THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Reactive Motion Planning and Control under Constraints

A Dynamical Systems Approach

Albin Dahlin

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2023

Reactive Motion Planning and Control under Constraints
A Dynamical Systems Approach

Albin Dahlin
ISBN 978-91-7905-972-9

© 2023 Albin Dahlin
All rights reserved.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5438
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Printed by Chalmers Digitaltryckeri
Gothenburg, Sweden, December 2023

Till Tove och Felix.

Abstract

Modern robots are increasingly being designed to operate in dynamic and unstructured environments shared with humans and other mobile agents. In these scenarios, the robot must react to any online detected changes to provide a correct and safe operation. This thesis introduces techniques for motion planning and control employing Dynamical Systems (DSs) formulations, such as Artificial Potential Fields (APF) and Dynamical Movement Primitives (DMP). While several DS-based methods facilitate reactivity to changes in the robot environment with guaranteed convergence to a specified goal, they often lack the integration of robot constraints. Moreover, convergence depends on specific environmental conditions, e.g., the environment being a *star world*.

To extend the range of practical scenarios in which convergence to the goal is guaranteed in star worlds, a method is proposed for online adjustment of the robot's perceived environment to align with the necessary conditions. This method is enabled through a comprehensive exploration of the concept star-shaped hull. To account for robot constraints, two strategies are proposed. The first approach employs online modification of the DS through a scaling factor to meet velocity and acceleration constraints. Compared to standard scaling approaches, focus is placed on mitigating the feasibility issues that are more prominent in reactive motion planning. In particular, by proactively scaling the DS before reaching the acceleration limits, the proposed method retains the feasibility for a wider set of trajectories. The efficacy of the scaling is showcased by experiments on a robotic arm. The second approach integrates a DS method within a Model Predictive Control (MPC) scheme. This integration effectively combines the favorable convergence characteristics of the DS with the intuitive representation of system constraints offered by MPC. In this configuration, the DS generates a path with an obstacle clearance, while the MPC provides a feasible trajectory that adheres to this clearance distance. The scheme is proven to accomplish collision avoidance and to preserve the convergence guarantees provided by the DS for a substantial portion of the workspace. Furthermore, the scheme is extended to encompass path-following control as well.

Keywords: Robotics, dynamical systems, optimization-based control, computational geometry, online trajectory generation, navigation.

List of Publications

This thesis is based on the following publications:

[A] **Albin Dahlin**, Yiannis Karayiannidis, "Adaptive Trajectory Generation under Velocity Constraints using Dynamical Movement Primitives". Published in IEEE Control Systems Letters, Apr. 2020.

[B] **Albin Dahlin**, Yiannis Karayiannidis, "Temporal Coupling of Dynamical Movement Primitives for Constrained Velocities and Accelerations". Published in IEEE Robotics and Automation Letters, Apr. 2021.

[C] **Albin Dahlin**, Yiannis Karayiannidis, "Trajectory Scaling for Reactive Motion Planning". Published in 2022 International Conference on Robotics and Automation (ICRA).

[D] **Albin Dahlin**, Yiannis Karayiannidis, "Creating Star Worlds: Reshaping the Robot Workspace for Online Motion Planning". Published in IEEE Transactions on Robotics, Oct. 2023.

[E] **Albin Dahlin**, Yiannis Karayiannidis, "Obstacle Avoidance in Dynamic Environments via Tunnel-following MPC with Adaptive Guiding Vector Fields". Accepted for publication in 62nd IEEE Conference on Decision and Control (CDC 2023).

[F] **Albin Dahlin**, Yiannis Karayiannidis, "Autonomous Navigation with Convergence Guarantees in Complex Dynamic Environments". To be submitted [journal article].

Acknowledgments

First and foremost, I want to thank my supervisor Yiannis Karayiannidis. Thank you for the insightful discussions, encouraging support, and invaluable guidance. I am truly grateful for the opportunity to pursue a doctoral degree under your mentorship. I also want to express my appreciation to Professor Jonas Sjöberg, my examiner, for accepting and welcoming me into the Mechatronics research group. A big thanks goes to all my past and present colleagues at Chalmers. Good discussions have always been going on that have made an inspiring working environment. I want to convey my appreciation to the HIAB partners, with special recognition of Hans Lyngbäck, for their collaboration in the project that sparked the initial ideas for this thesis. In addition, I am grateful to the people at the Department of Automatic Control and the RobotLab at Lunds Tekniska Högskola for hosting and welcoming me during the final phase of my doctoral studies.

A heartfelt acknowledgment goes to Sirius. Many solutions to the challenges I encountered in my Ph.D. journey were discovered during our walks together. I also want to express my appreciation to my parents and sisters, whose love and support throughout life has made this endeavor possible. Lastly, I extend my deepest thanks to my beloved Maggan for your patience, positivity, and unwavering belief in me. Your presence is a constant source of strength, and I feel exceptionally fortunate to have you by my side.

Albin Dahlin
December 2023

Acronyms

APF:	Artificial Potential Fields
DMP:	Dynamical Movement Primitives
DS:	Dynamical System
DSW:	Disjoint Star World
MPC:	Model Predictive Control
PbD:	Programming by Demonstration
PRM:	Probabilistic Roadmap
ProMP:	Probabilistic Movement Primitives
RHRP:	Receding Horizon Reference Path
RRT:	Rapidly exploring Random Tree
SEA:	Statically Evaluated Acceleration
SEDS:	Stable Estimator of Dynamical Systems
SOADS:	Starshaped Obstacle Avoidance through Dynamical Systems

Contents

Abstract i

List of Papers iii

Acknowledgements v

Acronyms vi

I Overview 1

1 Introduction 3
- 1.1 Motion Planning: An Overview 4
- 1.2 Dynamical Systems for Motion Planning 6
- 1.3 Problem Formulation 8
- 1.4 Contributions 9
- 1.5 Thesis Outline 9

2 Temporal Scaling of Dynamical Systems under Constraints for Path Preservation 11
- 2.1 Accommodating System Constraints 14
- 2.2 Velocity Constraints 15

2.3 Combined Velocity and Acceleration Constraints 17
2.3.1 Reactive vs Proactive Temporal Scaling 18
2.3.2 States of Infeasibility 19
2.3.3 Timing Law with Proactive Scaling 21
2.3.4 Smoothing by Constraint Neglect 24

3 Dynamical Systems for Navigating Star Worlds 27
3.1 Starshaped Sets and Star Worlds 29
3.2 Modulation of Dynamical Systems for Obstacle Avoidance in Star Worlds 30
3.3 Enhancing Convergence through Environment Modification . . 32
3.3.1 Excluding Points from the Starshaped Hull 33
3.3.2 Starshaped Hull for Strictly Starshaped Sets 34
3.3.3 ModEnv*: Forming a DSW 36
3.4 MPC Framework to Adopt System Constraints 38
3.4.1 Control Scheme 39
3.4.2 Control Law 43
3.4.3 Control Law Scheduler for Convergence Guarantees . . 44
3.5 Extension to Path-following MPC 48

4 Summary of Included Papers 51
4.1 Paper A 51
4.2 Paper B 52
4.3 Paper C 52
4.4 Paper D 53
4.5 Paper E 53
4.6 Paper F 54

5 Concluding Remarks and Future Work 55
5.1 Future Work 56

References 57

II Papers **67**

A Adaptive Trajectory Generation under Velocity Constraints using Dynamical Movement Primitives **A1**
1 Introduction . . . A3
2 Related Work . . . A4
3 Dynamical Movement Primitives . . . A5
4 Temporal Coupling for Constrained Velocity . . . A6
5 Evaluation . . . A12
5.1 Performance and Comparison . . . A13
5.2 Parameter Tuning . . . A15
6 Conclusions and Future Work . . . A16
Appendix . . . A17
References . . . A18

B Temporal Coupling of Dynamical Movement Primitives for Constrained Velocities and Accelerations **B1**
1 Introduction . . . B3
2 Dynamical Movement Primitives . . . B5
3 Temporal Coupling for Constrained Velocity and Acceleration . B7
3.1 Acceleration Constraints Imposed on $\dot{\tau}$ . . . B8
3.2 Velocity Constraints . . . B9
3.3 Feasibility of Acceleration Limits . . . B9
3.4 Path Traversal Speed . . . B10
3.5 Base Update Law . . . B10
3.6 Algorithm . . . B12
4 Simulations . . . B13
5 Experiments . . . B17
6 Conclusions . . . B18
References . . . B20

C Trajectory Scaling for Reactive Motion Planning **C1**
1 Introduction . . . C3
2 Problem Formulation . . . C5
3 Trajectory Scaling . . . C7
3.1 Constraint Transformation . . . C8
3.2 Timing Law for Maximal Delay Recovery . . . C10

3.3 Improved Feasibility C10
4 Simulations C14
5 Experiments C16
6 Conclusions C18
References C19

D Creating Star Worlds: Reshaping the Robot Workspace for Online Motion Planning **D1**
1 Introduction D3
2 Preliminaries D7
2.1 Mathematical Notation D7
2.2 Starshaped Sets D8
3 Problem Formulation D8
4 Starshaped Hull D10
4.1 Excluding Points from the Starshaped Hull D12
4.2 Starshaped Hull for Strictly Starshaped Sets D17
4.3 Global Starshaped Hull (*) D21
5 Forming Disjoint Star Worlds D23
5.1 Algorithm D24
5.2 Excluding Obstacle Points D27
5.3 Kernel Point Selection D29
5.4 Implementation D31
6 Obstacle Avoidance in a Starshaped World D34
6.1 Obstacle Representation D34
6.2 Examples D35
7 Conclusion D36
Appendix D38
References D43

E Obstacle Avoidance in Dynamic Environments via Tunnel-following MPC with Adaptive Guiding Vector Fields **E1**
1 Introduction E3
2 Preliminaries E5
2.1 Starshaped Sets and Star Worlds E5
2.2 Obstacle Avoidance for Dynamical Systems in Star Worlds E5
3 Problem Formulation E6

4 Control Design E7
4.1 Environment Modulation E8
4.2 Receding Horizon Reference Path E9
4.3 Tunnel-following MPC E10
5 Implementation Aspects E12
6 Results E13
7 Conclusion E17
References E18

F Autonomous Navigation with Convergence Guarantees in Complex Dynamic Environments F1
1 Introduction F3
1.1 Related Work F4
1.2 Contribution F5
2 Preliminaries F6
2.1 Notation F6
2.2 Starshaped Sets and Star Worlds F6
2.3 Obstacle Avoidance for Dynamical Systems in Star Worlds F7
3 Problem Formulation F7
4 Guaranteed DSW Generation F8
5 Setpoint Stabilization with Obstacle Avoidance F9
5.1 Environment Modification F10
5.2 DS-based Receding Horizon Reference Path F14
5.3 Model Predictive Controller F14
5.4 Stabilizing Backup Controller F16
5.5 Control Law Scheduler F17
5.6 Motion Control Scheme F18
6 Path-following with Obstacle Avoidance F20
7 Results F22
7.1 Setpoint Stabilization F23
7.2 Path-following F28
8 Conclusion F29
Appendix F30
References F34

Part I

Overview

CHAPTER 1

Introduction

Motion planning has been a central problem in robotics, studied extensively for decades [1]. Traditionally, robots have operated in safety cages performing a priori well-defined tasks, with many planning schemes having been developed for such static scenarios. Modern robotic systems are increasingly designed to operate in dynamic settings where they coexist with moving objects, including humans and other autonomous systems. Moreover, these robots are intended for tasks where the desired motion may vary during real-time execution. This trend is evident in a variety of applications, such as collaborative robots that work hand-in-hand with human employees [2], mobile robots for food delivery in restaurants [3], and unmanned aerial vehicles deployed for visual monitoring of civil infrastructure [4], to name just a few examples. In contrast to the traditional operating conditions, a robot must constantly adjust its planned trajectory when it operates in such dynamically changing environments.

Reactivity to detected changes in the robot environment is a key factor for correct and safe operation. It is important that these responsive actions are executed cautiously, adhering to any robot limitations. This thesis focuses on reactive motion planning and control that takes robot constraints into consideration.

1.1 Motion Planning: An Overview

The classical approach to planning robot motion, widely used in industrial manipulators, involves hard-coded point-to-point motions. The trajectory in such cases is defined by initial, goal and possible intermediate waypoints and can be computed using, e.g., trapezoidal acceleration motion [5], or polynomial interpolation [6]. While extensions for collision avoidance have been introduced, e.g., [7], the core method they enhance is primarily suited for structured, well-known environments. In later years, Programming by Demonstration (PbD) has grown in popularity as a simplified user-friendly way of programming the desired robot motion [8]–[10]. As the name suggests, PbD relies on learning from demonstrations performed by a human, e.g., from visual or kinesthetic demonstrations, instead of using explicit programming through machine commands. While early methods operated in a pure "record-and-play" fashion, modern PbD techniques focus on generalizing the taught skills to new situations or variations of the same task. This makes it suitable for applications where reactivity to changes in the environment is needed, such as grabbing a specific object that might be moving or obstructed by unforeseen obstacles.

Rather than programming the complete motion of the robot, the motion planning problem has primarily been considered as the problem to find a collision-free motion for a robot from an initial configuration to a goal configuration. Motion planning is here distinguished from path planning in that the latter is a purely geometric process focused solely on finding a collision-free path. In contrast, motion planning involves finding a feasible trajectory that adheres to the robot's constraints. In the initial stages of motion/path planning research, the focus was primarily on developing a *complete*[1] planning algorithm, meaning that when executed, the algorithm within a finite time either produces a valid solution if one exists or indicates failure. It has however been shown that the problem of complete motion planning is impractical from a computational complexity perspective [11]. An alternative planning paradigm which has grown popular is sampling-based techniques, where the two most influential methods are probabilistic roadmaps (PRM) [12] and rapidly exploring random trees (RRT) [13]. The main idea in such strategies depend on a collision-checking module enabling construction of a

[1] In this thesis a control theory jargon will be used, where the motion planning problem can be seen as analogous to setpoint stabilization, and completeness as analogous to global convergence.

graphic roadmap by connecting a set of points sampled from the obstacle-free space. In this way, no explicit representation of the environment is needed. While these methods are not complete, many provide probabilistic completeness, meaning that the probability of finding a solution converges to one as the number of samples increases. These are in their original formulation not suitable for online motion planning and various methods to reduce computational complexity have been proposed [14]–[17]. An alternative approach to handle dynamic or unknown environment is to locally deform a pre-computed candidate path online. Elastic strips [18] compute a tunnel region around the candidate path to enable locally deviating from the path to avoid obstacles. This is however dependent on the existence of a free passage around the candidate path. Another method for collision avoidance involves the use of barrier functions [19], [20], where proof of (almost) global convergence has been derived for scenarios with circular, adequately isolated obstacles [21], [22].

With the increase of computational power and development of robust numerical solvers for optimization problems, optimization-based planners have in later years emerged. These methods facilitate straightforward integration of system constraints while optimizing for additional criteria, for instance to achieve a smooth motion. Trajectory optimization planners, such as CHOMP [23] and TrajOpt [24], offer a way to online compute a feasible trajectory given a candidate path. However, they provide no guarantee to find a feasible solution and may be trapped in local minima [25]. An increasing number of local planning methods are based on Model Predictive Control (MPC). In MPC-based trajectory planning, only the closest future of the trajectory is planned in a receding horizon manner to track a specified setpoint [26], trajectory [27] or path [28]. By explicitly including information of the obstacle regions in the optimization problem [29]–[33] collision avoidance can be achieved also when dynamic obstacles appear along the target path. Due to the receding horizon nature of MPC, convergence guarantees are however not provided. Specifically, in environments with obstacles that obstruct a large area of the target path, the MPC solution may lead to local attractors at obstacle boundaries.

1.2 Dynamical Systems for Motion Planning

A different approach to tackle the motion planning problem is to express the desired motion in terms of a Dynamical System (DS). While generation of a complete path in one go for static scenes is possible with these methods, such formulations provide the capability to plan the next motion in response to current observations. As a result, the desired motion is promptly computed and the system can respond to perturbations in an immediate manner. Artificial Potential Fields (APF) [34] are widely used for obstacle avoidance in various scenarios [35]–[38]. The core concept of APF involves creating a scalar potential function that is maximal at the obstacle surfaces and minimal at the goal. This enables guiding the robot away from obstacles toward the goal by following the negative gradient of the potential function. A well-known drawback with APF is the possible existence of local minima [39] and various methods have been proposed to treat this issue. Harmonic potential fields [40] restrict the potential function to be a solution to Laplace's equation and provide a way to ensure (almost) global convergence. Numerical evaluation is however necessary to identify them because of the difficulty in constructing them [41]. Although closed-form solutions have been derived [42] these are limited to static scenarios. Navigation functions, defined in [43], are a special subclass of potential functions designed to be bounded and have gained a lot of interest in the field [44]–[47]. Navigation functions provide (almost) global convergence in a *disjoint star world* (DSW), i.e., when all obstacles are mutually disjoint and *strictly starshaped*[2]. However, the need for correct tuning of critical parameters dependent on the environment makes it hard to achieve full convergence in practice. A method has been proposed for deriving tuning-free navigation functions [48], but this presumes that the mapping from a star world to an environment of disjoint circular obstacle is provided. As an alternative to defining potential functions, direct construction of the underlying vector fields can be made. Vector fields given a predefined cell-decomposition of the environment was proposed in [49] and for environments with circular obstacles in [50]. A method to modulate a DS was presented in [51] which locally deform the original dynamics near obstacles such that collision avoidance is guaranteed for environments with convex obstacles. This has later

[2] A set is strictly starshaped if there exists a point such that any ray emanating from this point crosses the boundary once and only once. For a thorough definition of starshaped sets and star worlds, see Section 3.1.

been extended to apply for DSWs with proven (almost) global convergence to a goal point [52].

A repeated assumption enabling the proof of (almost) global convergence in many of the aforementioned methods is the premise of disjoint obstacles. However, closely positioned obstacles are frequently perceived as intersecting, e.g., when inflation is used to account for robot radius or safety margins, or in situations involving perception uncertainties. A modification of the perceived environment into disjoint obstacles is thus needed to apply these methods in practice with preserved convergence properties. Forest of Stars [53] presents a way to transform a set of intersecting starshaped obstacles into a DSW. However, the transformation relies on a particular structure of the intersection and correct tuning of scene-specific parameters which make it impractical for dynamic environments.

Several methods have been proposed also for PbD using DS formulation which provide the reactivity inherent of DS-based approaches. Notable works include Stable Estimator of Dynamical Systems (SEDS) [54], where the dynamics are based on Gaussian Mixture Models and provide asymptotic stability of a goal point, and Probabilistic Movement Primitives (ProMP) [55], that takes a probabilistic approach to encode a movement primitive as a probability distribution over trajectories. Another widely used framework is Dynamical Movement Primitives (DMP) [56]–[58], where convergence to a goal point is achieved with a stable second-order system while high flexibility in replicating complex trajectories is attained by including a diminishing additive nonlinear force. The frameworks have demonstrated capability to learn complex tasks such as playing table tennis [55], assistive exoskeleton control [59], and peg in hole assembly [60].

When employing a DS to generate a reference trajectory for a closed-loop controller, care must be taken to the robot constraints in order to avoid poor tracking performance and a distorted resulting path. Since the target trajectory is computed online, a strategy is needed to prevent the target from requesting unachievable motions. Such a strategy should ideally not compromise the inherent characteristics of the DS, such as path shape. Techniques have been proposed to online temporally scale DSs which inherently provide path preservation [57], [61], [62], although these do not directly address robot constraints.

1.3 Problem Formulation

This thesis studies reactive motion planning and control based on DS methods, focusing on the integration of robot constraints while maintaining the desired DS characteristics. In particular, it explores the following research questions:

Q1. How should a DS formulation be adjusted to integrate velocity and acceleration constraints, all while maintaining path consistency?

Q2. What are the requirements for representing the robot environment as a DSW? Can a constructive process be designed to online reshape the environment to form a DSW?

Q3. Can DS methods be incorporated in a combined motion planning and control scheme to address robot constraints while preserving the convergence characteristics of the DS?

An answer to question Q1 would enable close tracking when a DS serves as target generator within a closed-loop system that operates under constraints. Aside from the intuitive expectation that the target path should remain unaffected by velocity and acceleration constraints, the quest for path preserving solutions is motivated by scenarios where the original path is vital for successful task execution, such as when obstacle avoidance is addressed within the DS framework.

Question Q2 explores the extension of the range of practical scenarios where goal convergence can be achieved for DS methods designed to operate within DSWs. Moreover, an answer to Q2 includes an approach for achieving this in an online fashion.

Finally, question Q3 seeks to find a method for integrating a more general set of robot constraints, e.g., non-holonomic constraints, in a reactive motion planning and control approach. In contrast to Q1, the primary focus is not to maintain path integrity but to preserve the convergence properties of the applied DS method.

1.4 Contributions

The main contributions of this thesis are:

- A continuous-time method for temporal scaling of DSs to keep the velocity within specified limits while ensuring path shape invariance (Chapter 2 and Paper A).
- A discrete-time method for temporal scaling of DSs to keep the velocity and acceleration within specified limits while ensuring path shape invariance (Chapter 2 and Paper B, C).
- Theoretical analysis of the starshaped hull with introduction of related concepts (Chapter 3 and Paper D).
- An algorithm to reshape the obstacles perceived by the robot to improve convergence properties of DS methods operating in DSWs (Chapter 3 and Paper D, F).
- An MPC-based setpoint stabilization control scheme that utilizes the reactivity, collision avoidance, and convergence characteristics of a DS, along with the derivation of necessary conditions to guarantee convergence (Chapter 3 and Paper E, F).
- An MPC-based path-following control scheme that utilizes the reactivity, collision avoidance, and convergence characteristics of a DS (Chapter 3 and Paper F).

1.5 Thesis Outline

This thesis is divided into two parts. Part I consists of five chapters, and serves as an introduction to Part II. Chapter 1 introduces the motivation and scope of this thesis. In Chapter 2, a unified summary of the work for temporal scaling of DSs is presented, with particular attention to challenges arising when incorporating acceleration constraints. In Chapter 3, essential concepts for establishing a DSW are introduced, accompanied by an algorithm dedicated to this objective. The chapter also outlines a control scheme, covering both setpoint stabilization and path-following tasks. Chapter 4 provides a summary of the included papers from Part II. Finally, Chapter 5 concludes Part I with final remarks and outlines future research directions.

CHAPTER 2

Temporal Scaling of Dynamical Systems under Constraints for Path Preservation

In this chapter, we explore a scenario in which a DS is employed to generate a target trajectory in an online fashion for a closed-loop system, as depicted in Fig. 2.1. In this context, the DS could, for instance, encode a desired robot motion taught in a PbD fashion. In Section 2.1, temporal scaling is introduced in the context of DSs, and a continuous-time approach is proposed in Section 2.2 to impose velocity constraints on DSs. Finally, issues that may arise when considering both velocity and acceleration constraints in such a reactive motion planning setup are discussed in Section 2.3, and a discrete-time temporal scaling method to cope with these is presented.

To enable imposing constraints on both velocity and acceleration, the DS formulation is designed to be in a general second-order form

$$\dot{\zeta} = \begin{bmatrix} \dot{q} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} h^q(\zeta) \\ h^v(\zeta, \eta, t) \end{bmatrix} = h(\zeta, \eta, t), \quad \zeta(0) = \zeta_0, \tag{2.1}$$

where $\zeta \in \mathbb{R}^{2n}$ is the DS state with $q \in \mathbb{R}^n$ being the reference position, $h^q(\zeta)$ is a differentiable function, η is any external measurable input, and t is time. The notation $\dot{\zeta}(t) = \frac{d\zeta(t)}{dt}$ is used for the time derivative and the

time argument is omitted for convenience. The external input can be used for sensor-based adjustment of the dynamics to allow for obstacle avoidance [51], adapt motion to moving goal points [63], and adjusting the DS attractor landscape by reinforcement learning to improve the task performance according to a given metric [64]. We will refer to the DS (2.1) as the *nominal DS* with corresponding nominal velocity and acceleration

$$\dot{q} = h^q(\zeta), \tag{2.2}$$

$$\ddot{q} = \frac{\partial h^q(\zeta)}{\partial \zeta} h(\zeta, \eta, t). \tag{2.3}$$

It is clear from (2.3) that the structure in (2.1), with the effect from external input acting only on acceleration level, is instrumental to avoid introduction of sensory derivative in the target acceleration.

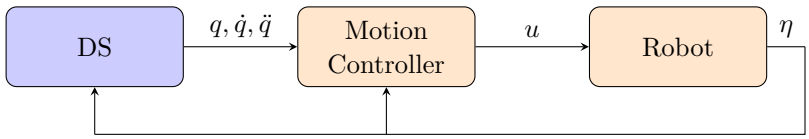


Figure 2.1: A DS generates the target for trajectory tracking by a motion controller. The sensor measurements, η, can be used both for feedback action in the controller and adapting the desired motion by online adjustments of the DS motion landscape.

Dynamical Movement Primitives

The DS formulation (2.1) admits direct use of trajectory descriptors at acceleration level such as artificial potentials [65] and SE-SODS [66]. This holds true also for DMP [57], which is the DS structure considered in both [67] and [68] (Paper A and B). For reference, we explicitly provide the formulation of DMP in terms of (2.1). The DMP dynamics are formed by a linear globally stable part with an added nonlinear virtual force acting at acceleration level as

$$\begin{aligned} h^q(\zeta) &= \frac{1}{\tau} v, \\ h^v(\zeta, \eta, t) &= \frac{1}{\tau} \left(K(q_g - q) - Dv + F(\xi) \right), \end{aligned} \tag{2.4}$$

with K and D being positive constants, and where the goal q_g and desired timing τ are considered as part of the external input η. The virtual force, F, is formed using weighted Gaussian basis functions and depends on a phase variable, ξ, which is an extended state of the system. With the added non-linearity, the DMP can, with increasing precision as the number of kernels grows, encode any trajectory along a sufficiently smooth path by learning of the weight parameters. The phase variable dynamics are typically defined as an autonomous system, with $\dot{\xi} = -\frac{1}{\tau}\xi,\ \xi(0) = 1$, although variations exists, e.g., to obtain periodic motions. The time-dependent solution for $\xi(t)$ can then be used in (2.4) to evaluate the virtual force as a time-dependent function $F(t)$, i.e., $F(e^{-\frac{1}{\tau}t})$. In cases when the phase variable dynamics are dependent on external factors, ξ can be seen as part of η.

Adopting DS formulations on velocity level

Trajectories defined in the velocity level, such as SEDS [54], ELM [69] and ProMP [55], with dynamics described as $\dot{q} = f_v(q, \eta, t)$, may be directly transferred to the second-order system as

$$\begin{aligned} h^q(\zeta) &= v, \\ h^v(\zeta, \eta, t) &= \frac{\partial f_v}{\partial q} v + \frac{\partial f_v}{\partial \eta}\dot{\eta} + \frac{\partial f_v}{\partial t}, \end{aligned} \tag{2.5}$$

where the function arguments of f_v are omitted for brevity. This assumes that f_v is partially differentiable and that the derivative of the external input $\dot{\eta}$ is measurable. To avoid the need for sensory derivatives, the corresponding term can be replaced by a feedback term

$$h^v(\zeta, \eta, t) = \frac{\partial f_v}{\partial q} v + \frac{\partial f_v}{\partial t} + K(f_v - v) \tag{2.6}$$

where K is a positive constant. Assuming the dynamics (2.5) are not dominated by fast changes in the external input, (2.6) closely captures the given dynamics. Alternatively, all partial derivatives can be omitted such that the acceleration dynamics are completely described in a feedback control manner $h^v(\zeta, \eta, t) = K(f_v - v)$.

2.1 Accommodating System Constraints

All real-world systems are subject to limitations in the realizable motion. Here, we will consider a system with velocity and acceleration constraints

$$|\dot{q}| \leq \dot{q}^{\max}, \tag{2.7}$$
$$|\ddot{q}| \leq \ddot{q}^{\max}, \tag{2.8}$$

where the inequalities are defined elementwise. For close reference tracking, the target trajectory should obey the system constraints (2.7)-(2.8). The target trajectory can be adjusted to respect the constraints by introducing reference governors [70], [71] or by direct saturation in the controller. An inherent feature of such methods is however that constraint satisfaction comes at the expense of path deviation. An alternative approach to consider the constraints (2.7)-(2.8) while preserving the original path is by means of temporal scaling. In temporal scaling, a path coordinate, s, is introduced which specifies the target state along the original target path. The progression of this path coordinate in time is regulated by a timing law. Adopting this technique, the DS (2.1) can be expressed as

$$\begin{aligned} \dot{\zeta} &= h(\zeta, \eta, s)\dot{s}, & \zeta(0) &= \zeta_0, \\ \ddot{s} &= w, & \dot{s}(0) &= 1,\ s(0) = 0, \end{aligned} \tag{2.9}$$

where w is determined by the timing law. The path speed, $\dot{s}$, can be viewed as a temporal scaling factor of the DS. When $\dot{s} > 1$, the trajectory speed is increased compared to the nominal trajectory, and when $\dot{s} < 1$, the speed is decreased. The nominal DS is obtained with the trivial timing law $w(t) = 0, \forall t$. Since all dimensions of $\dot{\zeta}$ are scaled with the same factor, $\dot{s}$, only the magnitude is modified while the direction is preserved. This results in path preservation, provided that the external impact remains unchanged. Contrary to most approaches in the temporal scaling literature where a static nominal trajectory is considered [72]–[75], the introduction of a path coordinate does not result in the classical path-velocity decomposition [76] since the path may be modified online due to the external input, η. That is, the resulting path is a priori unknown and cannot be computed in an offline fashion.

Remark 1: *In the DMP literature a more common notation for temporal scaling is based on the inversed path speed $\tau = \frac{1}{\dot{s}}$. This notation is used in [67] and [68] (Paper A and B). A major advantage by defining the temporal scaling*

in terms of the path speed $\dot{s}$, as in [77] (Paper C), is that this allows for easily performing a full stop by letting $\dot{s} = 0$. This is not practically achievable in terms of τ which would need to reach infinity to accomplish a full stop.

The velocity and acceleration for the temporally scaled DS (2.9) is given by

$$\dot{q} = h^q(\zeta)\dot{s}, \tag{2.10}$$

$$\ddot{q} = \frac{\partial h^q(\zeta)}{\partial \zeta} h(\zeta, \eta, s)\dot{s}^2 + h^q(\zeta)w. \tag{2.11}$$

It is clear that the magnitude of velocity and acceleration are affected by the path coordinate, with velocity being influenced by $\dot{s}$, and acceleration by both $\dot{s}$ and w. To a great extent, it is hence possible to comply with the system constraints without altering the underlying motion by appropriate selection of the timing law. However, it is important to note that instances of conflicting constraints, such that constraint violations are inevitable, cannot be entirely eliminated, as will be discussed later.

Typically, in scenarios when temporal scaling is used, it is desired to follow the path speed of the nominal trajectory whenever allowed by the constraints. In terms of the temporally scaled DS (2.9) this means that $\dot{s} = 1$ is desired. This will however lead to a delay with respect to the nominal task due to the target trajectory being slowed down during some time intervals. In some scenarios, it is of interest to recover not only the path speed but also the original path coordinate timing, i.e., $s = t$ is desired. We will consider both scenarios, and the problem can be stated as follows.

Problem 1. *Given the DS* (2.9) *and constraints* (2.7)-(2.8)*, construct a timing law for w which scales the DS such that the constraints are satisfied and recover the delay introduced by previous scaling by returning to, and follow, (a) the nominal path speed, $\dot{s} = 1$, or (b) the nominal path coordinate, $s = t$.*

2.2 Velocity Constraints

In this section we consider a system with velocity constraints only and ignore the acceleration bounds (2.8). By introducing the normalized velocity

$$\bar{\dot{q}} = (I\dot{q}^{\max})^{-1} h^q(\zeta)\dot{s}, \tag{2.12}$$

where I is the identity matrix with proper dimension, the constraints (2.7) can equivalently be described as $\bar{\dot{q}} \in [-1, 1]$. As proposed in [67] (Paper A)

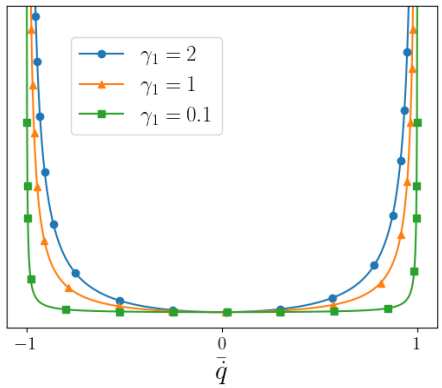


Figure 2.2: The barrier term $\gamma_1 \bar{q}\sigma(\bar{q})$ for three values of γ_1.

the normalized velocity can be utilized in the formulation of the timing law as

$$w = \gamma_0 \dot{s}(1 - \dot{s}) - \gamma_1 \dot{s} \bar{q}^T \sigma(\bar{q}). \tag{2.13}$$

Here, $\sigma(\bar{q}) = [\sigma_1(\bar{q}_1), ..., \sigma_n(\bar{q}_n)]^T$ is a vector function with $\sigma_i(\bar{q}_i)$ being barrier functions such that

$$\sigma_i : (-1, 1) \to (-\infty, \infty), \tag{2.14a}$$
$$\sigma_i(0) = 0, \tag{2.14b}$$
$$\frac{d\sigma_i}{d\bar{q}_i} \geq 0, \quad \bar{q}_i \in (-1, 1), \tag{2.14c}$$

and where $\bar{q}_i$ indicates the i^{th} element of the vector $\bar{q}$. A possible selection for σ is

$$\sigma_i(\bar{q}_i) = \frac{\bar{q}_i}{(1 - \bar{q}_i)(1 + \bar{q}_i)}, \tag{2.15}$$

with resulting barrier term for the one-dimensional case illustrated in Fig. 2.2. The timing law (2.13) consists of two terms: one driving $\dot{s}$ towards the nominal value of 1 and one acting as a barrier function decreasing $\dot{s}$ towards 0 as the velocity limits are approached. As stated by Theorem 1 in [67] (Paper A), applying the timing law (2.13) ensures the satisfaction of velocity constraints (2.7) by maintaining a bounded $\sigma(\bar{q})$. Moreover, from [67] (Paper A),

Property 1 and step 4 in the proof of Theorem 1 gives

$$0 < \dot{s} \leq 1. \tag{2.16}$$

Hence, the timing law (2.13) ensures a constant forward path traversal, tries to track but not exceed the nominal path speed, and guarantees that the velocity stays within the bounds (2.7).

2.3 Combined Velocity and Acceleration Constraints

When implementing (2.9) with timing law (2.13) in practical (discrete) settings, care must be taken when selecting the sampling period, Δt, and the gain parameters, γ_0 and γ_1. Specifically, $\bar{\dot{q}}$ may exit the region $(-1, 1)$ between two sampling instances such that $\sigma(\bar{\dot{q}})$ is not well-defined. To address this issue, a discrete implementation is formalized. The discrete formulation additionally simplifies considering velocity and acceleration constraints simultaneously. In particular, the constraints can be directly transformed into bounds on w_k. Here, the subscript k denotes quantities evaluated at the k^{th} time instant, i.e., $w_k = w(k\Delta t)$. The velocity constraint yields an upper bound, $w_k^{\max,\dot{q}}$, while the acceleration constraint yields both a lower and an upper bound, $w_k^{\min,\ddot{q}}$ and $w_k^{\max,\ddot{q}}$, respectively. Apart from the robot constraints, additional constraints are also included to avoid motion inversion ($\dot{s} \geq 0$), to prevent the generated trajectory being ahead of the nominal trajectory ($s \leq t$), and to allow for specifying a maximum path traversal speed ($\dot{s} \leq \dot{s}^{\max}$), where $\dot{s}^{\max} \geq 1$ is a tuning parameter. In all, the bounds for w_k are derived to

$$w_k^{\min} = \max(w_k^{\min,\ddot{q}}, w_k^{\min,\dot{s}}), \tag{2.17}$$

$$w_k^{\max} = \min(w_k^{\max,\ddot{q}}, w_k^{\max,\dot{q}}, w_k^{\max,\dot{s}}, w_k^{\max,s}). \tag{2.18}$$

Whenever $w_k \in [w_k^{\min}, w_k^{\max}]$, all velocity and acceleration constraints as well as the constraints related to the path coordinate are satisfied. For full derivation of the bounds, see [77] (Paper C).

Consider a saturated timing law as

$$w_k = \begin{cases} \text{sat}(\hat{w}_k, w_k^{\min}, w_k^{\max}), & w_k^{\min} \leq w_k^{\max}, \\ w_k' & w_k^{\min} > w_k^{\max}, \end{cases} \tag{2.19}$$

where $\hat{w}_k$ is a *base timing law* and w'_k is a *fallback timing law*. The function $\text{sat}(\cdot, a, b) = \min(\max(\cdot, a), b)$ is the saturation function, with a, b being the lower and upper bound, respectively. A timing law in the form (2.19) ensures constraint satisfaction whenever possible. A possible choice for fallback timing law is $w'_k = w_k^{\min,\dot{s}}$ as in [77] (Paper C). This will ensure feasibility, i.e., $w_k^{\min} \leq w_k^{\max}$, at next time step and the velocity limits are guaranteed to be satisfied at all times. Implicitly, this means that an acceleration bound will be neglected and some acceleration limit will be exceeded at this time instant. The following sections focus on the impact and design of the base timing law.

2.3.1 Reactive vs Proactive Temporal Scaling

One approach for the base timing law is a maximization approach, similar to [73], by setting $\hat{w}_k = w_k^{\max}$ such that the trajectory follow the nominal trajectory whenever no constraints are violated and slow down otherwise. This can lead to abrupt variations in w_k that in turn result in drastic alterations in acceleration. An alternative strategy is to include a filtered action towards the nominal path speed whenever delay has been introduced, similar to [72]. The base timing law can for this purpose be $\hat{w}_k = \gamma_0(1 - \dot{s})$. The filter smoothens the return to nominal path speed after saturation. These methods exhibit a reactive nature by adapting the temporal scaling when the limits are reached. We will refer to such approaches as *reactive temporal scaling*. This can be compared with the timing law (2.13) which adjust the temporal scaling as the velocity limits are approached. We refer to such an approach as *proactive temporal scaling*. Predictive techniques, such as [78], [79], provide an alternative means for proactive temporal scaling by computation of a scale factor which makes the trajectory feasible within a receding horizon. These are however formulated considering a predefined path. For the DS-based trajectory formulation (2.9) such methods would involve predictions of the external input, as well as possibly solving high-dimensional nonlinear constraints, making it impractical for real-time implementation. As will be discussed in the following, reactive temporal scaling may be prone to reaching states where no path acceleration, w_k, satisfying all system constraints exists.

2.3.2 States of Infeasibility

At some system states, the bounds (2.17)-(2.18) yield $w_k^{\min} > w_k^{\max}$ such that no path acceleration, w_k, exists that simultaneously satisfies all constraints. Such states are here referred to as *states of infeasibility*. Reaching such states typically occur when the system operates close to the acceleration limits. This can be understood by analysing the *statically evaluated acceleration* which is an important quantity in the timing law design in [68] and [77] (Paper B, C) and is defined as follows.

Definition 1: *The statically evaluated acceleration (SEA), denoted by ψ, is the resulting acceleration when assuming maintained path speed, i.e., assuming $w = 0$. That is, $\psi = \frac{\partial h^q}{\partial \zeta} h \dot{s}^2$.*

Using the SEA and the temporally scaled acceleration (2.11), the acceleration bounds (2.8) can be written as

$$|\psi_k + h_k^q w_k| \leq \ddot{q}_k^{\max}. \tag{2.20}$$

If the SEA for some index i exceeds the acceleration bounds, $|\psi_{k,i}| > \ddot{q}_i^{\max}$, the acceleration needs to be compensated appropriately through w_k to satisfy the constraints. Specifically, the following equivalences can be stated:

$$|\psi_k| < \ddot{q}^{\max} \qquad \Leftrightarrow w_k^{\min,\ddot{q}} < 0 < w_k^{\max,\ddot{q}}, \tag{2.21}$$

$$\exists i \ |\psi_{k,i}| > \ddot{q}_i^{\max}, \ \psi_{k,i} h_{k,i}^q > 0 \qquad \Leftrightarrow w_k^{\max,\ddot{q}} < 0, \tag{2.22}$$

$$\exists i \ |\psi_{k,i}| > \ddot{q}_i^{\max}, \ \psi_{k,i} h_{k,i}^q < 0 \qquad \Leftrightarrow w_k^{\min,\ddot{q}} > 0. \tag{2.23}$$

Moreover, for the cases (2.22) and (2.23), larger values of $|\psi_{k,i}|$ yields larger magnitude of $w_k^{\max,\ddot{q}}$ and $w_k^{\min,\ddot{q}}$, respectively. When using reactive temporal scaling in a scenario where the acceleration limit is approached, the path speed is not adjusted until the SEA is exceeding the acceleration limits. Depending on the current velocity, the path acceleration may need to be positive to satisfy the acceleration constraints, i.e., in case (2.23). This counterintuitive practice - to increase the path speed and thereby narrowing the feasible region at future time steps - is in fact the cause that drives the system to a state of infeasibility in many cases as illustrated by the following example.

Example 1. *A filter $\hat{w}_k = 1 - \dot{s}$ is used as base timing law for reactive temporal scaling of a DS with resulting trajectory depicted in Fig. 2.3. There are 2 time intervals where the SEA exceeds the acceleration bound while the*

velocity is of opposite sign, such that $w_k^{\min,\ddot{q}} > 0$ *in accordance with* (2.23). *The acceleration is hence saturated by increasing the path speed. Since* ψ_k *is proportional to* $\dot{s}_k^2$, *the magnitude of the SEA increases as well such that* $w_{k+1}^{\min,\ddot{q}} > w_k^{\min,\ddot{q}}$. *At some point,* $w_k^{\min,\ddot{q}} > w_k^{\max}$ *and the DS has reached a state of infeasibility.*

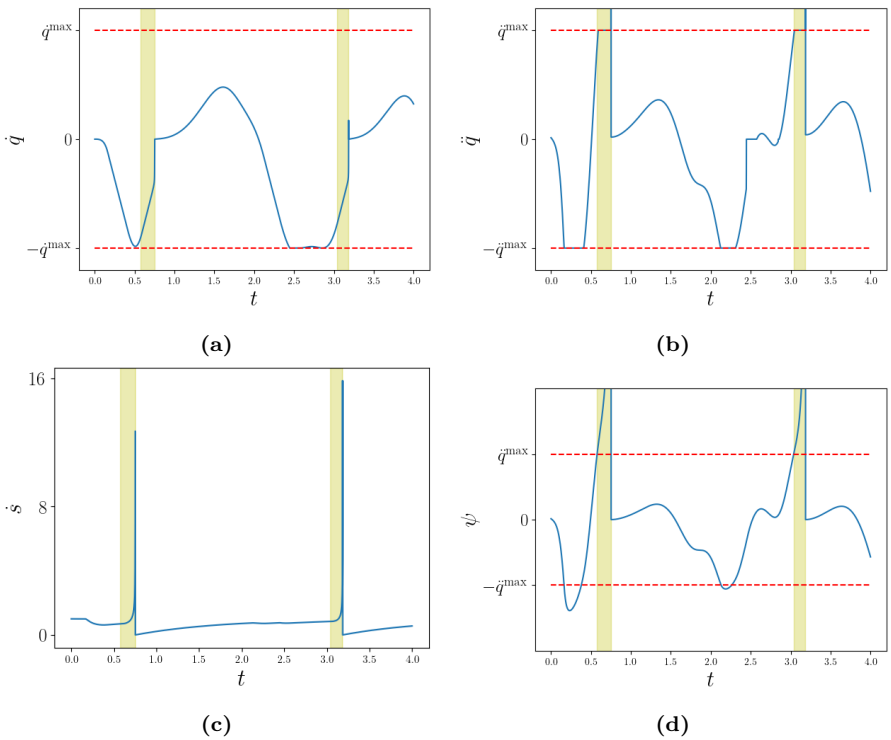


Figure 2.3: Illustration of Example 1. In the intervals where $w_k > 0$ to accommodate acceleration constraints, shown as yellow shaded regions, the DS eventually reach a state of infeasibility.

Rather than applying reactive temporal scaling when the limits are reached, a timing law aiming to slow down the trajectory in a proactive manner as the acceleration limits are approached is presented next.

2.3.3 Timing Law with Proactive Scaling

The aim of the base timing law presented in this section is to prevent ever reaching states of infeasibility. Motivated by the previous example and accompanying analysis, the base timing law will depend on the SEA. To this end, consider the ∞-norm of the normalized SEA

$$\bar{\psi}_k = \|(\ddot{q}^{\max} I)^{-1} \psi_k\|_\infty. \tag{2.24}$$

The value of $\bar{\psi}_k$ gives an indication of how close to the acceleration bounds the most critical dimension of the trajectory is at the current path speed. More specifically, $\bar{\psi}_k$ is approaching one as the SEA of any dimension is approaching the acceleration bound. It follows directly from (2.21) that $w_k^{\min,\ddot{q}} < 0 < w_k^{\max,\ddot{q}}$ if $\bar{\psi}_k < 1$ and from (2.23)-(2.22) that the acceleration must be compensated through w_k to satisfy the acceleration constraints whenever $\bar{\psi}_k > 1$. The idea with the proactive scaling is to keep $\bar{\psi}_k < 1$ at all times, and thereby maintaining the SEA within the acceleration bounds, for improved feasibility over time. Similar to the velocity, the magnitude of the SEA is decreasing with decreasing values of $\dot{s}_k$. With this in mind and inspired by the timing law (2.13) for bounded velocity, the base timing law in [68] (Paper B) is set to[1]

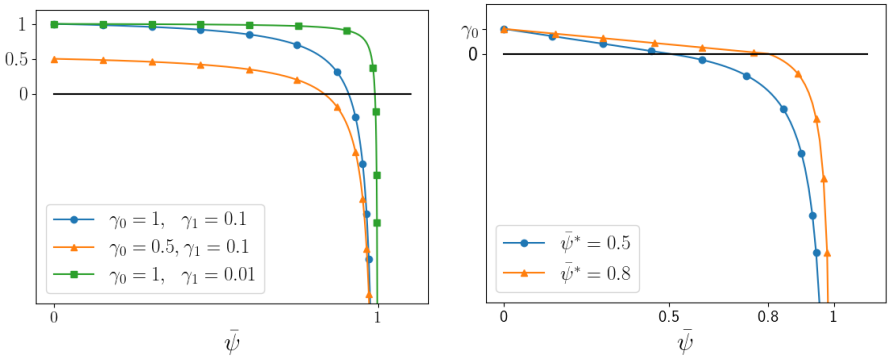
$$\hat{w}_k = \gamma_0(1 - \dot{s}_k) - \gamma_1 \hat{\sigma}(\bar{\psi}_k). \tag{2.25}$$

Here γ_0 and γ_1 are positive tuning parameters, and $\hat{\sigma}$ is defined as

$$\hat{\sigma}(\bar{\psi}) = \frac{\bar{\psi}}{\max(1 - \bar{\psi}, \epsilon)}, \tag{2.26}$$

with ϵ being a small positive constant. Similar to (2.13), the base timing law (2.25) consists of two terms: one driving $\dot{s}_k$ towards the nominal value of 1, and one acting to maintain $\bar{\psi}_k$ below the limit 1. The base timing law (2.25) is depicted in Fig. 2.4a. Contrary to the function σ used in (2.13), the function $\hat{\sigma}$ is not a proper barrier function, but is well-defined for all $\bar{\psi} \in \mathbb{R}$. In this way, $\hat{w}$ is well-defined also when the SEA exceeds the acceleration bounds. The filter (2.25) aims at tracking the nominal path speed as delay recovery. For nominal path coordinate tracking, the term $\gamma_0(1 - \dot{s}_k)$ could be replaced by a saturated gain of $(t_k - s_k)$.

[1] A direct derivation of the timing law (B.28) would in fact include a factor $\dot{s}_k$, yielding $\hat{w}_k = \dot{s}_k \left(\gamma_0(1 - \dot{s}_k) - \gamma_1 \hat{\sigma}(\bar{\psi}_k)\right)$. Since $\dot{s}_k = 0$ would then lead to $w_k = 0$, this would however prevent delay recovery when the trajectory has reached a full stop.



(a) $\hat{w}$ given by (2.25) for three values of γ_1 when $\dot{s} = 0$.

(b) $\hat{w}$ given by (2.27) for two values of $\bar{\psi}^*$ when $t - s > 1$.

Figure 2.4: Two proposed candidates for base timing law $\hat{w}$.

An alternative base timing law is presented in [77] (Paper C) as

$$\hat{w}_k = \begin{cases} \frac{\bar{\psi}^* - \bar{\psi}_k}{\bar{\psi}^*} \gamma_0 \min(t_k - s_k, 1), & \bar{\psi}_k \leq \bar{\psi}^*, \\ \frac{\bar{\psi}^* - \bar{\psi}_k}{1 - \bar{\psi}_k}, & \bar{\psi}^* < \bar{\psi}_k < 1, \\ w_k^{\min}, & \bar{\psi}_k \geq 1, \end{cases} \tag{2.27}$$

where $\gamma_0 > 0$ and $\bar{\psi}^* \in (0, 1)$ are tuning parameters. The parameters can be interpreted from Fig. 2.4b where (2.27) is depicted for all cases when $t_k - s_k \geq 1$, i.e., when the trajectory delay is at least one second. For other cases, the linear region $\bar{\psi}_k \leq \bar{\psi}^*$ is simply scaled with the delay $t_k - s_k$. In this region, $\hat{w}_k$ is positive leading to delay recovery. Outside the linear region, the path speed is reduced to relax the constraints at future time steps for improved feasibility. The behavior of $\hat{w}_k$ in the region $\bar{\psi}^* < \bar{\psi}_k < 1$ is logarithmic-like and goes towards negative infinity as $\bar{\psi}_k$ approaches one. The tuning parameter $\bar{\psi}^*$, which specifies the switching point between reduced path speed and delay recovery, can be interpreted as the target for $\bar{\psi}_k$ and thus determines how cautious the temporal scaling should be about approaching the acceleration limits. The parameter γ_0, being the feedback gain of the time delay, determines how aggressive the delay recovery should be.

The effect of including the proactive scaling is illustrated in Fig. 2.5. In this scenario, a DMP has been encoded using a 2D "omega"-shaped trajectory as

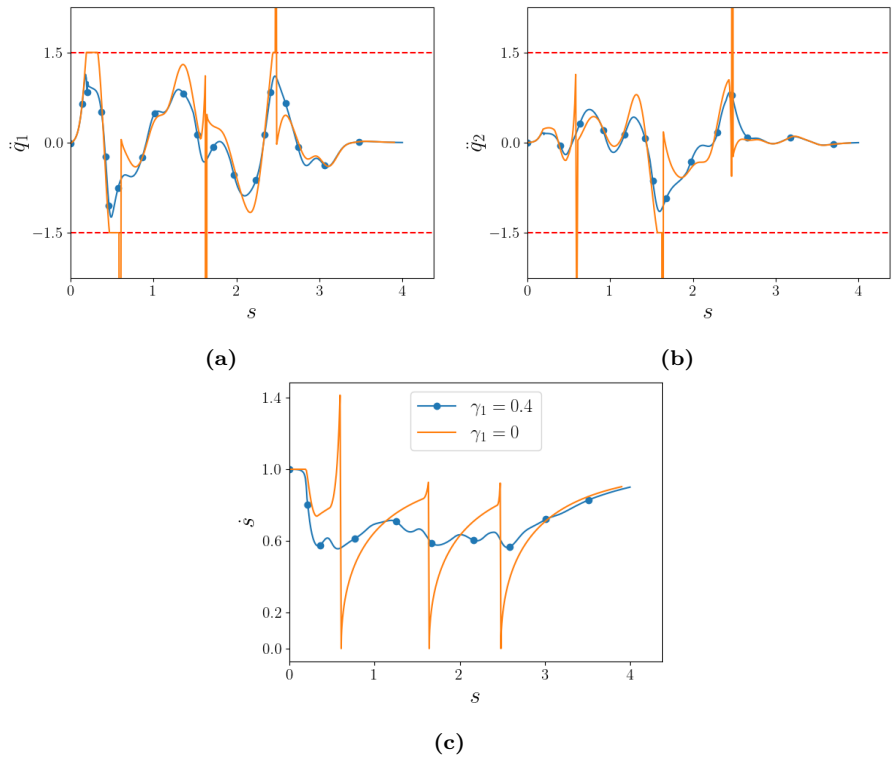


Figure 2.5: Comparison of including and excluding proactive scaling in (2.25).

shown in Fig. 3a in [67] (Paper A), and the nominal trajectory has a maximal acceleration of 3. A bound is set to half the nominal acceleration peak, i.e., $\ddot{q}^{\max} = [1.5\ 1.5]^T$. In Fig. 2.5 the resulting acceleration and path speed for the trajectory when using temporal scaling (2.25) for two cases are shown, both having the same gain for delay recovery, $\gamma_0 = 1$. The two differ in that one use proactive scaling with $\gamma_1 = 0.4$, while the other excludes the proactive scaling term, i.e., $\gamma_1 = 0$. It is clear that by reducing the path speed as the acceleration bounds are approached, the proactive scaling can prevent reaching states of infeasibility.

Remark 2: *As* $\gamma_1 \to 0$ *for* (2.25) *or* $\bar{\psi}^* \to 1$ *for* (2.27)*, the proposed timing law exhibits a more reactive rather than proactive nature.*

2.3.4 Smoothing by Constraint Neglect

The proactive scaling is introduced to relax the acceleration related bounds in a preventing manner as these are approached. This reduces the occasions when $w_k^{\min,\ddot{q}}$ is positive, i.e., when the path speed is increased to accommodate the acceleration constraints. It does however not completely remove this phenomenon. A method presented in [77] (Paper C) to completely avoid improper increase of path speed at the acceleration bounds is to ignore the constraint for the dimensions where the SEA exceeds the acceleration bound in the calculation of $w_k^{\min,\ddot{q}}$. Since $\bar{\psi}_k > 1$ for such cases, the base timing law (2.27) result in $\hat{w}_k = w_k^{\min}$ and the timing law (2.19) leads to a maximal reduction of path speed such that no other bound is exceeded. Also for the filter-based base timing law (2.25) the result will in practice lead to $w_k = w_k^{\min}$. This is realized by the upper bound $\hat{w}_k = \gamma_0(1 - \dot{s}_k) - \gamma_1 \frac{\bar{\ddot{q}}}{\epsilon} \leq \gamma_0 - \frac{\gamma_1}{\epsilon}$ which takes a large negative value (for reasonable tuning parameters where $\epsilon \ll \gamma_1$). If the path speed is sufficiently reduced, the SEA will at next time step no longer exceed the acceleration bounds and the bound will again be considered in $w_k^{\min,\ddot{q}}$.

The effect of including constraint neglect is illustrated using the same example as for Fig. 2.5, but the acceleration bound is reduced to $\ddot{q}^{\max} = [0.75\ 0.75]^T$. In Fig. 2.6 the resulting acceleration and path speed for the trajectory when using temporal scaling (2.27) for two cases is shown, both having the same tuning parameters, $\gamma_0 = 1$ and $\bar{\psi}^* = 0.7$. The two differ in that one use constraint neglect while the other does not. For the case without constraint neglect, the path speed is increased at two intervals (around $s = 1.5$ and $s = 2.5$) although the trajectory is close to/at the acceleration bounds. As for the scenario in Example 1 this drives the system to states of infeasibility. Although the acceleration bounds are exceeded to some extent for these intervals when constraint neglect is used, the acceleration is soon again below the bound as the path speed is continuously reduced. This results in a smoother trajectory.

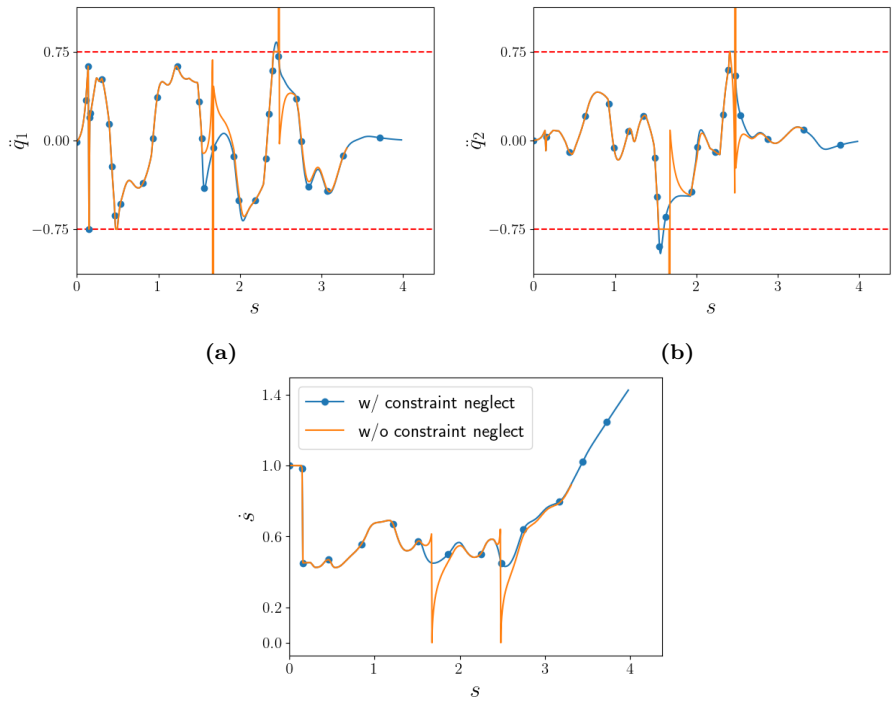


Figure 2.6: Comparison of using constraint neglect or not.

CHAPTER 3

Dynamical Systems for Navigating Star Worlds

This chapter addresses the problem of planning and executing robot motion to reach a goal position in a dynamic setting. The proposed method is built upon a DS approach suitable for navigation in DSWs. Given the foundational significance of starshaped sets in this chapter, an introduction to this concept is outlined in Section 3.1. Section 3.2 describes the featured DS, and in Section 3.3, a method is proposed to modify the robot environment into a DSW which enriches the applicability for the considered DS. Next, in Section 3.4, an MPC scheme for setpoint stabilization is outlined building on the DS and the environment modification. Finally, the control scheme is extended to apply for path-following control in Section 3.5.

This chapter considers a robot with a first-order differential kinematics model

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \\ p(t) &= \ell(x(t)), \end{aligned} \tag{3.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$ is the robot state, $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ is the control signal and $p \in \mathbb{R}^n$ is the robot position with $n = 2$ or $n = 3$. To stress the fact that the robot is kinematically modelled, we state the following assumption.

Assumption 1. *At any feasible state, there exists a control input such that the robot does not move, i.e.,* $\forall x \in \mathcal{X}, \exists u' \in \mathcal{U}$ s.t. $\frac{\partial \ell(x)}{\partial x} f(x, u') = 0$.

The robot operates in a workspace $\mathcal{W}(t) \subset \mathbb{R}^n$ which is strictly starshaped[1] or the full Euclidean space. The workspace is populated by a set of obstacles $\mathcal{O}(t) = \{\mathcal{O}_1(t), \mathcal{O}_2(t), ...\}$, where each obstacle $\mathcal{O}_i(t) \subset \mathbb{R}^n$ is either a simple polygon or convex. The obstacles are not restricted to be disjoint, but intersections may exist. The workspace and obstacles are jointly called the robot environment, denoted by $E(t) = \{\mathcal{W}(t), \mathcal{O}(t)\}$, and the environment has the corresponding free space $\mathcal{F}(t) = \mathcal{W}(t) \setminus \mathcal{O}_\cup(t)$, with notation $\mathcal{O}_\cup = \bigcup_{\mathcal{O}_i \in \mathcal{O}} \mathcal{O}_i$. The robot body is modelled as a point, assuming that any robot radius is taken into account by the environment model. For sound motion planning, some restrictions on the allowed workspace changes is considered, stated by the following assumption.

Assumption 2. *The workspace does not change such that the current robot position becomes an exterior point of the workspace.*

The objective is to find a control policy that enforces the robot to stay in the free space at all times while driving the robot to a specified goal position, as stated by Problem 2. In the following sections, we will omit the time notation for convenience unless some ambiguity exists.

Problem 2. *Given the robot dynamics* (3.1)*, the environment* $E(t)$*, and a goal position* $p_g \in \mathbb{R}^n$*, design a control scheme that computes* $u(t) \in \mathcal{U}$ *such that robot stays in the free space at all times and converges to the goal. That is,* $p(t) \in \mathcal{F}(t) \forall t$*, and* $\lim_{t\to\infty} p(t) = p^g$.

Remark 3: *Although* $\mathcal{O}(t)$ *formally contains only polygons and convex shapes, the formulation admit for more general complex obstacles as intersections are allowed. In particular, any shape can be described as a combination of several polygon and/or convex regions.*

Remark 4: *Including time-varying workspace description allows for scenarios with non-starshaped workspaces. One approach to achieve this involves partitioning the workspace into intersecting starshaped subregions, sequentially activated by a high-level planner, as in Fig. 7 of [80] (Paper F).*

Remark 5: *Since the robot radius is included in the environment description, two obstacles that physically do not intersect may do so in* $E(t)$.

[1] See definition in Section 3.1.

3.1 Starshaped Sets and Star Worlds

A set A is *starshaped with respect to* (w.r.t.) x if for every point $y \in A$ the line segment $l[x, y]$ is contained by A, i.e., $l[x, y] \subset A, \forall y \in A$. The set A is said to be *starshaped* if it is starshaped w.r.t. some point. The set of all such points is called the *kernel of* A and is denoted by $\ker(A)$, i.e., $\ker(A) = \{x \in A : l[x, y] \subset A, \forall y \in A\}$. Any convex set A is starshaped with kernel $\ker(A) = A$. The set A is *strictly starshaped w.r.t.* x if it is starshaped w.r.t. x and any ray emanating from x crosses the boundary at a single point. We say that A is strictly starshaped if it is strictly starshaped w.r.t. some point. The *starshaped hull of* A *w.r.t.* x, denoted by $SH_x(A)$, is the smallest starshaped set w.r.t. x containing A.

The robot environment is said to be a *star world* if all obstacles are strictly starshaped and the workspace is strictly starshaped or the full Euclidean space. Note that we do not restrict the obstacles to be mutually disjoint in a star world in contrast to the original notation in [53]. A *disjoint star world* (DSW) refers to a star world where all obstacles are mutually disjoint and where any obstacle which is not fully contained in the workspace has a kernel point in the exterior of the workspace, as exemplified in Fig. 3.1. For more information on starshaped sets and star worlds the reader is referred to [81] and [82] (Paper D).

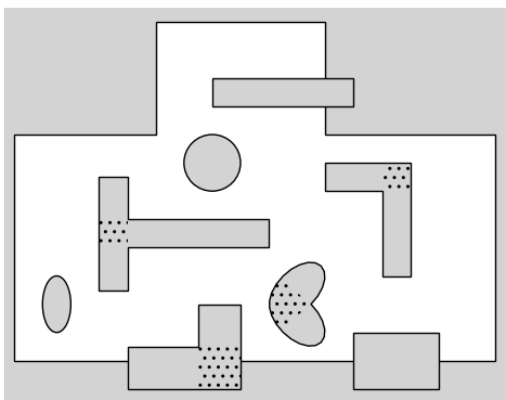


Figure 3.1: Example of a DSW. The kernels of all concave obstacles are shown as dotted regions.

3.2 Modulation of Dynamical Systems for Obstacle Avoidance in Star Worlds

An obstacle avoidance approach based on modulation of DSs was presented in [52] and extended in [83]. It assumes a holonomic robot where the kinematic model (3.1) gives $f(x,u) = u$, $\ell(x) = x$, and thus $\dot{p} = \dot{x} = u$. Moreover, no control bounds are considered, i.e., $\mathcal{U} = \mathbb{R}^n$. For convenience, and in the absence of better notation, we will refer to the method as Starshaped Obstacle Avoidance through Dynamical Systems (SOADS), as it is designed for operating in star worlds. In contrast to navigation functions which also allow for operation in star worlds, SOADS is not restricted to static environments with smooth surface obstacles. Rather, it allows for moving and deforming obstacles with boundaries that may include sharp edges. SOADS is based on online adjustments of an autonomous linear first-order DS with global convergence to p_g. For simplicity, we consider the nominal dynamics to be $\dot{p} = p_g - p$. The method relies on the assignment of a *center point* to each obstacle which must lie in the kernel interior so that a single boundary point exists in each direction from the center point. Given one obstacle $\mathcal{O}_i$, the modified DS is defined as

$$\dot{p} = M(p, \mathcal{O}_i)(p_g - p), \tag{3.2}$$

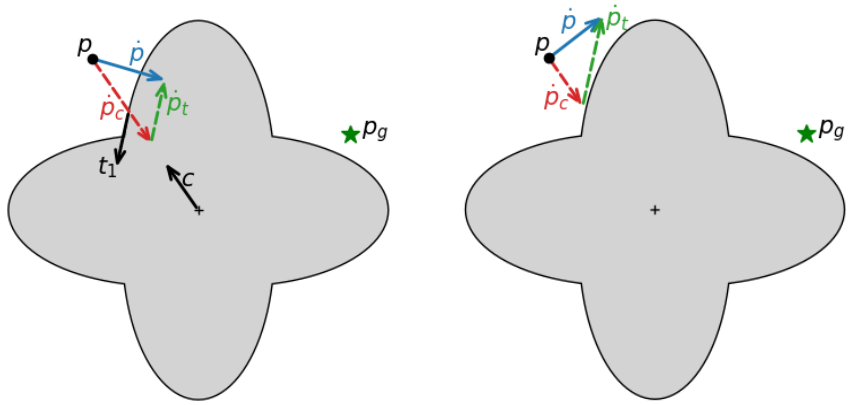
where the modulation matrix $M(\cdot) = T(\cdot)D(\cdot)T(\cdot)^{-1}$ is decomposed into a basis matrix T and an associated diagonal eigenvalue matrix D. The basis matrix $T = [c\ t_1\ ..\ t_{n-1}]$ depends on the *center direction*, c, – the direction from center point to current robot position – as well as the obstacle tangent space. By appropriate selection of the eigenvalues in D, the dynamics can be scaled along the directions in T such that collision avoidance can be obtained, as illustrated in Fig. 3.2. Specifically, by designing the scaling such that the velocity along the center direction vanishes at the obstacle boundary it is guaranteed that no obstacle region will be penetrated. In case of multiple obstacles, the velocity $\dot{p}$ is obtained by evaluating (3.2) individually for each obstacle and taking a weighted mean depending on obstacle proximity. For later reference, we denote the resulting dynamics for this procedure given a star world E as

$$\dot{p} = \nu(p, p_g, E). \tag{3.3}$$

The free space for any star world is positively invariant for the dynamics (3.3) such that collision avoidance is guaranteed for a trajectory following (3.3) from

any initial position $p(0) \in \mathcal{F}(0)$. Convergence conditions can also be derived under the following environment convergence assumption.

Assumption 3. *There exists a time instance after which the environment is static, i.e.,* $\exists t_s \in \mathbb{R}$ *s.t.* $E(t) = E(t_s), \forall t \geq t_s$.



(a) Using original attractor dynamics $\dot{p} = p_g - p$. The center direction, c, and obstacle tangent direction, t_1, are also depicted.

(b) Using SOADS dynamics $\dot{p} = M(p, \mathcal{O})(p_g - p)$.

Figure 3.2: The velocity vector $\dot{p}$ and the vectors $\dot{p}_c$ and $\dot{p}_t$ defined as the decomposition along the center direction c and the obstacle tangent direction t_1, respectively.

Assumption 3 is needed since finite convergence guarantees cannot be stated for generic scenarios with dynamic obstacles (consider the case where two separated gaps in a room are iteratively opening and closing). With Assumption 3, convergence to p_g is guaranteed if $E(t_s)$ is a DSW and no obstacle center point in $\mathcal{O}(t_s)$ is contained by the line segment $l[p(t_s), p_g]$. For more details on the formulation and theory of SOADS, the reader is referred to [52] and [83].

3.3 Enhancing Convergence through Environment Modification

As stated above, the obstacles must be disjoint for guaranteed convergence of the SOADS dynamics (3.3). Closely positioned obstacles are however frequently conceived as intersecting, breaking this condition and jeopardizing convergence to the desired goal. This occurs for instance when inflation of the obstacle region is used to account for robot radius or safety margins, or in case of perception uncertainties. To apply SOADS in practice with preserved convergence properties, online modification of the environment to obtain a DSW is thus needed. The obstacles in this DSW should fully cover the original obstacles to ensure collision avoidance. In addition, the center point for each obstacle should not be contained by $l[p, p_g]$. Since the center point is chosen from the kernel interior, the *convergence center point set* defined as $CCP(\mathcal{O}_i, p, p_g) = \text{int ker}(\mathcal{O}_i) \setminus l[p, p_g]$ must be nonempty to allow for selecting such center points. Finally, neither the robot nor goal position should be included in the extended obstacle regions. To this end, the following problem is stated.

Problem 3. *Given the environment E with corresponding free space $\mathcal{F}$, the robot position p, and the goal position p_g, create a DSW $E^\star = \{\mathcal{W}, \mathcal{O}^\star\}$ with corresponding free space $\mathcal{F}^\star$ that satisfies*

i. $\mathcal{F}^\star \subset \mathcal{F}$,

ii. $p \in \mathcal{F}^\star$,

iii. $p_g \in \mathcal{F}^\star$,

iv. $CCP(\mathcal{O}_i^\star, p, p_g) \neq \emptyset, \quad \forall \mathcal{O}_i^\star \in \mathcal{O}^\star$.

It is further desired that the resulting free space is as large as possible. In other words, the starshaped enclosing of the original obstacles should not be too conservative. A reasonable approach to ensure starshaped obstacles with minimal extended regions is to employ the starshaped hull. Directly applying the starshaped hull to represent obstacle regions is impractical for two reasons: 1) the starshaped hull does not account for excluding points from the expanded obstacle, potentially leading to the robot or goal position being an interior point of the resulting obstacles, and 2) the starshaped hull is not

inherently strictly starshaped, such that a direct utilization does not result in a star world in general. To address these issues, two new concepts have been defined in [82] (Paper D) which enable construction of strictly starshaped obstacles that exclude both the robot and the goal position. The introduced concepts are instrumental in the algorithm ModEnv* which is designed to solve Problem 3 and to be applied online along with SOADS as illustrated in Fig 3.3.

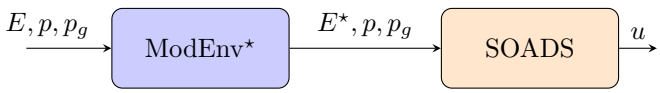


Figure 3.3: ModEnv* generates the star world $E^\star$ which is used by SOADS to compute control input. If $E^\star$ is successfully constructed as a DSW, convergence from the robot position p to the goal position p_g is guaranteed.

3.3.1 Excluding Points from the Starshaped Hull

The starshaped hull of the set A w.r.t. x, $SH_x(A)$, provides a starshaped enclosing of A, but it does not inherently provide any way to exclude specific points of interest, $\bar{X}$. The shape of $SH_x(A)$ depends on the point x and as a consequence we may have that $SH_x(A)$ is disjoint from $\bar{X}$ for some x, while it is not for another x. To enable an informed construction of the starshaped hull which is guaranteed to exclude $\bar{X}$, we introduce the *admissible kernel* defined as follows.

Definition 2: *Let $A \subset \mathbb{R}^n$ and $\bar{X} \subset \mathbb{R}^n$. The admissible kernel for A excluding $\bar{X}$, denoted by* ad ker$(A, \bar{X})$*, is the set such that the starshaped hull of A w.r.t. any $x \in$ ad ker$(A, \bar{X})$ does not contain any point in $\bar{X}$. That is,*

$$\text{ad ker}(A, \bar{X}) = \{x \in \mathbb{R}^n : SH_x(A) \cap \bar{X} = \emptyset\}. \tag{3.4}$$

Given the admissible kernel for an obstacle $\mathcal{O}_i$ with excluding set $\{p, p_g\}$, any point $x \in$ ad ker$(\mathcal{O}_i, \{p, p_g\})$ can be used to generate a starshaped obstacle $\mathcal{O}_i^\star = SH_x(\mathcal{O}_i)$ which contains $\mathcal{O}_i$ and excludes both the robot and the goal position. Several properties and expressions for computing the admissible kernel are found in [82] (Paper D). An illustration of how the admissible kernel can be used to generate a starshaped set that excludes some points is provided in Fig. 3.4.

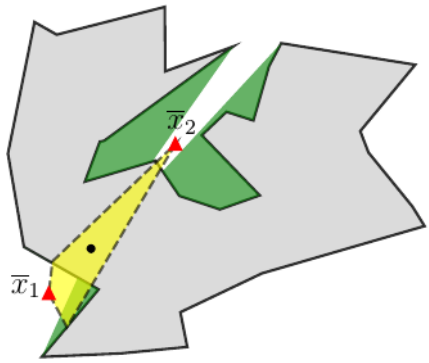


Figure 3.4: Example of a non-starshaped polygon A in grey, with two points to exclude, $\{\bar{x}_1, \bar{x}_2\}$, shown as a red triangles, and the resulting admissible kernel ad $\ker(A, \{\bar{x}_1, \bar{x}_2\})$ depicted in yellow. The hull extended region, $SH_x(A) \setminus A$, is shown in green for a point $x \in \text{ad ker}(A, \bar{X})$, shown as black dot. Clearly, $\bar{x}_1 \notin SH_x(A)$ and $\bar{x}_2 \notin SH_x(A)$ as desired.

3.3.2 Starshaped Hull for Strictly Starshaped Sets

Depending on the set A and the point x used for generating $SH_x(A)$, the kernel of the starshaped hull can be the singleton $\ker(SH_x(A)) = \{x\}$. Furthermore, from the manner that the starshaped hull is constructed, there may exist more than one boundary point along some direction from all (or the only) kernel point(s). In other words, it may not be strictly starshaped. An example of this is illustrated in Fig. 3.5a, where the starshaped hull is generated w.r.t. a point x, depicted as a black circle, and several boundary points of the hull are located along the four directions shown as red dashed lines from the singleton kernel x. For this reason, the starshaped hull in its original definition is not appropriate to apply when strictly starshaped sets are needed. Hence, it is not appropriate for constructing star worlds. To address this issue, we introduce the *starshaped hull with specified kernel.*

Definition 3: *Let $A \subset \mathbb{R}^n$ and $K \subset \mathbb{R}^n$. The starshaped hull of A with specified kernel K, denoted by $SH_{\ker K}(A)$, is defined as the smallest starshaped set such that $A \subset SH_{\ker K}(A)$ and $K \subset \ker\left(SH_{\ker K}(A)\right)$.*

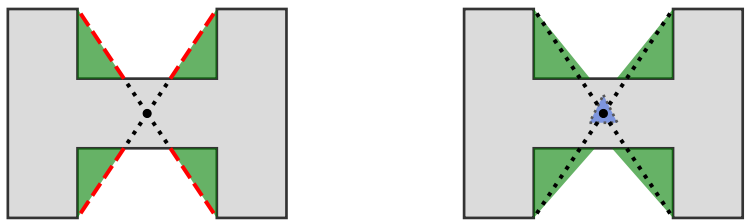
A useful property of the starshaped hull with specified kernel is that $CH(K)$ belongs to the kernel of $SH_{\ker K}(A)$, where $CH(K)$ is the convex hull of K. This can be used to guarantee the generation of a strictly starshaped set

by appropriate selection of the kernel points K, as stated by the following Proposition.

Proposition 1: *Let $A \subset \mathbb{R}^n$ and $K \subset \mathbb{R}^n$. The starshaped hull of A with specified kernel K, $SH_{\ker K}(A)$, is strictly starshaped if K contains $n+1$ affinely independent points.*

Proof. See proof in [82] (Paper D). □

In Fig. 3.5b, the starshaped hull with specified kernel given by three affinely independent points, i.e., a triangle, is depicted for a polygon, P. In contrast to the starshaped hull w.r.t. a single point, shown in Fig. 3.5a, $SH_{\ker K}(P)$ is strictly starshaped in accordance with Proposition 1 and therefore only one boundary point exists in each direction from any interior point of $CH(K)$.



(a) There exists several boundary points of $SH_x(A)$ along the red dashed lines in four directions from x.

(b) There exists only one boundary point of $SH_{\ker K}(A)$ along each direction from $x \in \text{int ker} SH_{\ker K}(A)$ when K contains three affinely independent points. $CH(K)$ is shown in blue.

Figure 3.5: A non-starshaped polygon, A, shown in grey with hull extended region shown in green.

While the admissible kernel is defined based on the starshaped hull w.r.t. a single point, it can be used to formulate a condition for kernel point selection to exclude points also from the starshaped hull with specified kernel.

Proposition 2: *Let $A \subset \mathbb{R}^n$, $K \subset \mathbb{R}^n$ and $\bar{X} \subset \mathbb{R}^n$. If $CH(K)$ is contained in* ad ker$(A, \bar{X})$*, no point $\bar{x} \in \bar{X}$ is included in $SH_{\ker K}(A)$.*

Proof. See proof in [82] (Paper D). □

Combining Proposition 1 and 2, we can conclude that $SH_{\ker K}(\mathcal{O}_i)$ is a strictly starshaped set with both robot and goal positions as exterior points if K is chosen as $n+1$ affinely independent points such that $CH(K) \subset \text{ad ker}(\mathcal{O}_i, \{p, p_g\})$. Several properties and derivations for computing the starshaped hull with specified kernel are found in [82] (Paper D).

3.3.3 ModEnv*: Forming a DSW

To construct a DSW that satisfies the conditions of Problem 3, the algorithm ModEnv*, outlined in Algorithm 1, was developed in [82] (Paper D) and refined in [80] (Paper F). The fundamental idea of ModEnv* is to create clusters of intersecting obstacles, followed by a starshaped enclosing of each cluster. To ensure that a strictly starshaped obstacle is obtained, starshaped hull with specified kernel is used, and to ensure that neither the robot nor the goal position is made an interior point of the generated obstacle, the kernel points are extracted from the admissible kernel of the cluster. These steps are illustrated in Fig. 3.6. In case intersections exist in the newly created obstacle set, the process is reiterated. If at some stage the admissible kernel is the empty set for a cluster, no DSW is found and ModEnv* terminates by returning the star world generated by convex decomposition of each obstacle. The core steps of ModEnv* do not account for the workspace boundary; they solely focus on reshaping the obstacles. Nonetheless, the modification into a DSW is achievable also for confined workspaces by employing a suitable kernel point selection, as detailed in Algorithm 4 in [80] (Paper F). In particular, the specified kernel for any obstacle cluster intersecting with the workspace boundary is selected in the workspace exterior, when feasible.

To state a sufficient condition for $E^\star$ being a DSW, the concept *DSW equivalent* is introduced, defined as follows.

Definition 4: *A star world $E = \{\mathcal{W}, \mathcal{O}\}$ is DSW equivalent if the set cl, formed by partitioning $\mathcal{O}$ into mutually disjoint clusters of obstacles, satisfies*

i) the obstacles in each cluster have intersecting kernels,

$$\ker_\cap(cl_i) \neq \emptyset, \ \forall cl_i \in cl, \tag{3.5}$$

ii) any cluster which intersects with the workspace exterior has an intersecting kernel region that to some extent lies in the workspace exterior,

$$cl_{i,\cup} \cap \text{ext}\mathcal{W} \neq \emptyset \Rightarrow \ker_\cap(cl_i) \cap \text{ext}\mathcal{W} \neq \emptyset, \quad \forall cl_i \in cl. \tag{3.6}$$

Algorithm 1: ModEnv*

Input : The environment $E = \{\mathcal{W}, \mathcal{O}\}$, the robot position p and the goal position p_g.

Output : A (disjoint) star world $E^\star$

Init : Initialize the cluster set $cl \leftarrow \{\{\mathcal{O}_i\} : \mathcal{O}_i \in \mathcal{O}\}$

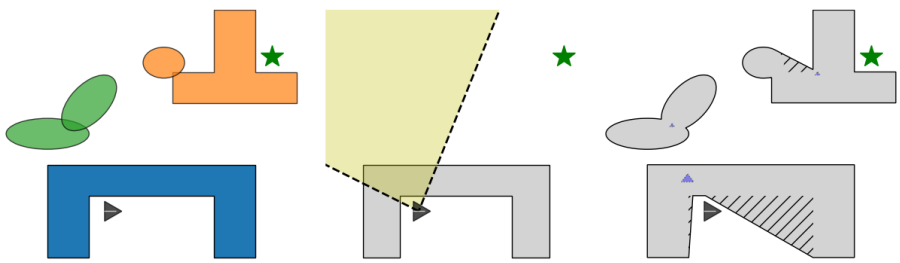
1 Compute ad $\ker(cl_i, \{p, p_g\})$ for each cluster, cl_i;

2 For each cl_i, select $n+1$ affinely independent points, K_i, such that $CH(K_i) \subset \text{ad ker}(cl_i, \{p, p_g\})$;

3 For each cl_i, generate a starshaped obstacle $\mathcal{O}_i^\star = SH_{\ker K_i}(cl_i)$;

4 Identify all intersections in $\mathcal{O}^\star$. If no intersection exists, return $\mathcal{O}^\star$;

5 Compute new clusters by combining obstacles in $\mathcal{O}$ corresponding to intersecting obstacles in $\mathcal{O}^\star$. Go to step 1;



(a) Clusters of intersecting obstacles illustrated by different colors.

(b) Admissible kernel for the non-starshaped polygon shown in yellow.

(c) Starshaped obstacles generated using starshaped hull. Extended regions are shown as dashed area and specified kernels in blue.

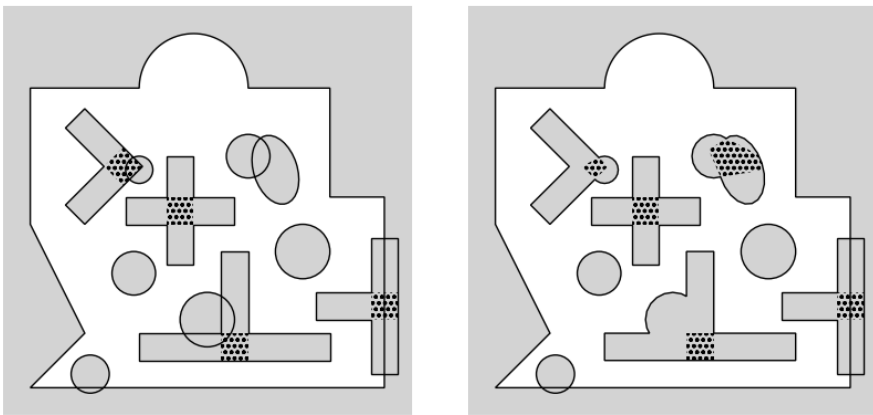
Figure 3.6: Illustration of the three main steps of ModEnv*.

Here $\ker_\cap(cl_i) = \bigcap_{\mathcal{O}_j \in cl_i} \ker(\mathcal{O}_j)$ is the kernel intersection of all individual obstacles in cl_i. An example of a DSW equivalent scene is shown in Fig. 3.7a. ModEnv* successfully generates a DSW for any DSW equivalent environment as stated by the following theorem and illustrated in Fig. 3.7b.

Theorem 1 (Guaranteed DSW generation)**:** *Given a DSW equivalent environment E with free space $\mathcal{F}$, a robot position $p \in \mathcal{F}$, and a goal position*

$p_g \in \mathcal{F}$, the resulting environment $E^\star$ from ModEnv* is a DSW with free space $\mathcal{F}^\star = \mathcal{F}$.

Proof. See proof in [80] (Paper F). □



(a) **(b)**

Figure 3.7: (a) A DSW equivalent environment and (b) the corresponding DSW environment generated by ModEnv*. The kernels for all non-convex obstacles are shown as dotted regions.

As discussed in [82] (Paper D), all obstacles generated by ModEnv* have nonempty sets $CCP(\mathcal{O}_i^\star, p, p_g)$ and any DSW returned by ModEnv* satisfies all conditions in Problem 3.

3.4 MPC Framework to Adopt System Constraints

SOADS can be directly applied in combination with ModEnv*, as in Fig. 3.3, to ensure obstacle avoidance for a holonomic robot with no control input bounds. Convergence to the goal is guaranteed if a DSW is succesfully generated by ModEnv*. To enable more general robot models, an MPC framework was derived in [84] (Paper E) and refined in [80] (Paper F). The target direction for robot motion is still determined by SOADS, but the MPC formulation allows to consider non-holonomic as well as input constraints. Moreover, the

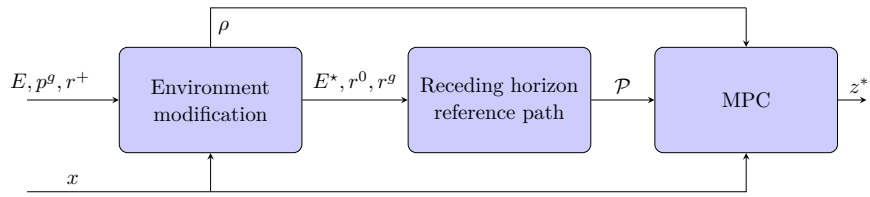


Figure 3.8: Embedding SOADS in an MPC framework.

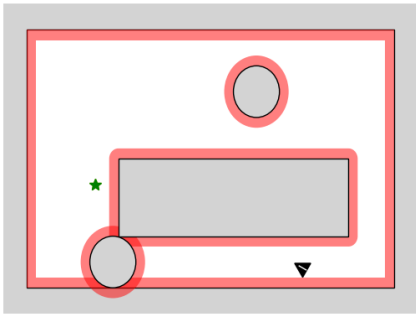
receding horizon optimization approach allows for formulating desired robot behavior, such as smooth movement and control input changes.

3.4.1 Control Scheme

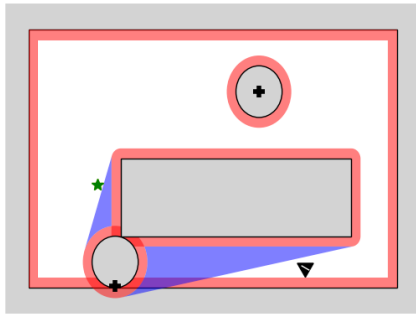
The general idea of the control scheme can be formulated in three steps as depicted in Fig. 3.8. First a DSW is constructed where any free point has an appropriately selected minimum clearance, ρ, to the original obstacles and workspace boundary. Next, a *receding horizon reference path* (RHRP) is generated by evaluating the SOADS dynamics over a defined distance in the constructed DSW. Consequently, any point along the RHRP maintains a minimum distance of ρ from the obstacles and workspace boundary. Finally, an MPC is formulated to induce motion along the RHRP while imposing constraints on path deviation to ensure adherence to the specified clearance. The control scheme is evaluated with a sampling interval Δt, such that the MPC solution z^* is constant over each period $t \in [t_k, t_k + 1)$ with $t_k = k\Delta t, k \in \mathbb{N}$. The steps are illustrated in Fig. 3.9 and elaborated more in the following.

Environment modification

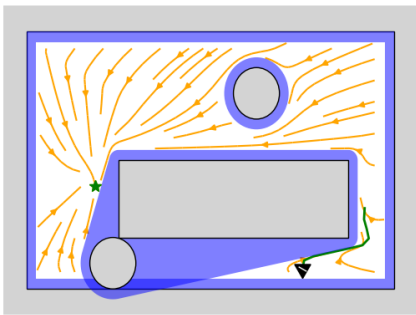
The framework relies on generating the RHRP with a (time-varying) minimum clearance, ρ, to the workspace boundary and all obstacles using SOADS. Hence, the environment considered when generating the RHRP should be a DSW, in addition to the minimum distance requirement. This ensures the utilization of collision avoidance and convergence properties inherent to SOADS. Ideally, the RHRP is a curve from the current robot position, p, to the goal, p_g (or an initial subset of this curve). However, the minimum clearance condition is violated if the robot or goal position is located closer than a distance ρ to



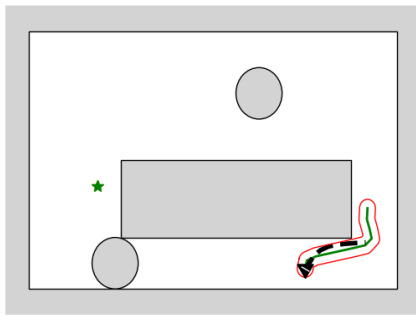
(a) A clearance (red) is added to all obstacles and the workspace boundary.



(b) Utilizing ModEnv*, the obstacle regions are enlarged (blue) such that the environment is a DSW.



(c) The RHRP (green line) is obtained by following the vector field of (3.9) (yellow arrows).



(d) A control sequence yielding a robot trajectory (dashed black) that stays within a ρ-neighborhood of the RHRP is computed using an MPC.

Figure 3.9: Illustration of the main steps in the MPC framework.

an obstacle. To account for these situations, the initial reference position, r_0, and goal reference position, r_g, are also defined. The procedure for the environment modification is depicted in Figs. 3.9a-3.9b. Initially, the clearance environment $E^\rho = \{\mathcal{W}^\rho, \mathcal{O}^\rho\}$, with corresponding free space $\mathcal{F}^\rho$, is defined, where $\mathcal{W}^\rho = \mathcal{W} \ominus \mathbb{B}[0, \rho]$ and $\mathcal{O}^\rho = \{\mathcal{O}_i \oplus \mathbb{B}[0, \rho] : \mathcal{O}_i \in \mathcal{O}\}$. The initial reference position is chosen as the point closest to a candidate r^+ within an initial reference set $\mathcal{P}^0 = \mathcal{F}^\rho \cap \mathbb{B}[p, \rho]$, and the reference goal is chosen as the point in $\mathcal{F}^\rho$ closest to p_g. The candidate r^+ can for simplicity be chosen as the cur-

rent robot position, but a more strategic selection can be made for improved convergence properties, as will be discussed in Section 3.4.3. The clearance ρ is set such that $\mathcal{P}^0 \neq \emptyset$. Since E^ρ may include both intersecting and non-starshaped obstacles, a new environment $E^\star$ is constructed using ModEnv* which is guaranteed to be a star world with corresponding free space $\mathcal{F}^\star \subset \mathcal{F}^\rho$ such that $r_0 \in \mathcal{F}^\star$ and $r_g \in \mathcal{F}^\star$.

Receding horizon reference path

The RHRP is given as a parametrized regular curve

$$\mathcal{P} = \{r \in \mathbb{R}^n : s \in [0, L] \to r(s)\} \tag{3.7}$$

with $L = Tw^{\max}$. Here, $T = N\Delta t$ is determined by the MPC horizon $N \in \mathbb{N}^+$ and $w^{\max} = \max_{u\in\mathcal{U},x\in\mathcal{X}} \| \frac{\partial \ell}{\partial x}(x) f(x,u) \|_2$ is the maximum linear speed which can be achieved by the robot. Given the normalized SOADS dynamics (3.3)

$$\bar{\nu}(\cdot) = \begin{cases} \frac{\nu(\cdot)}{\|\nu(\cdot)\|_2}, & \|\nu(\cdot)\|_2 > 0 \\ 0, & \|\nu(\cdot)\|_2 = 0 \end{cases}, \tag{3.8}$$

the mapping r is determined by solving the ODE

$$\frac{dr(s)}{ds} = \bar{\nu}(r(s), r_g, E^\star), \quad r(0) = r_0. \tag{3.9}$$

As the path is initialized in the star world $\mathcal{F}^\star$ and the dynamics are positively invariant in any star world, we have $\mathcal{P} \subset \mathcal{F}^\star \subset \mathcal{F}^\rho$. Thus, the tunnel region $\mathcal{P}^\rho = \mathcal{P} \oplus \mathbb{B}[0, \rho]$ is in the free space $\mathcal{F}$. The vector field of the dynamics (3.9) is illustrated in Fig. 3.9c and the RHRP is obtained by following the vector field from r_0 until an arc length of L is obtained or it reaches to r_g.

MPC

To find a control input which drives the robot along the RHRP, a nonlinear MPC is formulated. The objective is to find a solution which results in a fast movement while staying close enough to the RHRP such that collision avoidance is ensured. Adopting the path-following MPC framework [28], the system state is augmented with a path coordinate s, similar to the temporal scaling approach in Chapter 2, and the path speed w is introduced as a

decision variable. In this way, the mapping from reference path to reference trajectory is embedded as part of the optimization problem. Note that due to the normalized dynamics in (3.9), the reference speed coincides with the path speed, $\|\dot{r}\|_2 = \|\frac{dr(s)}{ds} w\|_2 = |w|$. For collision avoidance, a constraint is imposed on the tracking error, $\varepsilon(\tau) = \|r(s(\tau)) - \ell(\bar{x}(\tau))\|_2$, to guarantee that the robot stays in the tunnel region $\mathcal{P}^\rho$, as illustrated in Fig. 3.9d. To motivate a solution where the robot moves in a forward direction of $\mathcal{P}$, the cost function is designed to favor high reference speed while penalizing tracking error. The optimization problem for the MPC to find the control sequence, $z = \{\bar{u}_i,\ w_i : i \in [0, 1, .., N-1]\}$, at time t_k is given as

$$\min_z \int_0^T -c_w w(\tau) + c_\varepsilon \varepsilon(\tau) d\tau + J(z) \tag{3.10a}$$

subject to

$$\tau \in [0, T]: \quad \dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)),\ \bar{x}(0) = x(t_k) \tag{3.10b}$$

$$\bar{x}(\tau) \in \mathcal{X},\ \bar{u}(\tau) \in \mathcal{U}, \tag{3.10c}$$

$$\dot{s}(\tau) = w,\ s(0) = 0, \tag{3.10d}$$

$$s(\tau) \in [0, L],\ w(\tau) \in [0, w^{\max}], \tag{3.10e}$$

$$\bar{u}(\tau) = \bar{u}_i,\ w(\tau) = w_i,\ i = \left\lfloor \frac{\tau}{\Delta t} \right\rfloor, \tag{3.10f}$$

$$\varepsilon(\tau) \leq \rho, \tag{3.10g}$$

$$w_0 \geq \frac{\lambda \rho}{\Delta t}. \tag{3.10h}$$

Here, $\lfloor \cdot \rfloor$ is the floor function, and the notation $\bar{x}$ and $\bar{u}$ is used to denote the internal variables of the controller and distinguish them from the real system variables. The scalars c_w and c_ε are positive tuning parameters, and $J(z)$ is a regularization term for the control input which can be tailored for the robot at hand, if desired. Constraint (3.10b) initializes the predicted robot states to the current robot position and imposes the predicted states to satisfy the system dynamics (3.1), (3.10c) enforces the predicted states and control inputs to be admissible, and (3.10f) defines the control variables as piecewise constant over a sampling interval. Constraint (3.10d) specifies the path coordinate dynamics, while (3.10e) restricts s to provide a valid mapping $r(s)$ and ensures that the reference moves in forward direction along $\mathcal{P}$ with a reference speed less or equal to the maximum linear speed of the robot, $w^{\max}$. The final constraint (3.10h) is introduced to enforce initial movement

of the reference position, where the level of enforced movement depends on the tuning parameter $\lambda \in [0, 1)$.

3.4.2 Control Law

If no enforced initial movement of the reference is applied, i.e., setting $\lambda = 0$, problem (3.10) is feasible at all times. In particular, the trivial solution z', with $w_i = 0,\ \bar{u}_i = u'\ \forall i \in [0, 1, .., N-1]$, is always a feasible solution. Here $u' \in \mathcal{U}$ is a control input such that the robot does not move, $\frac{\partial \ell(\bar{x}(0))}{\partial x} f(\bar{x}(0), u') = 0$. While constraints (3.10b)-(3.10f) and (3.10h) trivially hold, the error constraint (3.10g) is ensured by the tailored selection of initial reference position $r(0) \in \mathcal{P}^0 \subset \mathbb{B}[\ell(x), \rho]$, such that $\varepsilon(\tau) = \varepsilon(0) \leq \rho\ \forall \tau \in [0, T]$. Since a solution to (3.10) exists at all times, it is valid to directly apply the optimal solution by defining the control law

$$u(t) = \bar{u}_0^*(t_k), \quad t \in [t_k, t_{k+1}), \tag{3.11}$$

where $\bar{u}_0^*(t_k)$ is extracted from the optimal solution $z^*(t_k)$. To derive collision avoidance guarantees, the following two assumptions are made.

Assumption 4. *The obstacles move slow compared to the sampling frequency such that the obstacle positions are constant over a sampling period, i.e.,* $\mathcal{F}(t) = \mathcal{F}(t_k),\ \forall t \in [t_k, t_{k+1})$.

Assumption 5. *The obstacles do not actively move into a region occupied by the robot, such that the implication* $p(t_k) \in \mathcal{F}(t_{k-1}) \Rightarrow p(t_k) \in \mathcal{F}(t_k)$ *holds.*

Assumption 4 suggests that the movements of obstacles can be accurately represented using discrete motion models, which is a common approach in development of sample-based control laws. Assumption 5 implies that the obstacles are aware of the robot's position and are not hostile, thereby preventing any collisions while the robot remains stationary. Note that this does not by itself imply collision avoidance when the robot is in motion. The following theorem provides guarantees of collision avoidance also when the robot is moving.

Theorem 2: *The trajectory for a robot with dynamics* (3.1) *and initial position* $p(t_0) \in \mathcal{F}(t_0)$ *under control law* (3.11) *is collision-free with respect to the environment* $E(t)$*, i.e.,* $p(t) \in \mathcal{F}(t), \forall t \geq t_0$.

Proof. Since the trivial solution z' is always feasible, a solution exists for (3.10) at all times. Any solution, z, to (3.10) satisfies $\varepsilon(\tau) < \rho(t), \forall \tau \in [0, T]$. The trajectory $\bar{x}(\tau)$ given by the solution to (3.10) at time instance t_k thus satisfies $\ell(\bar{x}(\tau)) \in \mathcal{P}^{\rho}(t_k) \subset \mathcal{F}(t_k), \forall \tau \in [0, T]$. Applying control law (3.11) leads to $x(t_k + \tau) = \bar{x}(\tau),\ \forall \tau \in [0, \Delta t]$ from (3.10b) and (3.10f). Hence, $p(t_k + \tau) = \ell(\bar{x}(\tau)) \in \mathcal{F}(t_k),\ \forall \tau \in [0, \Delta t]$. It follows from Assumption 4 that $p(t) \in \mathcal{F}(t), \forall t \in [t_k, t_{k+1})$ and from Assumption 5 that $p(t_{k+1}) \in \mathcal{F}(t_{k+1})$. Together this gives $p(t) \in \mathcal{F}(t), \forall t \in [t_k, t_{k+1}]$, which holds for any sampling instance t_k. □

3.4.3 Control Law Scheduler for Convergence Guarantees

Whereas collision avoidance is achieved with control law (3.11), local attractors away from the goal may arise in the workspace depending on control horizon and robot constraints. To improve attracting behavior towards the goal and derive convergence conditions, the enforced initial movement can be utilized. With $\lambda > 0$, the trivial solution z' is not longer a feasible solution and there is no longer a guarantee for existence of solution to (3.10). Consider for instance the example with a non-holonomic robot in Fig. 3.10 where the robot position is outside the region $\mathbb{B}[r(\lambda\rho), \rho]$. Depending on robot constraints, forcing an initial displacement such that $s(\Delta t) = \lambda\rho$ may lead to $\varepsilon(\Delta t) > \rho$, violating constraint (3.10g). To handle these cases, the concept of *stabilizing backup controller* (SBC) is defined.

Definition 5: *An SBC is a control law* $\kappa : \mathcal{X} \times \mathbb{R}^n \to \mathcal{U}$ *which, when applying* $u(\cdot) = \kappa(\cdot, r_0)$ *in* (3.1) *given* r_0*,*

i) renders the closed-loop error dynamics $\dot{e}$ *asymptotically stable in the origin for the error* $e(t) = p(t) - r_0$*,*

ii) does not allow the error to exceed its initial value, i.e., $\|e(t)\|_2 \leq \|e(t_0)\|_2,\ \forall t \geq t_0$.

The SBC must be designed for the robot at hand, but an example of an SBC for a unicycle robot is given in [80] (Paper F). It is here assumed that an SBC exists and has been designed.

To improve attracting behavior towards the goal, the control scheme from Fig. 3.8 is extended with a control law scheduler as depicted in Fig. 3.11. The control law scheduler determines the control law, μ, to be used over the

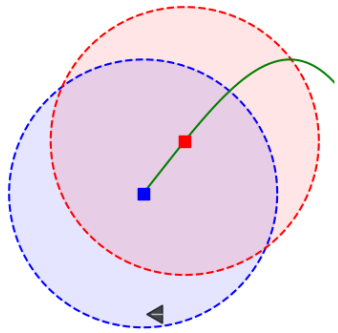


Figure 3.10: The initial part of $\mathcal{P}$ (green line) is depicted alongside r^0 (blue square) and $r(\lambda\rho)$ (red square) with corresponding regions $\mathbb{B}[r^0, \rho]$ (blue) and $\mathbb{B}[r(\lambda\rho), \rho]$ (red). If the robot constraints prohibits movement to achieve $p(\Delta t) \in \mathbb{B}[r(\lambda\rho), \rho]$, no solution to (3.10) exists.

next sampling period and the next initial reference position candidate, r^+, according to the logic

$$\mu(\cdot) = \begin{cases} \bar{u}_0^*, & \textbf{MPC MODE}, \\ \kappa(\cdot, r_0), & \textbf{SBC MODE}, \end{cases} \tag{3.12}$$

$$r^+ = \begin{cases} r(w_0^* \Delta t), & \textbf{MPC MODE}, \\ r_0, & \textbf{SBC MODE}, \end{cases} \tag{3.13}$$

where the two modes are

$$\begin{aligned} \textbf{MPC MODE} &: r_0 \neq r_g \text{ and (3.10) is feasible}, \\ \textbf{SBC MODE} &: \text{otherwise}. \end{aligned} \tag{3.14}$$

The control law (3.11) is hence applied when a solution to (3.10) is found and the RHRP is not the singleton set[2] $\mathcal{P} = \{r_g\}$. Otherwise, the feedback controller κ is applied with r_0 as setpoint. When in **MPC MODE**, a solution to the MPC problem exists and r^+ is chosen as the 1-step predicted reference position of the MPC solution. This encourages forward shift of the RHRP at the next sampling instance, while ensuring r^+ to stay in a ρ-neighborhood

[2] $r_0 = r_g$ implies that the RHRP dynamics (3.9) are $\frac{dr}{ds} = 0$.

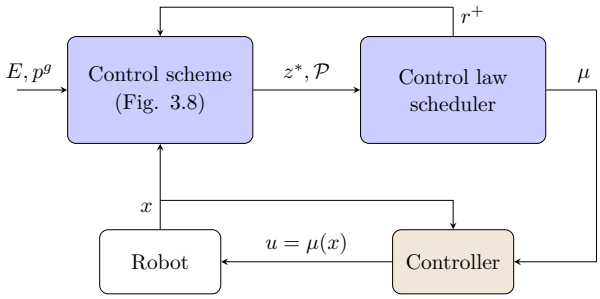


Figure 3.11: Control architecture for improved convergence.

of the robot due to (3.10g). When in **SBC MODE**, the control target is to realign the robot configuration to enable MPC feasibility at future sampling instances and the next initial reference candidate is chosen as the current initial reference, suggesting no forward shift of the RHRP.

While the trivial solution z' as defined in Section 3.4.2 no longer is a feasible solution to (3.10) when $\lambda > 0$, similar reasoning as for the proof of Theorem 2 can be made to state the following theorem for guaranteed collision avoidance using the modified control architecture.

Theorem 3: *The trajectory for a robot with dynamics* (3.1) *and initial position* $p(t_0) \in \mathcal{F}(t_0)$ *under control law* (3.12) *is collision-free with respect to the environment* $E(t)$*, i.e.,* $p(t) \in \mathcal{F}(t), \forall t \geq t_0$*.*

Proof. See proof in [80] (Paper F). □

Convergence conditions

The combination of enforcing initial reference movement ($\lambda > 0$), stimulating forward shift of the RHRP when (3.10) is feasible ($r^+ = r(w_0^* \Delta t)$), and realigning the robot at instances of infeasibility (SBC MODE) enables the derivation of conditions for guaranteed convergence. Specifically, convergence properties can be derived from the set

$$\mathcal{C}^{\bar{\rho}} = \mathcal{F}^{\bar{\rho}} \oplus \mathbb{B}[0, \bar{\rho}]. \tag{3.15}$$

The set $\mathcal{C}^{\bar{\rho}}$ can be interpreted as the free space obtained after an expansion by $\bar{\rho}$ of each obstacle, followed by a contraction of $\bar{\rho}$ of the resulting obstacle

regions. As a consequence, the positions from where guaranteed convergence cannot be derived, $\mathcal{F}\backslash\mathcal{C}^{\bar{\rho}}$, appear in the neighborhood of obstacle intersections, narrow passages, and in the neighborhood of concave obstacle vertices, as seen in Fig. 3.12. Hence, $\mathcal{C}^{\bar{\rho}}$ coincides with the original free space in an environment with convex obstacles where each obstacle maintains a minimum clearance of $\bar{\rho}$ from all others. To provide convergence properties to p_g, and not only to a $\bar{\rho}$-neighborhood of it, the following assumption is made.

Assumption 6. *There exists a time instance t' after which the workspace boundary and all obstacles stay at least at a distance $\bar{\rho}$ from the goal, i.e., $\|p_g - p_\delta\|_2 \geq \bar{\rho},\ \forall p_\delta \in \delta\mathcal{F}(t), \forall t \geq t'$.*

A convergence condition can now be derived dependent on a successful environment modification at time t_s, where t_s is the time instance from where the environment is static, as given by Assumption 3.

Proposition 3: *The trajectory for a robot with dynamics* (3.1) *following the control law* (3.12) *converges to p_g from any position $p(t_s) \in \mathcal{C}^{\bar{\rho}}(t_s)$ if $E^\star(t_s)$ is a DSW.*

Proof. See proof in [80] (Paper F). □

While the convergence in Proposition 3 is dependent on a successful environment modification, it directly follows from Theorem 1 that convergence to p_g from any position $p \in \mathcal{C}^{\bar{\rho}}$ is guaranteed if the clearance environment $E^{\bar{\rho}}(t_s) = \{\mathcal{W}^{\bar{\rho}}(t_s), \mathcal{O}^{\bar{\rho}}(t_s)\}$ is DSW equivalent, as stated by the following theorem.

Theorem 4: *The trajectory for a robot with dynamics* (3.1) *following the control law* (3.12) *converges to p_g from any position $p(t_s) \in \mathcal{C}^{\bar{\rho}}(t_s)$ if $E^{\bar{\rho}}(t_s)$ is DSW equivalent.*

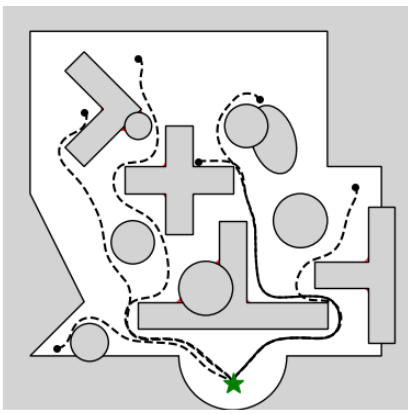
Proof. See proof in [80] (Paper F). □

The convergence properties are illustrated in Fig. 3.12 where the results from simulations using a unicycle robot are shown for two static scenes.

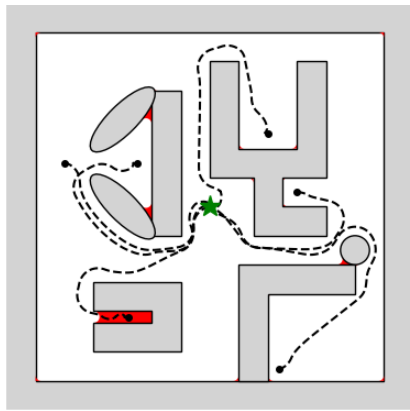
Remark 6: *Assumptions 2-5 trivially hold for a static environment while Assumption 6 can easily be attained by adjustment of $\bar{\rho}$.*

Remark 7: *While the derived MPC scheme is using a specific DS for the RHRP generation, namely SOADS, the approach can be applied also with other DS methods. Of course, the convergence conditions would need to be*

adjusted accordingly to fit the DS at hand. For instance, by replacing the use of ModEnv* by a circular approximation of the obstacles, the approach in [50] can be used with guaranteed convergence conditioned on all circular obstacles having a minimum distance $\bar{\rho}$.



(a) $E^{\bar{\rho}}$ is DSW equivalent and convergence to p_g from initial positions $p(t_0) \in \mathcal{C}^{\bar{\rho}}$ can be concluded.



(b) $E^\star(t_0)$ is a DSW for all given initial positions and convergence to p_g from any $p(t_0) \in \mathcal{C}^{\bar{\rho}}$ can be concluded.

Figure 3.12: A static set of obstacles $\mathcal{O}$ (grey), and the set $\mathcal{F} \backslash \mathcal{C}^{\bar{\rho}}$ (red) from where convergence cannot be stated. Travelled path (dashed black lines) to a goal position p_g (green star) is shown from different initial positions $p(t_0)$ (black dots).

3.5 Extension to Path-following MPC

The control scheme described in Section 3.4 considers the task of setpoint stabilization. In some applications, the robot should follow a predefined path rather than moving to a single point. The predefined, or *global path*, is here assumed to be given as a parametrized regular curve

$$\Gamma = \{p \in \mathbb{R}^n : \theta \in [0, \theta_g] \rightarrow p = \gamma(\theta)\}, \tag{3.16}$$

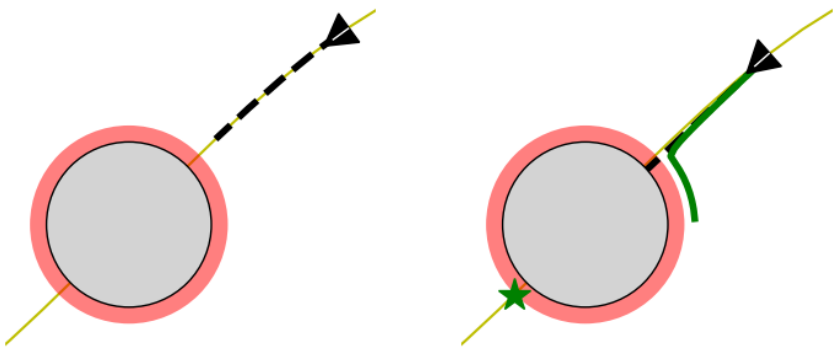
where $\gamma : \mathbb{R} \rightarrow \mathbb{R}^n$ is a natural parametrization of Γ. The scalar variable θ denotes the global path coordinate and is included in the control scheme as

an internal variable to specify the current global reference position, $\gamma(\theta)$. The considered robot task is to track Γ from start to end, assuming the path is obstacle-free. In the presence of obstacles obstructing the route, local path deviation should be performed to circumvent the obstacle before returning to, and continue following, the global path.

Compared to the setpoint control scheme, where a single point p_g specifies the target behavior, the global path Γ needs to be included in the control scheme for path-following. To this end, the generation of the RHRP is adjusted. Specifically, the computation of ρ and $\mathcal{P}$ carried out by the two leftmost blocks in Fig. 3.8 is substituted with the process outlined in Algorithm 2. This procedure is illustrated in Fig. 3.13. Instead of purely relying on the dynamics (3.9) to compute the RHRP, an alternative path, $\mathcal{P}^{\text{nom}}$, referred to as *nominal RHRP*, is computed which follows the global reference path Γ. The nominal RHRP is constructed to initially travel along the line from the initial reference candidate, r^+, to the current global reference, $\gamma(\theta)$, after which it follows Γ from $\gamma(\theta)$ until $\mathcal{P}^{\text{nom}}$ has an arc length of L^{nom}, where $L^{\text{nom}} \in [L, \infty)$ is a user-defined parameter. Should the nominal RHRP satisfy the clearance condition, i.e., $\mathcal{P}^{\text{nom}} \subset \mathcal{F}^{\bar{\rho}}$, it can safely be used as reference path in the MPC formulation for a collision-free trajectory generation. If the nominal RHRP does not achieve the desired clearance, the RHRP is obtained according to the setpoint strategy from Section 3.4.1 with the current global reference as setpoint. To ensure that the setpoint is collision-free and at an appropriate distance from the robot along Γ, the global path coordinate is first updated such that it maps to the first collision-free position along Γ after the endpoint of $\mathcal{P}^{\text{nom}}$.

Algorithm 2: RHRP for path-following

$\mathcal{P}^{\text{nom}} \leftarrow l[r^+, \gamma(\theta)]$ and follow Γ from $\gamma(\theta)$ until $\mathcal{P}^{\text{nom}}$ has an arc length of L^{nom};
if $\mathcal{P}^{\text{nom}} \subset \mathcal{F}^{\bar{\rho}}$ **then**
    $\mathcal{P} \leftarrow \mathcal{P}^{\text{nom}}$, $\rho \leftarrow \bar{\rho}$;
else
    Update θ such that $\gamma(\theta)$ is the first collision-free position (w.r.t. $\mathcal{F}^{\bar{\rho}}$) along Γ after the endpoint of $\mathcal{P}^{\text{nom}}$;
    Compute $\mathcal{P}$ and ρ as in Section 3.4.1 with $p_g = \gamma(\theta)$ as setpoint;
if MPC MODE and $r^+ \in \mathbb{B}[\gamma(\theta), \rho]$ **then**
    $\theta \leftarrow \theta + w_0^* \Delta t$;



(a) The nominal RHRP is collision-free with respect to the inflated obstacle (red) and is used as RHRP.

(b) The nominal RHRP intersects the inflated obstacle (red) and the RHRP (green line) is generated using the setpoint stabilization approach with goal position (green star) set as next collision-free position after the nominal RHRP.

Figure 3.13: The nominal RHRP (dashed black line) at two time instances where the global path (yellow line) is obstructed by an obstacle (grey).

CHAPTER 4

Summary of Included Papers

This chapter provides a summary of the included papers.

4.1 Paper A

Albin Dahlin, Yiannis Karayiannidis
Adaptive Trajectory Generation under Velocity Constraints using Dynamical Movement Primitives
Published in IEEE Control Systems Letters,
vol. 4, no. 2, pp. 438–443, Apr. 2020.
©2020 IEEE DOI: 10.1109/LCSYS.2019.2946761 .

This paper presents a method for online temporal scaling of DMPs to enforce velocity constraints. The formulation of the timing law is derived using a barrier function which ensures that the trajectory velocity stays within specified limits while the resulting path shape is preserved. The performance and stability of the proposed method are proven by Lyapunov-like arguments, and simulations are presented to verify the result.

The thesis author contributed with the problem formulation, control design,

theoretical derivation, simulation implementation, and writing of the paper.

4.2 Paper B

Albin Dahlin, Yiannis Karayiannidis
Temporal Coupling of Dynamical Movement Primitives for Constrained Velocities and Accelerations
Published in IEEE Robotics and Automation Letters,
vol. 6, no. 2, pp. 2233–2239, Apr. 2021.
©2021 IEEE DOI: 10.1109/LRA.2021.3058874 .

This paper presents a discrete-time online temporal scaling of DMPs to simultaneously address velocity and acceleration constraints. Direct lower and upper bounds of the scaling factor corresponding to the trajectory constraints are derived, and a saturated timing law is introduced, ensuring constraint satisfaction whenever feasible. Proactive scaling with respect to the acceleration constraints is introduced to decrease occasions of infeasibility. The method is evaluated by means of simulation and experiments on a 6-degrees of freedom robotic manipulator.

The thesis author contributed with the problem formulation, control design, simulation and experimental implementation, and writing of the paper.

4.3 Paper C

Albin Dahlin, Yiannis Karayiannidis
Trajectory Scaling for Reactive Motion Planning
Published in 2022 International Conference on Robotics and Automation (ICRA),
pp. 5242-5248, May 2022.
©2022 IEEE DOI: 10.1109/ICRA46639.2022.9811657 .

This paper extends the method presented in [68] (Paper B) to apply for a more general class of DSs, such that the temporal scaling is not specifically designed for DMPs. Moreover, the timing law is reformulated, offering several advantageous characteristics. It provides a simpler interpretation of the scaling variable, allows for performing full stop of trajectories, enables delay recovery to regain the original trajectory timing, and results in smoother

trajectories through the introduction of constraint neglect. The method is evaluated by means of simulation and experiments on a 6-degrees of freedom robotic manipulator.

The thesis author contributed with the problem formulation, control design, simulation and experimental implementation, and writing of the paper.

4.4 Paper D

Albin Dahlin, Yiannis Karayiannidis
Creating Star Worlds: Reshaping the Robot Workspace for Online Motion Planning
Published in IEEE Transactions on Robotics,
vol. 39, no. 5, pp. 3655–3670, Oct. 2023.
©2023 IEEE DOI: 10.1109/TRO.2023.3279029 .

This paper presents an algorithm, ModEnv*, which modifies the robot environment by merging and reshaping the obstacles to obtain a DSW. Several properties of the starshaped hull are established which are instrumental in the algorithm design. In particular, two novel concepts related to the starshaped hull are presented, *admissible kernel* and *starshaped hull with specified kernel*, which are both core components in ModEnv*.

The thesis author contributed with the problem formulation, theoretical derivation, algorithm design, simulation implementation, and writing of the paper.

4.5 Paper E

Albin Dahlin, Yiannis Karayiannidis
Obstacle Avoidance in Dynamic Environments via Tunnel-following MPC with Adaptive Guiding Vector Fields
Accepted for publication in 62nd IEEE Conference on Decision and Control (CDC 2023) .

This paper introduces a control scheme which ensures collision avoidance in dynamic environments. The method relies on generating a receding horizon path through a DS motion planning approach in combination with ModEnv*,

developed in [82] (Paper D). By formulating an MPC with path tracking constraints, an attractive behavior towards the goal while ensuring collision avoidance is achieved, even though explicit obstacle constraints are not included in the optimization problem formulation. The effectiveness of the control scheme is shown by means of simulation.

The thesis author contributed with the problem formulation, control design, theoretical derivation, simulation implementation, and writing of the paper.

4.6 Paper F

Albin Dahlin, Yiannis Karayiannidis
Autonomous Navigation with Convergence Guarantees in Complex Dynamic Environments
To be submitted [journal article] .

This paper extends the control scheme introduced in [84] (Paper E) to enhance convergence properties and enable implementation within confined workspaces. To achieve this, ModEnv*, introduced in [82] (Paper D), is adjusted, and necessary conditions for successful generation of a DSW are derived. Convergence conditions are also established for the complete setpoint control scheme. Moreover, a path-following control scheme with collision avoidance is developed as an extension of the setpoint controller. Various simulations are provided for both the setpoint and the path-following control scheme to illustrate the efficacy of the methods.

The thesis author contributed with the problem formulation, control design, theoretical derivation, simulation implementation, and writing of the paper.

CHAPTER 5

Concluding Remarks and Future Work

A method has been introduced for real-time adjustments of a DS through temporal scaling, accommodating velocity and acceleration constraints while preserving preserving path integrity (Question Q1). Although infeasibility may arise, this approach mitigates such risks by cautiously addressing acceleration constraints. Additionally, an approach to online *rationally* reshape the robot environment into a DSW has been developed (Question Q2) through analysis of the starshaped hull and introduction of related concepts. Here, "rationally" indicates that both robot and goal position remain in the free space. Finally, it has been shown that convergence properties of a DS method, namely SOADS [52], can be maintained for a substantial portion of the free space when adopted in an MPC framework (Question Q3). While the set from where convergence is guaranteed is slightly reduced compared to a direct use of SOADS, the method effectively expands the range of environments with assured convergence, including those not adhering to the DSW criteria.

In summary, this thesis has introduced techniques for integrating robot constraints into reactive motion planning and control through the utilization of DS methods, all while maintaining a primary emphasis on preserving desired characteristics of the DS.

5.1 Future Work

The temporal scaling techniques from [67], [68], [77] (Paper A, B, C) address velocity and acceleration bounds on the DS trajectory. Expanding this method to accommodate more general constraints is a natural extension. For instance, including the closed-loop control law within the constraint definition, similar to [72], would enable imposing joint-level constraints while the DS describes task space motion, or to impose force/torque constraints.

The theory on starshaped sets derived in [82] (Paper D) is focused on the starshaped hull. To analyze if similar concepts can be derived for Forest of Stars and explore feasibility for online derivation of such environments could lead to a less conservative reduction of the free space. While Theorem 1 provides conditions for guaranteed DSW generation by use of ModEnv*, the conditions are relatively strict. Thus, it is of interest to investigate method adjustments that broaden the range of environment configurations where successful DSW generation can be ensured. Additionally, further research towards reducing computational complexity is essential for compatibility with high-frequency controllers in 3D environments.

The control scheme for setpoint stabilization and path-following derived in [80], [84] (Paper E, F) relies on momentary observations of the environment. In scenarios where predictions of obstacle movements are available, it would be valuable to adapt the predictive controller to leverage these. Although directly including obstacle areas as forbidden regions in the MPC would break the convergence guarantees provided by Theorem 4, such an approach might prove advantageous in some scenarios. Another development direction involves incorporating dynamics and disturbances in the robot model, allowing for a robust control law, e.g., using tube-based MPC [85] or corridor MPC [86]. The current MPC formulation employs a conservative tracking error constraint based on a known minimum distance to the nearest obstacle. This could be relaxed by evaluating the actual distance to the obstacles after generating the RHRP to adapt the error bound dynamically along the RHRP, similar to [87]. Finally, to ensure the practical applicability of the MPC control scheme, performance should be assessed in a real-world scenario.

References

[1] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[2] A. Hentout, M. Aouache, A. Maoudj, and I. Akli, "Human–robot interaction in industrial collaborative robotics: A literature review of the decade 2008–2017," *Advanced Robotics*, vol. 33, no. 15-16, pp. 764–799, 2019.

[3] K. Byrd, A. Fan, E. Her, Y. Liu, B. Almanza, and S. Leitch, "Robot vs human: Expectations, performances and gaps in off-premise restaurant service modes," *International Journal of Contemporary Hospitality Management*, vol. 33, no. 11, pp. 3996–4016, 2021.

[4] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): A review of related works," *Visualization in Engineering*, vol. 4, no. 1, pp. 2213–7459, 2016.

[5] R. H. Castain and R. P. Paul, "An on-line dynamic trajectory generator," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 68–72, 1984.

[6] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1074, 1983.

[7] X. Chu, Q. Hu, and J. Zhang, "Path planning and collision avoidance for a multi-arm space maneuverable robot," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 1, pp. 217–232, 2018.

[8] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1371–1394, ISBN: 978-3-540-30301-5.

[9] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, collaborative robots: Challenges and opportunities.," in *IJCAI*, 2018, pp. 18–25.

[10] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[11] S. LaValle, "Motion planning: The essentials," *Robotics & Automation Magazine, IEEE*, vol. 18, pp. 79–89, Mar. 2011.

[12] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[13] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics*, A K Peters, 2001.

[14] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Trans. on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

[15] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *IEEE Int. Conf. Robotics and Automation*, 2020, pp. 9535–9541.

[16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[17] K. Yang, S. Moon, S. Yoo, *et al.*, "Spline-based rrt path planner for non-holonomic robots," *J. Intelligent & Robotic Systems*, vol. 73, pp. 763–782, 2014.

[18] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[19] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8129–8136.

[20] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2021.

[21] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination control for multi-robot networks using lyapunov-like barrier functions," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 617–632, 2016.

[22] K. D. von Ellenrieder, "Control barrier function based collision avoidance control for underactuated usvs," in *OCEANS 2022 - Chennai*, 2022, pp. 1–8.

[23] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[24] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems, Vol. IX, RSS*, 2013, pp. 1–10.

[25] Y. Yang, J. Pan, and W. Wan, "Survey of optimal motion planning," *IET Cyber-Systems and Robotics*, vol. 1, no. 1, pp. 13–19, 2019.

[26] D. Limon, A. Ferramosca, I. Alvarado, and T. Alamo, "Nonlinear mpc for tracking piece-wise constant reference signals," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3735–3750, 2018.

[27] C. Shen, Y. Shi, and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5796–5805, 2018.

[28] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2016.

[29] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[30] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Jul. 2019.

[31] B. Hermans, G. Pipeleers, and P. Patrinos, "A penalty method for nonlinear programs with set exclusion constraints," *Automatica*, vol. 127, 2021.

[32] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4467–4474.

[33] R. Oliveira, P. F. Lima, G. Collares Pereira, J. Mårtensson, and B. Wahlberg, "Path planning for autonomous bus driving in highly constrained environments," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2743–2749.

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

[35] Y. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, 1992.

[36] P. Wang, S. Gao, L. Li, B. Sun, and S. Cheng, "Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm," *Energies*, vol. 12, no. 12, 2019.

[37] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied Soft Computing*, vol. 77, pp. 236–251, 2019.

[38] S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[39] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, 1398–1404 vol.2.

[40] H. Feder and J.-J. Slotine, "Real-time path planning using harmonic potentials in dynamic environments," in *Proceedings of International Conference on Robotics and Automation*, 1997, pp. 874–881.

[41] C. Connolly, J. Burns, and R. Weiss, "Path planning using laplace's equation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1990, pp. 2102–2106.

[42] J.-O. Kim and P. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[43] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[44] S. G. Loizou and E. D. Rimon, "Mobile robot navigation functions tuned by sensor readings in partially known environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3803–3810, 2022.

[45] H. Tanner and A. Kumar, "Towards decentralization of multi-robot navigation functions," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 4132–4137.

[46] S. Paternain, D. E. Koditschek, and A. Ribeiro, "Navigation functions for convex potentials in a space with convex obstacles," *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 2944–2959, 2018.

[47] S. Hacohen, S. Shoval, and N. Shvalb, "Probability navigation function for stochastic static environments," *International Journal of Control, Automation and Systems*, vol. 17, pp. 2097–2113, 2019.

[48] S. G. Loizou, "The navigation transformation," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1516–1523, 2017.

[49] S. R. Lindemann and S. M. LaValle, "Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions," *International Journal of Robotics Research*, vol. 28, no. 5, pp. 600–621, 2009.

[50] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2014, pp. 2513–2518.

[51] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–464, 2012.

[52] 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.

[53] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

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

[55] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proceedings of Advances in Neural Information Processing Systems*, 2013, pp. 2618–2626.

[56] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2002, pp. 1398–1403.

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

[58] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, 2023.

[59] L. Peternel, T. Noda, T. Petrič, A. Ude, J. Morimoto, and J. Babič, "Adaptive control of exoskeleton robots for periodic assistive behaviours based on emg feedback minimisation," *PLOS ONE*, vol. 11, no. 2, pp. 1–26, Feb. 2016.

[60] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, Aug. 2015.

[61] M. Karlsson, F. B. Carlson, A. Robertsson, and R. Johansson, "Two-degree-of-freedom control for trajectory tracking and perturbation recovery during execution of dynamical movement primitives," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1923–1930, 2017.

[62] M. Karlsson, A. Robertsson, and R. Johansson, "Temporally coupled dynamical movement primitives in cartesian space," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9219–9226, 2020, 21st IFAC World Congress, ISSN: 2405-8963.

[63] L. Koutras and Z. Doulgeri, "Dynamic movement primitives for moving goals with temporal scaling adaptation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2020, pp. 144–150.

[64] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[65] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing). Springer London, 2008.

[66] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach* (Intelligent robotics and autonomous agents series). The MIT Press, 2022.

[67] A. Dahlin and Y. Karayiannidis, "Adaptive trajectory generation under velocity constraints using dynamical movement primitives," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 438–443, 2020.

[68] A. Dahlin and Y. Karayiannidis, "Temporal coupling of dynamical movement primitives for constrained velocities and accelerations," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2233–2239, 2021.

[69] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields," in *Proceedings of European Symposium on Artificial Neural Networks*, 2013.

[70] I. Kolmanovsky, E. Garone, and S. Di Cairano, "Reference and command governors: A tutorial on their theory and automotive applications," in *Proceedings of American Control Conference*, 2014, pp. 226–241.

[71] M. M. Nicotra and E. Garone, "Explicit reference governor for continuous time nonlinear systems subject to convex constraints," in *2015 American Control Conference (ACC)*, 2015, pp. 4561–4566.

[72] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 5, pp. 554–561, 1990.

[73] G. Antonelli, S. Chiaverini, and G. Fusco, "A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 162–167, 2003.

[74] C. Guarino Lo Bianco and F. M. Wahl, "A novel second order filter for the real-time trajectory scaling," in *Proceedings of International Conference on Robotics and Automation*, 2011, pp. 5813–5818.

[75] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Proceedings of Robotics: Science and Systems*, 2012.

[76] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[77] A. Dahlin and Y. Karayiannidis, "Trajectory scaling for reactive motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2022, pp. 5242–5248.

[78] M. Faroni, M. Beschi, C. Guarino Lo Bianco, and A. Visioli, "Predictive joint trajectory scaling for manipulators with kinodynamic constraints," *Control Engineering Practice*, vol. 95, 2020.

[79] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2017.

[80] A. Dahlin and Y. Karayiannidis, "Autonomous navigation with convergence guarantees in complex dynamic environments," Preprint available at https://arxiv.org/pdf/2306.12333.pdf, 2023.

[81] G. Hansen, I. Herburt, H. Martini, and M. Moszyńska, "Starshaped sets," *Aequationes mathematicae*, vol. 94, pp. 1001–1092, Dec. 2020.

[82] A. Dahlin and Y. Karayiannidis, "Creating star worlds: Reshaping the robot workspace for online motion planning," *IEEE Trans. Robotics*, vol. 39, no. 5, pp. 3655–3670, 2023.

[83] 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.

[84] A. Dahlin and Y. Karayiannidis, "Obstacle avoidance in dynamic environments via tunnel-following mpc with adaptive guiding vector fields," *Accepted for publication in 62nd IEEE Conference on Decision and Control (CDC 2023)*,
Preprint available at https://arxiv.org/pdf/2303.15869.pdf, 2023.

[85] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[86] P. Roque, W. S. Cortez, L. Lindemann, and D. V. Dimarogonas, "Corridor mpc: Towards optimal and safe trajectory tracking," in *2022 American Control Conference (ACC)*, 2022, pp. 2025–2032.

[87] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 378–383.