). "Agent Prioritization and Virtual Drag Minimization in Dynamical System Modulation For Obstacle Avoidance of Decentralized Swarms". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Duan, A. et al. (2020). "Learning to Avoid Obstacles With Minimal Intervention Control". In: *Frontiers in Robotics and AI* 7.

Durrant-Whyte, H. and T. Bailey (2006). "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2, pp. 99–110.

Elbanhawi, M. and M. Simic (2014). "Sampling-based robot motion planning: A review". In: *Ieee access* 2, pp. 56–77.

Elfes, A. (1989). "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6, pp. 46–57.

Emika, F. (2023). *Franka Emika*. URL: https://www.franka.de/ (visited on 10/07/2023).

Erdmann, M. and T. Lozano-Perez (1987). "On multiple moving objects". In: *Algorithmica* 2, pp. 477–521.

Falanga, D., K. Kleber, and D. Scaramuzza (2020). "Dynamic obstacle avoidance for quadrotors with event cameras". In: *Science Robotics* 5.40, eaaz9712.

Fallatah, A. et al. (2021). "Towards user-centric robot furniture arrangement". In: *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, pp. 1066–1073.

Feder, H. J. S. and J.-J. Slotine (1997). "Real-time path planning using harmonic potentials in dynamic environments". In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. Vol. 1. IEEE, pp. 874–881.

Ferraguti, F., C. Secchi, and C. Fantuzzi (2013). "A tank-based approach to impedance control with variable stiffness". In: *2013 IEEE international conference on robotics and automation*. IEEE, pp. 4948–4953.

Fetecau, R. C. (2011). "Collective behavior of biological aggregations in two dimensions: a nonlocal kinetic model". In: *Mathematical Models and Methods in Applied Sciences* 21.07, pp. 1539–1569.

Figueroa, N. and A. Billard (2018). "A Physically-Consistent Bayesian Non-Parametric Mixture Model for Dynamical System Learning." In: *CoRL*, pp. 927–946.

Fiorini, P. and Z. Shiller (1998). "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7, pp. 760–772.

Flyability (2023). *Flyability*. URL: https://www.flyability.com/ (visited on 10/07/2023).

Fourment, M. and M. R. Gillings (2008). "A comparison of common programming languages used in bioinformatics". In: *BMC bioinformatics* 9.1, pp. 1–9.

Fox, D., W. Burgard, and S. Thrun (1997). "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1, pp. 23–33.

Fox, D., W. Burgard, S. Thrun, and A. B. Cremers (1998). "A hybrid collision avoidance method for mobile robots". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*. Vol. 2. IEEE, pp. 1238–1243.

Fraichard, T. (1998). "Trajectory planning in a dynamic workspace: a'state-time space'approach". In: *Advanced Robotics* 13.1, pp. 75–94.

Frasch, J. V. et al. (2013). "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles". In: *2013 European Control Conference (ECC)*. IEEE, pp. 4136–4141.

Friedland, B. (2012). *Control system design: an introduction to state-space methods*. Courier Corporation.

Fujiki, T. and K. Tahara (2022). "Series admittance–impedance controller for more robust and stable extension of force control". In: *ROBOMECH Journal* 9.1, p. 23.

Galceran, E. and M. Carreras (2013). "A survey on coverage path planning for robotics". In: *Robotics and Autonomous systems* 61.12, pp. 1258–1276.

Geraerts, R. and M. H. Overmars (2004). "A comparative study of probabilistic roadmap planners". In: *Algorithmic foundations of robotics V*. Springer, pp. 43–57.

Ginesi, M. et al. (2019). "Dynamic movement primitives: Volumetric obstacle avoidance". In: *2019 19th international conference on advanced robotics (ICAR)*. IEEE, pp. 234–239.

Glosser, G. D. and W. S. Newman (1994). "The implementation of a natural admittance controller on an industrial manipulator". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1209–1215.

Goncalves, V. M. et al. (2010). "Vector fields for robot navigation along time-varying curves in n-dimensions". In: *IEEE Transactions on Robotics* 26.4, pp. 647–659.

Gonon, D. J., D. Paez-Granados, and A. Billard (2021). "Reactive Navigation in Crowds for Non-Holonomic Robots With Convex Bounding Shape". In: *IEEE Robotics and Automation Letters* 6.3, pp. 4728–4735.

— (2022). "Robots' Motion Planning in Human Crowds by Acceleration Obstacles". In: *IEEE Robotics and Automation Letters* 7.4, pp. 11236–11243.

Granados, D. F. P., H. Kadone, and K. Suzuki (2018). "Unpowered Lower-Body Exoskeleton with Torso Lifting Mechanism for Supporting Sit-to-Stand Transitions". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2755–2761.

Gribovskaya, E., A. Kheddar, and A. Billard (2011). "Motion learning and adaptive impedance for robot control during physical interaction with humans". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4326–4332.

Grzeskowiak, F. et al. (2020). "Toward virtual reality-based evaluation of robot navigation among people". In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, pp. 766–774.

Guldner, J. and V. I. Utkin (1993). "Sliding mode control for an obstacle avoidance strategy based on an harmonic potential field". In: *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*. IEEE, pp. 424–429.

Haddadin, S. et al. (2010). "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, pp. 3109–3116.

Hansen, G. et al. (2020). "Starshaped sets". In: *Aequationes mathematicae* 94, pp. 1001–1092.

Hart, P. E., N. J. Nilsson, and B. Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Hauser, S. et al. (2020). "Roombots extended: Challenges in the next generation of self-reconfigurable modular robots and their application in adaptive and assistive furniture". In: *Robotics and Autonomous Systems* 127, p. 103467.

Hentout, A., A. Maoudj, and M. Aouache (2023). "A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots". In: *Artificial Intelligence Review* 56.4, pp. 3369–3444.

Hersch, M. et al. (2008). "Dynamical system modulation for robot learning via kinesthetic demonstrations". In: *IEEE Transactions on Robotics* 24.6, pp. 1463–1467.

Hoffmann, H. et al. (2009). "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 2587–2592.

Hogan, N. (1984). "Impedance control: An approach to manipulation". In: *1984 American control conference*. IEEE, pp. 304–313.

— (1985). "Impedance control: An approach to manipulation: Part II—Implementation". In.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Huang, L. (2009). "Velocity planning for a mobile robot to track a moving target—a potential field approach". In: *Robotics and Autonomous Systems* 57.1, pp. 55–63.

Hubbard, P. M. (1996). "Approximating polyhedra with spheres for time-critical collision detection". In: *ACM Transactions on Graphics (TOG)* 15.3, pp. 179–210.

Huber, L., A. Billard, and J.-J. Slotine (2019). "Avoidance of convex and concave obstacles with convergence ensured through contraction". In: *IEEE Robotics and Automation Letters* 4.2, pp. 1462–1469.

Huber, L., J.-J. Slotine, and A. Billard (2022a). "Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds". In: *IEEE Transactions on Robotics*.

— (2022b). "Fast obstacle avoidance based on real-time sensing". In: *IEEE Robotics and Automation Letters*.

— (2023). "Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics". In: *IEEE Transactions on Robotics*.

Huber, L., T. Trinca, et al. (2023). "Passive Obstacle Aware Control to Follow Desired Velocities". In: *IEEE Robotics and Automation Letters (submitted)*.

Ijspeert, A. J., J. Nakanishi, H. Hoffmann, et al. (2013). "Dynamical movement primitives: learning attractor models for motor behaviors". In: *Neural computation* 25.2, pp. 328–373.

Ijspeert, A. J., J. Nakanishi, and S. Schaal (2002). "Movement imitation with nonlinear dynamical systems in humanoid robots". In: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. Vol. 2. IEEE, pp. 1398–1403.

Jaillet, L. and T. Siméon (2004). "A PRM-based motion planner for dynamically changing environments". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 2. IEEE, pp. 1606–1611.

Ji, J. et al. (2016). "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints". In: *IEEE Transactions on Vehicular Technology* 66.2, pp. 952–964.

— (2017). "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints". In: *IEEE Transactions on Vehicular Technology* 66.2, pp. 952–964.

Jia, D., A. Hermans, and B. Leibe (2020). "DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10270–10277. DOI: 10.1109/IROS45743.2020.9341689.

Kahneman, D. (2017). *Thinking, fast and slow*.

Kalakrishnan, M. et al. (2011). "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation*. IEEE, pp. 4569–4574.

Kamon, I., E. Rimon, and E. Rivlin (1999). "Range-sensor based navigation in three dimensions". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*. Vol. 1. IEEE, pp. 163–169.

Kamon, I. and E. Rivlin (1997). "Sensory-based motion planning with global proofs". In: *IEEE transactions on Robotics and Automation* 13.6, pp. 814–822.

Kant, K. and S. W. Zucker (1986). "Toward efficient trajectory planning: The path-velocity decomposition". In: *The international journal of robotics research* 5.3, pp. 72–89.

Kapitanyuk, Y. A., S. A. Chepinskiy, and A. A. Kapitonov (2014). "Geometric path following control of a rigid body based on the stabilization of sets". In: *IFAC Proceedings Volumes* 47.3, pp. 7342–7347.

Kapitanyuk, Y. A., A. V. Proskurnikov, and M. Cao (2017). "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots". In: *IEEE Transactions on Control Systems Technology* 26.4, pp. 1372–1385.

Karaman, S. and E. Frazzoli (2011). "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7, pp. 846–894.

Karaman, S., M. R. Walter, et al. (2011). "Anytime motion planning using the RRT". In: *2011 IEEE international conference on robotics and automation*. IEEE, pp. 1478–1483.

Kass, M., A. Witkin, and D. Terzopoulos (1988). "Snakes: Active contour models". In: *International journal of computer vision* 1.4, pp. 321–331.

Kavraki, L. and J.-C. Latombe (1994). "Randomized preprocessing of configuration for fast path planning". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2138–2145.

Kavraki, L. E. et al. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580.

Khansari-Zadeh, S. M. (2012). *A dynamical system-based approach to modeling stable robot control policies via imitation learning*. Tech. rep. Epfl.

Khansari-Zadeh, S. M. and A. Billard (2011). "Learning stable nonlinear dynamical systems with gaussian mixture models". In: *IEEE Transactions on Robotics* 27.5, pp. 943–957.

Khansari-Zadeh, S. (2012). *A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning*. PhD Thesis.

Khansari-Zadeh, S. M. and A. Billard (2012). "A dynamical system approach to realtime obstacle avoidance". In: *Autonomous Robots* 32.4, pp. 433–454.

Khatib, O. (1986). "Real-time obstacle avoidance for manipulators and mobile robots". In: *The international journal of robotics research* 5.1, pp. 90–98.

Khoramshahi, M. and A. Billard (2019). "A dynamical system approach to task-adaptation in physical human–robot interaction". In: *Autonomous Robots* 43.4, pp. 927–946.

Khosla, P. and R. Volpe (1988). "Superquadric artificial potentials for obstacle avoidance and approach". In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1778–1784.

Kiai, A. (2019). "To protect credibility in science, banish "publish or perish"". In: *Nature human behaviour* 3.10, pp. 1017–1018.

Kim, J.-O. and P. K. Khosla (1992). "Real-time obstacle avoidance using harmonic potential functions". In: *IEEE Transactions on Robotics and Automation* 8.3, pp. 338–349.

Kim, S., A. Shukla, and A. Billard (2014). "Catching objects in flight". In: *IEEE Transactions on Robotics* 30.5, pp. 1049–1065.

Kingston, Z., M. Moll, and L. E. Kavraki (2018). "Sampling-based methods for motion planning with constraints". In: *Annual review of control, robotics, and autonomous systems* 1, pp. 159–185.

Kishi, Y. et al. (2003). "Passive impedance control with time-varying impedance center". In: *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*. Vol. 3. IEEE, pp. 1207–1212.

Knuth, D. E. (1997). *The art of computer programming*. Vol. 3. Pearson Education.

Koditschek, D. (1987). "Exact robot navigation by means of potential functions: Some topological considerations". In: *Proceedings. 1987 IEEE international conference on robotics and automation*. Vol. 4. IEEE, pp. 1–6.

Koditschek, D. E. and E. Rimon (1990). "Robot navigation functions on manifolds with boundary". In: *Advances in applied mathematics* 11.4, pp. 412–442.

Koenemann, J. et al. (2015). "Whole-body model-predictive control applied to the HRP-2 humanoid". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3346–3351.

Koptev, M., N. Figueroa, and A. Billard (2021). "Real-Time Self-Collision Avoidance in Joint Space for Humanoid Robots". In: *IEEE Robotics and Automation Letters* 6.2, pp. 1240–1247.

— (2022). "Neural Joint Space Implicit Signed Distance Functions for Reactive Robot Manipulator Control". In: *IEEE Robotics and Automation Letters* 8.2, pp. 480–487.

Koren, Y. and J. Borenstein (1991). "Potential field methods and their inherent limitations for mobile robot navigation". In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, pp. 1398–1399.

Kroemer, O., S. Niekum, and G. Konidaris (2021). "A review of robot learning for manipulation: Challenges, representations, and algorithms". In: *The Journal of Machine Learning Research* 22.1, pp. 1395–1476.

Kronander, K. and A. Billard (2015). "Passive interaction control with dynamical systems". In: *IEEE Robotics and Automation Letters* 1.1, pp. 106–113.

— (2016). "Stability considerations for variable impedance control". In: *IEEE Transactions on Robotics* 32.5, pp. 1298–1305.

Kronander, K., M. Khansari, and A. Billard (2015). "Incremental motion learning with locally modulated dynamical systems". In: *Robotics and Autonomous Systems* 70, pp. 52–62.

Kuffner, J. J. and S. M. LaValle (2000). "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE, pp. 995–1001.

Kufoalor, D. K. M., E. F. Brekke, and T. A. Johansen (2018). "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2402–2409.

Kumar, H., S. Paternain, and A. Ribeiro (2019). "Navigation of a quadratic potential with ellipsoidal obstacles". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, pp. 4777–4784.

Kurfess, T. R. et al. (2005). *Robotics and automation handbook*. Vol. 414. CRC press Boca Raton, FL.

Kyaw, P. T. et al. (2020). "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem". In: *IEEE Access* 8, pp. 225945–225956.

Lacevic, B. and P. Rocco (2010). "Kinetostatic danger field-a novel safety assessment for human-robot interaction". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2169–2174.

Lacevic, B., P. Rocco, and A. M. Zanchettin (2013). "Safety assessment and control of robotic manipulators using danger field". In: *IEEE Transactions on Robotics* 29.5, pp. 1257–1270.

Lacevic, B., A. M. Zanchettin, and P. Rocco (2022). "Safe Human-robot collaboration via collision checking and explicit representation of danger zones". In: *IEEE Transactions on Automation Science and Engineering* 20.2, pp. 846–861.

Landau, I. D. et al. (2011). *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media.

Lau, D., J. Eden, and D. Oetomo (2015). "Fluid motion planner for nonholonomic 3-D mobile robots with kinematic constraints". In: *IEEE Transactions on Robotics* 31.6, pp. 1537–1547.

LaValle, S. M. et al. (1998). "Rapidly-exploring random trees: A new tool for path planning". In.

LaValle, S. M., M. S. Branicky, and S. R. Lindemann (2004). "On the relationship between classical grid search and probabilistic roadmaps". In: *The International Journal of Robotics Research* 23.7-8, pp. 673–692.

LaValle, S. M. and J. J. Kuffner Jr (2000). "Rapidly-exploring random trees: Progress and prospects". In.

— (2001). "Randomized kinodynamic planning". In: *The international journal of robotics research* 20.5, pp. 378–400.

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Lee, C. et al. (2017). "Soft robot review". In: *International Journal of Control, Automation and Systems* 15, pp. 3–15.

Lee, D.-T. and F. P. Preparata (1979). "An optimal algorithm for finding the kernel of a polygon". In: *Journal of the ACM (JACM)* 26.3, pp. 415–421.

Lee, D. and K. Huang (2010). "Passive-set-position-modulation framework for interactive robotic systems". In: *IEEE Transactions on Robotics* 26.2, pp. 354–369.

Levine, S. P. et al. (1999). "The NavChair assistive wheelchair navigation system". In: *IEEE transactions on rehabilitation engineering* 7.4, pp. 443–451.

Li, P. Y. and R. Horowitz (1999). "Passive velocity field control of mechanical manipulators". In: *IEEE Transactions on robotics and automation* 15.4, pp. 751–763.

Li, Q., W. Chen, and J. Wang (2011). "Dynamic shared control for human-wheelchair cooperation". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4278–4283.

Li, Z. and T. Bui (1998). "Robot path planning using fluid model". In: *Journal of Intelligent and Robotic Systems* 21, pp. 29–50.

Liddy, T. et al. (2008). "Obstacle avoidance using complex vector fields". In: *Proc. of the 2008 Australasian Conference on Robotics and Automation, Canberra, Australia*.

Lindemann, S. R. and S. M. LaValle (2009). "Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions". In: *The International Journal of Robotics Research* 28.5, pp. 600–621.

Lohmiller, W. and J.-J. E. Slotine (1998). "On contraction analysis for non-linear systems". In: *Automatica* 34.6, pp. 683–696.

Loizou, S. and E. Rimon (2022). "Mobile Robot Navigation Functions Tuned by Sensor Readings in Partially Known Environments". In: *IEEE Robotics and Automation Letters*.

Loizou, S. G. (2011). "Closed form navigation functions based on harmonic potentials". In: *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, pp. 6361–6366.

— (2017). "The navigation transformation". In: *IEEE Transactions on Robotics* 33.6, pp. 1516–1523.

López-González, A. et al. (2020). "Multi robot distance based formation using Parallel Genetic Algorithm". In: *Applied Soft Computing* 86, p. 105929.

Lumelsky, V. and A. Stepanov (1986). "Dynamic path planning for a mobile automaton with limited information on the environment". In: *IEEE transactions on Automatic control* 31.11, pp. 1058–1063.

Lynch, K. M. and F. C. Park (2017). *Modern robotics*. Cambridge University Press.

Ma, Y., D. Manocha, and W. Wang (2018). "Efficient reciprocal collision avoidance between heterogeneous agents using ctmat". In: *arXiv preprint arXiv:1804.02512*.

Mainprice, J. et al. (2020). "An interior point method solving motion planning problems with narrow passages". In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, pp. 547–552.

Marchidan, A. and E. Bakolas (2022). "A local reactive steering law for 2D collision avoidance with curvature constraints and constant speed". In: *Robotics and Autonomous Systems* 155, p. 104156.

Matsubara, T., S.-H. Hyon, and J. Morimoto (2010). "Learning stylistic dynamic movement primitives from multiple demonstrations". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1277–1283.

Matthias, B., H. Ding, and V. Miegel (2013). "Die Zukunft der Mensch-Roboter Kollaboration in der industriellen Montage". In: *Internationales Forum Mechatronik*. Vol. 12. 121, p. 2013.

Mayne, D. Q. et al. (2000). "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6, pp. 789–814.

McGuire, K. N., G. C. de Croon, and K. Tuyls (2019). "A comparative study of bug algorithms for robot navigation". In: *Robotics and Autonomous Systems* 121, p. 103261.

Michel, Y., M. Saveriano, and D. Lee (2023). "A passivity-based approach for variable stiffness control with dynamical systems". In: *arXiv preprint arXiv:2307.09571*.

Michels, J., A. Saxena, and A. Y. Ng (2005). "High speed obstacle avoidance using monocular vision and reinforcement learning". In: *Proceedings of the 22nd international conference on Machine learning*. ACM, pp. 593–600.

Milne-Thomson, L. M. (1996). *Theoretical hydrodynamics*. Courier Corporation.

Missura, M. and M. Bennewitz (2019). "Predictive collision avoidance for the dynamic window approach". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8620–8626.

Mogilner, A. and L. Edelstein-Keshet (1999). "A non-local model for a swarm". In: *Journal of mathematical biology* 38.6, pp. 534–570.

Muller, U. et al. (2005). "Off-road obstacle avoidance through end-to-end learning". In: *Advances in neural information processing systems* 18.

"Multiresolution path planning for mobile robots" (1986). In: *IEEE Journal on Robotics and Automation* 2.3, pp. 135–145.

Murray, S. et al. (2016). "Robot Motion Planning on a Chip." In: *Robotics: Science and Systems*.

Nagumo, M. (1942). "Über die lage der integralkurven gewöhnlicher differentialgleichungen". In: *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series* 24, pp. 551–559.

Nan, F. et al. (2022). "Nonlinear MPC for quadrotor fault-tolerant control". In: *IEEE Robotics and Automation Letters* 7.2, pp. 5047–5054.

Narayanan, D. et al. (2021). "Efficient large-scale language model training on gpu clusters using megatron-lm". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15.

Nasrinahar, A. and J. H. Chuah (2018). "Intelligent motion planning of a mobile robot with dynamic obstacle avoidance". In: *Journal on Vehicle Routing Algorithms* 1, pp. 89–104.

Noreen, I., A. Khan, and Z. Habib (2016). "Optimal path planning using RRT* based approaches: a survey and future directions". In: *International Journal of Advanced Computer Science and Applications* 7.11.

Notomista, G. and M. Saveriano (2021). "Safety of Dynamical Systems with Multiple Non-Convex Unsafe Sets Using Control Barrier Functions". In: *IEEE Control Systems Letters*.

Ó'Dúnlaing, C. and C. K. Yap (1985). "A "retraction" method for planning the motion of a disc". In: *Journal of Algorithms* 6.1, pp. 104–111.

Ogren, P. and N. E. Leonard (2005). "A convergent dynamic window approach to obstacle avoidance". In: *IEEE Transactions on Robotics* 21.2, pp. 188–195.

Paez-Granados, D. et al. (2022). "Personal Mobility With Synchronous Trunk–Knee Passive Exoskeleton: Optimizing Human–Robot Energy Transfer". In: *IEEE/ASME Transactions on Mechatronics*. DOI: 10.1109/TMECH.2021.3135453.

Panagou, D. (2014). "Motion planning and collision avoidance using navigation vector fields". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2513–2518.

Park, C., J. Pan, and D. Manocha (2012). "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments". In: *Proceedings of the international conference on automated planning and scheduling*. Vol. 22, pp. 207–215.

Park, D.-H. et al. (2008). "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields". In: *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 91–98.

Park, M., E. Leahey, and R. J. Funk (2023). "Papers and patents are becoming less disruptive over time". In: *Nature* 613.7942, pp. 138–144.

Paternain, S., D. E. Koditschek, and A. Ribeiro (2017). "Navigation functions for convex potentials in a space with convex obstacles". In: *IEEE Transactions on Automatic Control*.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Perrin, N. and P. Schlehuber-Caissier (2016). "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems". In: *Systems & Control Letters* 96, pp. 51–59.

Peters, J. et al. (2008). "A unifying framework for robot control with redundant DOFs". In: *Autonomous Robots* 24, pp. 1–12.

Pierson, A. and D. Rus (2017). "Distributed target tracking in cluttered environments with guaranteed collision avoidance". In: *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, pp. 83–89.

Polverini, M. P., A. M. Zanchettin, and P. Rocco (2014). "Real-time collision avoidance in human-robot interaction based on kinetostatic safety field". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4136–4141.

Prajna, S. and A. Jadbabaie (2004). "Safety verification of hybrid systems using barrier certificates". In: *International Workshop on Hybrid Systems: Computation and Control*. Springer, pp. 477–492.

Prassler, E., J. Scholz, and P. Fiorini (2001). "A robotics wheelchair for crowded public environment". In: *IEEE Robotics & Automation Magazine* 8.1, pp. 38–45.

Quinlan, S. and O. Khatib (1993). "Elastic bands: Connecting path planning and control". In: *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, pp. 802–807.

Rai, A. et al. (2014). "Learning coupling terms for obstacle avoidance". In: *2014 IEEE-RAS international conference on humanoid robots*. IEEE, pp. 512–518.

Rao, N. S., N. Stoltzfus, and S. S. Iyengar (1991). "A'retraction'method for learned navigation in unknown terrains for a circular robot". In: *IEEE Transactions on Robotics and Automation* 7.5, pp. 699–707.

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

Ratliff, N., M. Zucker, et al. (2009). "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE international conference on robotics and automation*. IEEE, pp. 489–494.

Ravankar, A. A. et al. (2020). "HPPRM: hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots". In: *IEEE Access* 8, pp. 221743–221766.

Ravichandar, H. et al. (2020). "Recent advances in robot learning from demonstration". In: *Annual review of control, robotics, and autonomous systems* 3, pp. 297–330.

"Real-time obstacle avoidance using harmonic potential functions" (2011). In: *Journal of Automation Mobile Robotics and Intelligent Systems* 5.3, pp. 59–66.

Redmon, J. et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Reis, M. F., A. P. Aguiar, and P. Tabuada (2020). "Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria". In: *IEEE Control Systems Letters* 5.2, pp. 731–736.

Rimon, E. and D. Koditschek (1992). "Exact robot navigation using artificial potential functions". In: *IEEE Transactions on Robotics and Automation* 8.5, pp. 501–518.

Rimon, E. and D. E. Koditschek (1991). "The construction of analytic diffeomorphisms for exact robot navigation on star worlds". In: *Transactions of the American Mathematical Society* 327.1, pp. 71–116.

Riviere, B. et al. (2020). "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning". In: *IEEE robotics and automation letters* 5.3, pp. 4249–4256.

Rodriguez, S. et al. (2006). "An obstacle-based rapidly-exploring random tree". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 895–900.

Romdlony, M. Z. and B. Jayawardhana (2014). "Uniting control Lyapunov and control barrier functions". In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 2293–2298.

Saffiotti, A. (1997). "The uses of fuzzy logic in autonomous robot navigation". In: *Soft computing* 1.4, pp. 180–197.

Salehian, S. S. M. and A. Billard (2018). "A dynamical-system-based approach for controlling robotic manipulators during noncontact/contact transitions". In: *IEEE Robotics and Automation Letters* 3.4, pp. 2738–2745.

Sánchez, G. and J.-C. Latombe (2003). "A single-query bi-directional probabilistic roadmap planner with lazy collision checking". In: *Robotics research*. Springer, pp. 403–417.

Sánchez, I. et al. (2021). "Nonlinear model predictive path following controller with obstacle avoidance". In: *Journal of Intelligent & Robotic Systems* 102, pp. 1–18.

Sankaranarayanan, A. and M. Vidyasagar (1990). "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane". In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, pp. 1930–1936.

Saveriano, M., F. J. Abu-Dakka, et al. (2021). "Dynamic movement primitives in robotics: A tutorial survey". In: *arXiv preprint arXiv:2102.03861*.

Saveriano, M., F. Hirt, and D. Lee (2017). "Human-aware motion reshaping using dynamical systems". In: *Pattern Recognition Letters* 99, pp. 96–104.

Saveriano, M. and D. Lee (2013). "Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 5380–5387.

— (2014). "Distance based dynamical system modulation for reactive avoidance of moving obstacles". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 5618–5623.

— (2019). “Learning barrier functions for constrained motion planning with dynamical systems”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 112–119.

Schulman, J., Y. Duan, et al. (2014). “Motion planning with sequential convex optimization and convex collision checking”. In: *The International Journal of Robotics Research* 33.9, pp. 1251–1270.

Schulman, J., J. Ho, et al. (2013). “Finding locally optimal, collision-free trajectories with sequential convex optimization.” In: *Robotics: science and systems*. Vol. 9. 1. Berlin, Germany, pp. 1–10.

Schwager, M., D. Rus, and J.-J. Slotine (2011). “Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment”. In: *The International Journal of Robotics Research* 30.3, pp. 371–383.

Schwarting, W., J. Alonso-Mora, and D. Rus (2018). “Planning and decision-making for autonomous vehicles”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1, pp. 187–210.

Seidel, R. (1991). “A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons”. In: *Computational Geometry* 1.1, pp. 51–64.

Sepulchre, R., M. Jankovic, and P. V. Kokotovic (2012). *Constructive nonlinear control*. Springer Science & Business Media.

Shahriari, E. et al. (2017). “Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, pp. 136–142.

Shi, K., J. Denny, and N. M. Amato (2014). “Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4659–4666.

Siciliano, B. and O. Khatib (2008). *Springer handbook of robotics*. Vol. 200. Springer.

Slotine, J.-J. E. et al. (n.d.). *Applied nonlinear control*. Vol. 199. 1.

Smil, V. (2018). *Energy and civilization: a history*. MIT press.

Standard, I. (2016). “ISO/TS 15066: 2016: Robots and Robotic Devices–Collaborative Robots”. In: *International Organization for Standardization: Geneva, Switzerland*.

Stoddard, B., M.-R. Giolando, and H. Knight (2021). “Teleoperating Multi-robot Furniture”. In: *International Conference on Social Robotics*, pp. 521–531.

Stramigioli, S. et al. (2005). “Sampled data systems passivity and discrete port-hamiltonian systems”. In: *IEEE Transactions on Robotics* 21.4, pp. 574–587.

Sucan, I. A., M. Moll, and L. E. Kavraki (2012). “The open motion planning library”. In: *IEEE Robotics & Automation Magazine* 19.4, pp. 72–82.

Sultana, F., A. Sufian, and P. Dutta (2020). “A review of object detection models based on convolutional neural network”. In: *Intelligent computing: image processing based applications*, pp. 1–16.

Takegaki, M. and S. Arimoto (1981). “A new feedback method for dynamic control of manipulators”. In.

Taylor, A. et al. (2020). “Learning for safety-critical control with control barrier functions”. In: *Learning for Dynamics and Control*. PMLR, pp. 708–717.

Teli, T. A. and M. A. Wani (2021). “A fuzzy based local minima avoidance path planning in autonomous robots”. In: *International Journal of Information Technology* 13, pp. 33–40.

Tsounis, V. et al. (2020). “Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning”. In: *IEEE Robotics and Automation Letters* 5.2, pp. 3699–3706.

Tsourveloudis, N. C., K. P. Valavanis, and T. Hebert (2001). “Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic”. In: *IEEE transactions on robotics and automation* 17.4, pp. 490–497.

Tsymbal, A. (2004). “The problem of concept drift: definitions and related work”. In: *Computer Science Department, Trinity College Dublin* 106.2, p. 58.

Tulbure, A. and O. Khatib (2020). “Closing the loop: Real-time perception and control for robust collision avoidance with occluded obstacles”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5700–5707.

Tuncer, A. and M. Yildirim (2012). “Dynamic path planning of mobile robots with improved genetic algorithm”. In: *Computers & Electrical Engineering* 38.6, pp. 1564–1572.

Ude, A. et al. (2010). “Task-specific generalization of discrete and periodic dynamic movement primitives”. In: *IEEE Transactions on Robotics* 26.5, pp. 800–815.

Udwadia, F. (2003). “A new perspective on the tracking control of nonlinear structural and mechanical systems”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 459.2035, pp. 1783–1800.

Ulrich, I. and J. Borenstein (1998). “VFH+: Reliable obstacle avoidance for fast mobile robots”. In: *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*. Vol. 2. IEEE, pp. 1572–1577.

Van Den Berg, J., D. Ferguson, and J. Kuffner (2006). “Anytime path planning and replanning in dynamic environments”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 2366–2371.

Van Den Berg, J., S. J. Guy, et al. (2011). “Reciprocal n-body collision avoidance”. In: *Robotics Research: The 14th International Symposium ISRR*. Springer, pp. 3–19.

Van den Berg, J., M. Lin, and D. Manocha (2008). “Reciprocal velocity obstacles for real-time multi-agent navigation”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1928–1935.

Van Den Berg, J., J. Snape, et al. (2011). “Reciprocal collision avoidance with acceleration-velocity obstacles”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3475–3482.

Van Den Berg, J. P. and M. H. Overmars (2005). “Roadmap-based motion planning in dynamic environments”. In: *IEEE transactions on robotics* 21.5, pp. 885–897.

Van Wyk, K. et al. (2022). “Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior”. In: *IEEE Robotics and Automation Letters* 7.2, pp. 3202–3209.

Vanderborght, B. et al. (2013). “Variable impedance actuators: A review”. In: *Robotics and autonomous systems* 61.12, pp. 1601–1614.

Vannoy, J. and J. Xiao (2008). "Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes". In: *IEEE Transactions on Robotics* 24.5, pp. 1199–1212.

Vonásek, V. et al. (2009). "RRT-path–a guided rapidly exploring random tree". In: *Robot motion and control 2009*. Springer, pp. 307–316.

Voulodimos, A. et al. (2018). "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018.

Wang, C. et al. (2013). "A real-time obstacle avoidance strategy for safe autonomous navigation of intelligent hospital beds in dynamic uncertain environments". In: *Proceedings of Australasian Conference on Robotics and Automation*.

Warren, C. W. (1989). "Global path planning using artificial potential fields". In: *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, pp. 316–317.

Waydo, S. and R. M. Murray (2003). "Vehicle motion planning using stream functions". In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 2. IEEE, pp. 2484–2491.

Wieland, P. and F. Allgöwer (2007). "Constructive safety using control barrier functions". In: *IFAC Proceedings Volumes* 40.12, pp. 462–467.

Wilhelm, J. P. and G. Clem (2019). "Vector field UAV guidance for path following and obstacle avoidance with minimal deviation". In: *Journal of Guidance, Control, and Dynamics* 42.8, pp. 1848–1856.

Wilkie, D., J. Van Den Berg, and D. Manocha (2009). "Generalized velocity obstacles". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5573–5578.

Willems, J. C. (1972). "Dissipative dynamical systems part I: General theory". In: *Archive for rational mechanics and analysis* 45.5, pp. 321–351.

Williams, G., A. Aldrich, and E. A. Theodorou (2017). "Model predictive path integral control: From theory to parallel computation". In: *Journal of Guidance, Control, and Dynamics* 40.2, pp. 344–357.

Xiao, J. et al. (1997). "Adaptive evolutionary planner/navigator for mobile robots". In: *IEEE transactions on evolutionary computation* 1.1, pp. 18–28.

Xing, J. et al. (2023). "Autonomous Power Line Inspection with Drones via Perception-Aware MPC". In: *arXiv preprint arXiv:2304.00959*.

Yang, S. X., Y. Hu, and M. Q.-h. Meng (2006). "A knowledge based GA for path planning of multiple mobile robots in dynamic environments". In: *2006 IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, pp. 1–6.

Yang, Y. and O. Brock (2010). "Elastic roadmaps—motion generation for autonomous mobile manipulation". In: *Autonomous Robots* 28, pp. 113–130.

Yao, W., B. Lin, et al. (2022). "Guiding vector fields for following occluded paths". In: *IEEE Transactions on Automatic Control*.

Yao, W., H. G. de Marina, et al. (2021). "Singularity-free guiding vector field for robot navigation". In: *IEEE Transactions on Robotics* 37.4, pp. 1206–1221.

Zadeh, L. A. (1965). "Fuzzy sets". In: *Information and control* 8.3, pp. 338–353.

Zadeh, L. A. (1996). “Fuzzy sets”. In: *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*. World Scientific, pp. 394–432.

Zanchettin, A. M., B. Lacevic, and P. Rocco (2015). “Passivity-based control of robotic manipulators for safe cooperation with humans”. In: *International Journal of Control* 88.2, pp. 429–439.

Zhang, X., A. Liniger, and F. Borrelli (2020). “Optimization-based collision avoidance”. In: *IEEE Transactions on Control Systems Technology* 29.3, pp. 972–983.

Zhang, Y., N. Fattahi, and W. Li (2013). “Probabilistic roadmap with self-learning for path planning of a mobile robot in a dynamic and unstructured environment”. In: *2013 IEEE International Conference on Mechatronics and Automation*. IEEE, pp. 1074–1079.

Zheng, D. et al. (2020). “A Dynamical System Approach to Real-time Three-Dimensional Concave Obstacle Avoidance”. In: *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, pp. 1082–1087.

Zhou, D. et al. (2017). “Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells”. In: *IEEE Robotics and Automation Letters* 2.2, pp. 1047–1054.

Zhou, X. et al. (2022). “Swarm of micro flying robots in the wild”. In: *Science Robotics* 7.66, eabm5954.

Zucker, M. et al. (2013). “Chomp: Covariant hamiltonian optimization for motion planning”. In: *The International Journal of Robotics Research* 32.9-10, pp. 1164–1193.

LUKAS HUBER

Robotics Researcher, Software Engineer & Machine Learning Expert

lukas.huber@outlook.com | +41 77 417 33 47 | Antonsgasse 4, 6312 Steinhausen, Switzerland
May 26, 1992 | Swiss | lukas-huber-engineer | hubernikus

EXPERIENCE

Researcher / Ph.D.

EPFL

2018, September – Ongoing | Lausanne, Switzerland

- Research in motion planning, nonlinear control, machine learning
- Teaching Assistant of: Advanced Machine Learning, Learning and Adaptive Robot Control
- Scientific publication in peer-reviewed journals

Co-founder / CEO

AICA

2019, July – 2021, June | Lausanne, Switzerland

- Creation of start-up enhancing artificial intelligence for robotics
- Market research & business development
- Early stage fund-raising (totaling USD 350k)

Project Manager / Civilist

CEAS

2015, May – 2016, August | Antananarivo, Madagascar

- Design of a small wind turbine (1.5 kW) for decentralized electrification
- Supervising construction of 3 wind turbines
- Lead engineer in the workshop of 20 people

EDUCATION

Ph.D. in Robotics

EPFL

2018 – July, 2023 (expected)

Exact Obstacle Avoidance in Complex and Dynamic Environments Using Local Modulation of Dynamics

M.Sc. in Mechanical Engineering

MIT / EPFL

2016 - 2018

Focus: Control Theory, Renewable Energies
Master Thesis: *Avoidance of Convex and Concave Obstacles with Convergence ensured through Contraction*

B.Sc. in Mechanical Engineering

ETH Zurich

2011 - 2015

MY LIFE PHILOSOPHY

"Question everything, but do not reinvent the wheel."

MOST PROUD OF

Starting a Company
from zero to five employees

Development Project in Africa
building wind mills with few resources

Impactful Research
by combining analytical and technical skills

STRENGTHS

Analytical | Endurant | Critical | Motivator | Proactive | Trustworthy

SKILLS / SOFTWARE

Expert

C | Python | Octave / MATLAB | ROS2 | Git | LaTex | Office Suite

Advanced

C++ | Rust | Adobe | Gazebo | shell (Linux / Windows) | Simulink

Moderate

HTML, CSS, JS | Vue | NX / CAD | Lisp | Blender | Adobe Suite

LANGUAGES

German	●●●●●
English	●●●●●
French	●●●●○

PUBLICATIONS

Journal Articles

- L. Huber, J.-J. Slotine, and A. Billard, "Avoidance of concave obstacles through rotation of nonlinear dynamics," *IEEE Transactions on Robotics*, 2023.
- L. Huber, T. Thibaud, J.-J. Slotine, and A. Billard, "Passive obstacle aware control to follow desired velocities," *IEEE Robotics and Automation Letters (submitted)*, 2023.
- 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.
- L. Huber, J.-J. Slotine, and A. Billard, "Fast obstacle avoidance based on real-time sensing," *IEEE Robotics and Automation Letters*, 2022.
- L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, 2019.

Conference Proceedings

- L.-N. Douce, A. Menichelli, L. Huber, *et al.*, "Agent prioritization and virtual drag minimization in dynamical system modulation for obstacle avoidance of decentralized swarms," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023.
- F. M. Conzelmann, L. Huber, D. Paez-Granados, A. Bolotnikova, A. Ijspeert, and A. Billard, "A dynamical system approach to decentralized collision-free autonomous coordination of a mobile assistive furniture swarm," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 7259–7265.

Abstracts

- L. Huber, J.-J. Slotine, and A. Billard, *From obstacle avoidance to motion learning using local rotation of dynamical systems*, Workshop on Machine Learning and Automated Reasoning for Intelligent Robots and Systems, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.
- L. Huber, J.-J. Slotine, and A. Billard, *A unified approach to obstacle avoidance and motion planning*, Workshop on Deployable Decision Making in Embodied Systems (DDM), Conference on Neural Information Processing Systems (NeurIPS), 2021.