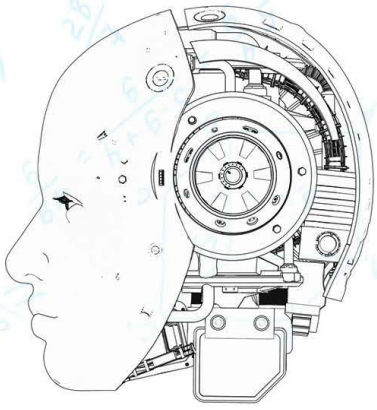
ADVANCES IN MODEL



TU Delft

PREDICTIVE CONTROL UNDER UNCERTAINTY

BALANCING PERFORMANCE, ROBUSTNESS, AND COMPUTATIONAL EFFICIENCY

FILIP SURMA

Advances in Model Predictive Control Under Uncertainty

Balancing Performance, Robustness, and Computational Efficiency

Advances in Model Predictive Control Under Uncertainty

Balancing Performance, Robustness, and Computational Efficiency

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
Thursday 23, April 2026 at 10:00 o'clock

by

Filip SURMA

Master of Science in Robotics,
University of Birmingham, United Kingdom
born in Świnoujście, Poland

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
dr. A. Jamshidnejad ,	Delft University of Technology, *promotor*
Prof. dr. ir. H. Hellendoorn	Delft University of Technology, *copromotor*

Independent members:

Prof. dr. ir. R. Negenborn	Delft University of Technology
Prof. dr. ir. U. Kaymak	Eindhoven University of Technology
Prof. dr hab. inż. P. Orłowski	West Pomeranian University of Technology in Szczecin
Prof. dr. S. Olaru	Paris-Saclay University
Prof. dr. ir. J. Alonso-Mora	Delft University of Technology

Prof. dr. ir. J. de Wit of Delft University of Technology has contributed greatly to the preparation of this dissertation.

Keywords: Model Predictive, Control (MPC), Robust Tube MPC, Dynamic Tube MPC, Robust Optimization, Fuzzy Optimization, Fuzzy Logic Framework, Hierarchical Model Predictive Control, Parent-Child MPC Architecture, Autonomous Exploration of Unknown Environments, Approximate MPC, Feasibility and Performance enhancement

Cover by: GetCovers

ISBN 000-00-0000-000-0

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ADAM Adaptive Moment Estimation. 74, 77

AMPC Approximate MPC. 8, 9, 70, 159

ASDD-TMPC Approximate State-Dependent Dynamic Tube-based MPC. xii, xvii, xix, 12, 71, 75, 77, 78, 116, 137, 157–160, 162, 163

Atan Arctangent. 77

AVN Angular Velocity Network. 76, 77

C-FLMPC Child Fuzzy-Logic-based MPC. xviii, xx, 120, 131–133, 135–137, 145, 147, 149

C-MPC Child MPC. xii, xviii, xx, 13, 85, 87, 89–101, 103–105, 131

EMPC Explicit MPC. 8, 70

FIS Fuzzy Inference System. ix, 7, 22–24, 38, 43, 44, 52, 55, 115, 118, 161

FL Fuzzy Logic. 2, 7, 10, 11, 29, 113–116, 118, 160–162

FLMPC Fuzzy-Logic-based MPC. ix, xviii, xx, 10, 11, 13, 111–114, 118–123, 125–128, 130, 131, 133, 134, 138, 139, 141, 144–149, 154, 155, 157, 158, 160–163

GA Genetic Algorithm. 24, 31, 43, 118, 146

LiF Leak-and-Fire. 74

LQR Linear Quadratic Regulator. 37, 38, 60, 86

LVN Linear Velocity Network. 76, 77

MPC Model Predictive Control. ix, xi, xvii–xx, 1–11, 13, 19–22, 24, 27, 31, 33, 35, 36, 39, 44–49, 52, 55, 69–71, 74, 75, 85–87, 89, 96–99, 103, 105, 111–118, 120, 138, 139, 141, 144, 145, 149, 154, 155, 157–163

NN Neural Network. 8, 22, 25, 38, 55, 69, 70, 73, 74, 76, 159

P-FLMPC Parent Fuzzy-Logic-based MPC. xviii, xx, 120, 131–138, 145–147, 149

P-MPC Parent MPC. xii, xviii, xx, 13, 85, 87, 89–105, 131, 163

PC-FLMPC Parent-Child Fuzzy-Logic-based MPC. 131, 138, 145–150, 161

PC-MPC Parent-Child MPC. xii, xviii, xx, 10, 13, 85, 87, 89–91, 93, 96–105, 111–114, 118, 131, 157–161, 163

PID Proportional–Integral–Derivative. 71

PS Particle Swarm. 31, 44–46, 118, 138, 161

ReLU Rectified Linear Unit. 70, 75

RL Reinforcement Learning. 24, 52, 70, 162

RMPC Robust MPC. xvii, xix, 5, 6, 9, 10, 24, 25, 40, 52, 69–71, 75, 111, 112, 158, 163

ROS2 Robotics Operating System 2. 76

SaR Search-and-Rescue. 2, 19–22, 29, 31, 32, 35, 37, 39, 40, 56, 111–114, 116–119, 122–128, 130, 138, 149, 150, 163

SDD-TMPC State-Dependent Dynamic Tube-based MPC. ix, xi, xii, xvii, xix, xx, 9–12, 19–23, 25–37, 39–56, 63, 64, 69–72, 75, 77, 78, 80, 105, 116, 157–163

SGD Stochastic Gradient Descent. 74, 75

SMPC Stochastic MPC. 5, 9, 139, 158, 163

SNN Spiking Neural Network. xvii, xix, 9, 10, 12, 69–71, 74–77, 80, 159–162

SQP Sequential Quadratic Programming. 31, 104

TMPC Tube-based MPC. ix, xi, xii, xvii, xix, 6, 7, 9–12, 19–22, 24, 25, 27–29, 31, 33, 35, 36, 39, 40, 43–49, 51–55, 61, 62, 67, 69, 70, 86–91, 93–105, 157–160, 162, 163

TSK-FIS Takagi-Sugeno-Kang Fuzzy Inference System. 24, 32, 33, 38

SUMMARY

Model Predictive Control (MPC) is an advanced control method that utilizes a model to make predictions about the state evolution for the controlled system, as well as an optimization solver to compute an optimal future control trajectory that satisfies state and control constraints. Inaccuracies in model predictions, caused by modelling errors or external disturbances, significantly hinder the performance of MPC and potentially lead to constraint violations. To address these challenges, Robust MPC (RMPC) explicitly accounts for uncertainties in the model, enhancing the control reliability. Tube-based MPC (TMPC) is a well-known RMPC method, where a nominal MPC generates a reference state trajectory, a tube (i.e., a trajectory of model uncertainties centered around the reference trajectory), and an ancillary control law that ensures the actual states of the system remain within this tube —following the reference trajectory, despite uncertainties.

This thesis presents a critical analysis of current state-of-the-art MPC methods, and proposes novel theoretical developments, architectures, and extensions for effective and computationally efficient handling of uncertainties via MPC . These contributions are rigorously supported by formal proofs. Furthermore, the proposed control frameworks are systematically evaluated through dedicated computer-based simulations, with comparisons drawn against existing methods in terms of optimality, robustness, and computational complexity.

The thesis begins with introducing State-Dependent Dynamic Tube-based MPC (SDD-TMPC), an extension of TMPC designed to more effectively handle the variability of model uncertainties and environmental disturbances. By leveraging available information about state-dependent uncertainties,SDD-TMPC enhances optimality and reduces risks of infeasibility, while maintaining the same level of robustness as TMPC. Although SDD-TMPC demonstrates applicability to systems with varying uncertainties across the state space, its practical implementation is limited by high computational demands.

To mitigate this limitation, Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC) is developed. This approach employs Spiking Neural Network (SNN) to approximate the behavior of SDD-TMPC. SNNs were selected for their event-driven processing and biologically inspired efficiency, offering significant advantages for low-power, real-time control. Recognizing that this approximation introduces additional uncertainties, and thus the risk of insufficient robustness to uncertainties, the SDD-TMPC framework is extended to incorporate these approximation errors as additional state-dependent disturbances, thereby preserving robustness. The reduced computational requirements of spiking neural networks enable implementation on

resource-constrained platforms, such as small-scale robotic platforms.

Next, the Parent-Child MPC (PC-MPC) architecture is proposed to further reduce computational complexity across a wide range of MPC frameworks. Compatible with both tube-based and deterministic MPC, the PC-MPC architecture decomposes the optimization problem into two linked problems: The Parent MPC (P-MPC) addresses long-term stability and constraint satisfaction, while the Child MPC (C-MPC) focuses on short-term stability and disturbance rejection. P-MPC communicates additional constraints to C-MPC, which determines and executes control strategies. This hierarchical approach, which guarantees robustness and stability, is extendable to systems with complex dynamics and large scales. This is done by incorporating additional Parent layers to further manage computational complexity in such systems.

While robustness is critical, there are environments where maintaining strict constraint satisfaction is infeasible due to the nature and extent of uncertainties. To address this, a new theoretical framework called Fuzzy-Logic-based MPC (FLMPC) is developed, particularly suited for controlling multi-agent systems with imperfect environmental perception operating in unknown environments. FLMPC uses fuzzy vectors to model uncertainties, where each element — being a fuzzy variable — represents the degree to which a region exhibits properties such as “dangerous” or “certain”. Fuzzy maps are constructed by grouping fuzzy vectors. FLMPC performs fuzzy optimization to compute optimal trajectories for all agents, demonstrating superior performance and reduced computational complexity compared to state-of-the-art control methods. This efficiency is achieved by enabling the execution of computationally intensive tasks of fuzzy map generation outside the real-time optimization loop. This is made possible by inheriting the fundamental strength of fuzzy logic, particularly its ability to handle uncertainties over a continuum of values, rather than discrete thresholds. This allows for reliable decision-making based on fuzzy maps within a flexible computational window, rather than being restricted to a specific time step.

The inherent limitation of finite prediction horizons in MPC poses a challenge for exploring tasks in large-scale environments. To improve scalability, a bi-level FLMPC framework — leveraging the PC-MPC architecture in the context of FLMPC — is introduced, with potential for extension to multi-level hierarchies. The Parent Fuzzy-Logic-based MPC (P-FLMPC) formulates a global plan using comprehensive environmental knowledge, while the Child Fuzzy-Logic-based MPC (C-FLMPC) focuses on local enhancement and execution of the plan, retaining flexibility for real-time adaptation.

SAMENVATTING

Model Predictive Control (MPC) is een geavanceerde regelmethode die gebruikmaakt van een model om voorspellingen te doen over de toestandsontwikkeling van het geregelde systeem, en van een optimalisatie-oplosser om een optimaal toekomstig regeltraject te berekenen dat voldoet aan toestands- en regelbeperkingen. Onnauwkeurigheden in modelvoorspellingen, veroorzaakt door modelleringsfouten of externe verstoringen, belemmeren de prestaties van MPC aanzienlijk en kunnen leiden tot schendingen van beperkingen. Om deze uitdagingen aan te pakken, houdt Robust MPC (RMPC) expliciet rekening met onzekerheden in het model, waardoor de betrouwbaarheid van de besturing wordt verbeterd. Tube-based MPC (TMPC) is een bekende RMPC-methode, waarbij een nominale MPC een referentietoestandstraject genereert, een buis (d.w.z. een traject van modelonzekerheden gecentreerd rond het referentietraject) en een aanvullende regelwet die ervoor zorgt dat de werkelijke toestanden van het systeem binnen deze buis blijven —het referentietraject volgen, ondanks onzekerheden.

Dit proefschrift presenteert een kritische analyse van de huidige state-of-the-art MPC methoden en stelt nieuwe theoretische ontwikkelingen, architecturen en uitbreidingen voor voor een effectieve en computationeel efficiënte omgang met onzekerheden via MPC . Deze bijdragen worden rigoureus ondersteund door formele bewijzen. Bovendien worden de voorgestelde besturingskaders systematisch geëvalueerd door middel van speciale computersimulaties, waarbij vergelijkingen worden gemaakt met bestaande methoden in termen van optimaliteit, robuustheid en computationele complexiteit.

Het proefschrift begint met de introductie van State-Dependent Dynamic Tube-based MPC (SDD-TMPC), een uitbreiding van TMPC die is ontworpen om de variabiliteit van modelonzekerheden en omgevingsverstoringen effectiever te behandelen. Door gebruik te maken van beschikbare informatie over toestandsafhankelijke onzekerheden, verbetert SDD-TMPC de optimaliteit en vermindert het de risico's van onhaalbaarheid, terwijl het hetzelfde niveau van robuustheid behoudt als TMPC. Hoewel SDD-TMPC toepasbaar is op systemen met variërende onzekerheden in de toestandsruimte, wordt de praktische implementatie ervan beperkt door hoge computationele eisen.

Om deze beperking te verminderen, is Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC) ontwikkeld. Deze benadering maakt gebruik van een Spiking Neural Network (SNN) om het gedrag van SDD-TMPC te benaderen. SNNs werden gekozen vanwege de gebeurtenisgestuurde verwerking en biologisch geïnspireerde efficiëntie, wat aanzienlijke voordelen biedt voor energiezuinige, realtime besturing. Omdat we ons ervan bewust zijn dat deze

benadering extra onzekerheden met zich meebrengt, en daarmee het risico van onvoldoende robuustheid ten opzichte van onzekerheden, is het SDD-TMPC-raamwerk uitgebreid om deze benaderingsfouten op te nemen als extra toestandsafhankelijke verstoringen, waardoor de robuustheid behouden blijft. De verminderde rekenvereisten van spiking neurale netwerken maken implementatie mogelijk op platforms met beperkte middelen, zoals kleinschalige robotplatforms.

Vervolgens wordt de Parent-Child MPC (PC-MPC) architectuur voorgesteld om de rekencomplexiteit in een breed scala aan MPC frameworks verder te verminderen. De PC-MPC-architectuur is compatibel met zowel tube-gebaseerde als deterministische MPC en splitst het optimalisatieprobleem op in twee gekoppelde problemen: de Parent MPC (P-MPC) richt zich op langetermijnstabiliteit en het voldoen aan beperkingen, terwijl de Child MPC (C-MPC) zich richt op kortetermijnstabiliteit en het afweren van verstoringen. P-MPC communiceert aanvullende beperkingen aan C-MPC, dat controlestrategieën bepaalt en uitvoert. Deze hiërarchische benadering, die robuustheid en stabiliteit garandeert, is uitbreidbaar naar systemen met complexe dynamica en grote schaalgroottes. Dit wordt gedaan door aanvullende Parent-lagen toe te voegen om de computationele complexiteit in dergelijke systemen verder te beheren.

Hoewel robuustheid van cruciaal belang is, zijn er omgevingen waar het handhaven van strikte beperkingsvoldoening onhaalbaar is vanwege de aard en omvang van onzekerheden. Om dit aan te pakken, is een nieuw theoretisch kader ontwikkeld, Fuzzy-Logic-based MPC (FLMPC) genaamd, dat bijzonder geschikt is voor het besturen van multi-agent systemen met onvolmaakte omgevingsperceptie die in onbekende omgevingen opereren. FLMPC gebruikt fuzzy vectoren om onzekerheden te modelleren, waarbij elk element — een fuzzy variabele — de mate weergeeft waarin een regio eigenschappen vertoont zoals “gevaarlijk’‘ of “zeker”. Fuzzy-kaarten worden geconstrueerd door fuzzy-vectoren te groeperen. FLMPC voert fuzzy-optimalisatie uit om optimale trajecten voor alle agents te berekenen, wat superieure prestaties en verminderde computationele complexiteit oplevert in vergelijking met de meest geavanceerde besturingsmethoden. Deze efficiëntie wordt bereikt door de rekenintensieve taken van het genereren van fuzzy kaarten buiten de realtime optimalisatielus uit te voeren. Dit wordt mogelijk gemaakt door de fundamentele kracht van fuzzy logica te benutten, met name het vermogen om onzekerheden over een continuüm van waarden te verwerken, in plaats van discrete drempels. Dit maakt betrouwbare besluitvorming mogelijk op basis van fuzzy kaarten binnen een flexibel rekenvenster, in plaats van beperkt te zijn tot een specifieke tijdstap.

De inherente beperking van eindige voorspellingshorizonnen in MPC vormt een uitdaging voor verkenningsopdrachten in grootschalige omgevingen. Om de schaalbaarheid te verbeteren, wordt een bi-level FLMPC-raamwerk geïntroduceerd — dat gebruikmaakt van de PC-MPC-architectuur in de context van FLMPC — met mogelijkheden voor uitbreiding naar hiërarchieën met meerdere niveaus. De Parent Fuzzy-Logic-based MPC (P-FLMPC) formuleert een globaal plan op basis van uitgebreide kennis van de omgeving, terwijl de Child Fuzzy-Logic-based MPC (C-FLMPC) zich richt op lokale verbetering en uitvoering van het plan, met behoud van flexibiliteit voor realtime aanpassing.

1

INTRODUCTION

This chapter outlines the motivation and context for this PhD thesis on advancing Model Predictive Control (MPC) under uncertainty, a problem that arises at various levels in modern applications. It highlights the practical and theoretical challenges involved in balancing performance, robustness, and computational efficiency when MPC should operate in uncertain environments. Key references, including recent surveys, are briefly discussed to position the contributions of this PhD thesis within the broader literature. The chapter concludes by summarizing the main contributions and by providing an overview of the structure of the thesis.

1.1. MOTIVATION

In the modern age, we interact daily with advanced systems such as cars, elevators, and smartphones. Despite their diverse functions, these devices all rely on control systems to operate reliably, safely, and optimally. Presently, robots are becoming an increasingly significant component of society. However, for robots to function effectively in a world designed for humans (e.g., hospitals and disaster zones), numerous challenges must be addressed. While specialized robots have demonstrated remarkable capabilities in specific contexts, such as robotic arms in manufacturing lines or autonomous vehicles in mapped outdoor areas, the adaptation of robotic systems to operate effectively in human-oriented spaces remains a significant challenge.

Model Predictive Control (MPC) is a powerful control method that optimizes the performance of a system, while respecting safety constraints. Traditionally, MPC assumes a deterministic environment and relies on linear models to predict future outcomes of the controlled system and to optimize control decisions within safety constraints. However, as systems grow more complex, such simplifying assumptions no longer hold.

A multitude of optimization-based controllers have been developed for robotic tasks, including dynamic motion planning for the Atlas robot [1] and Mars landing [2]. It is crucial

for robotics, where safety is critical, to explicitly account for the constraints of the problem, as MPC can do. Furthermore, the complex dynamics of the robots and the complexity of the mission's objective may be incorporated into the model and the cost function, respectively. Given the variety of robots and tasks in existence, a range of MPC controllers with differing levels of complexity has been put forth in the extant literature.

One of the biggest challenges in applying MPC in modern settings is uncertainty. This may raise from imperfect (e.g., linearized) models, noisy sensors, external disturbances, or incomplete information about the environment. When uncertainty is not properly accounted for, the control system may produce decisions that result in poor performance or may lead to unsafe situations. This issue assumes particular significance in the context of human-robot interactions. While the potential benefits of such collaboration are evident, any associated risks would be unacceptable. A prime example is multi-robot Search-and-Rescue (SaR) operations [3], where handling uncertainties raised from imperfect sensing and unknown surroundings is critical to maintaining both an optimal performance and safety.

At the same time, addressing uncertainty, especially through an MPC framework, often increases computational demands. The more complex models involved and robust formulations require greater processing power, and may render the problem infeasible in real time. This emerges a new, crucial scientific challenge, i.e., determining a balanced trade-off between achieving desirable performance, maintaining robustness against uncertainties, and keeping the computations efficient.

This thesis tackles this challenge by developing new MPC-based approaches for uncertain systems, with the primary goal of balancing performance, robustness, and computational efficiency to enable effective use of MPC in complex, uncertain environments.

1.2. BACKGROUND

Control theory focuses on developing frameworks and algorithms that guide systems — ranging from electrical circuits to autonomous robots — towards desired behaviors [4, 5]. Each system exhibits unique dynamics, which should be considered when designing a controller.

Controllers typically operate in a closed-loop manner (see Figure 1.1), where control inputs are generated based on sensed information from the controlled system. Systems are generally influenced by their operating environment. In this thesis, any factor affecting the system other than the controller itself is referred to as a disturbance. Such environmental disturbances are commonly random and unpredictable.

As this thesis builds upon extensions to MPC, the remainder of this chapter provides relevant background discussions and identifies key scientific challenges currently faced by MPC. We also include a brief discussion about Fuzzy Logic (FL), a powerful mathematical framework for handling uncertainties and incomplete information on systems. Each of the following chapters addresses specific aspects of the identified challenges and includes a

detailed discussion covering the most recent state-of-the-art research relevant to its topic.

MODEL PREDICTIVE CONTROL (MPC)

Within the context of MPC, finding optimal control inputs requires a prediction model, a cost (also called objective) function, properly formulated constraints, and an optimization solver. The goal is to control the system optimally by minimizing the cumulative cost — which evaluates the quality of state-action pairs — throughout the mission horizon [6]. Furthermore, MPC enables to incorporate safety constraints. Under the assumption of no disturbances or model inaccuracies, a properly designed MPC ensures that these constraints are always satisfied. This thesis, however, focuses on scenarios where such ideal assumptions do not hold — conditions that characterize the majority of real-world problems.

This PhD thesis is motivated by a central question: how to design control actions that achieve high performance, remain robust to uncertainty, and are computationally feasible for real-time implementation? Theoretically speaking, the performance of MPC is typically evaluated based on the optimality of its cost function. However, even a globally optimal MPC is of little practical use, in case it cannot be implemented efficiently in the real world. Thus, next to optimality, two other key criteria are crucial: robustness, which specifies how well the closed-loop system handles uncertainties, while ensuring satisfaction of constraints, and computational efficiency, which determines whether solutions to the optimization problem can be found in real time.

To strike a balance among these three criteria — optimality, robustness, computational efficiency — it is necessary to understand how MPC operates: MPC uses a model of the system dynamics to predict specified future outcomes based on the current state of the system and a trajectory of admissible control actions. The dynamics is formulated as a constraint that links state variables of the system at consecutive time steps under the impact of control inputs and possible disturbances. The number of future time steps predicted by MPC is called the *prediction horizon*, and the time interval between discrete time steps is the *sampling time*. Balancing these parameters is critical to optimizing the performance of MPC.

Additional safety and admissibility stage constraints on states and control inputs may also be involved within the optimization loop of MPC. Furthermore, because predictions of MPC occur within a finite time horizon, it is common to incorporate terminal costs and/or terminal constraints, which help to guarantee properties such as stability and recursive feasibility [6].

Depending on the problem type, a suitable optimization solver is selected that employs an optimization algorithm with problem-specific parameters. The solver returns an entire control input trajectory over the prediction horizon, but only the first control input is applied to the system. This approach is known as the *receding horizon strategy*. Subsequent to the execution of the control input, dynamic variables relevant for the system are sensed or estimated again. Then the horizon is rolled forward one time step, and using updated information, the

optimization problem is solved again. This process is repeated until the final goal of the control system is obtained or a system failure occurs.

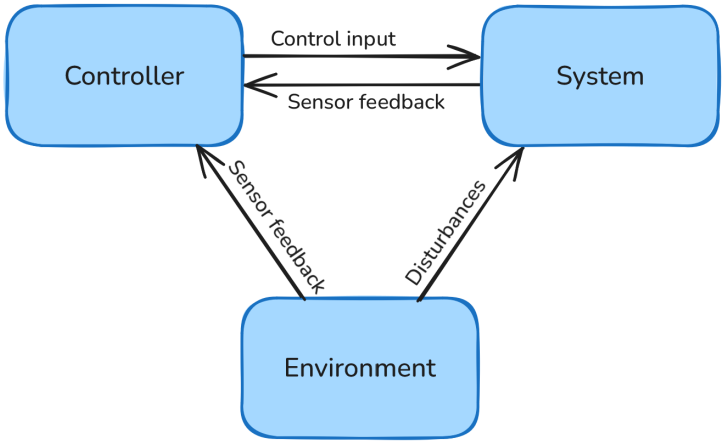


Figure 1.1: Closed-loop control system: The controller generates control inputs based on sensor feedback from the system, which operates within an environment that introduces disturbances. When relevant and feasible, certain characteristics of the environment may also be sensed and incorporated into the control loop.

LINEAR MPC

The most prevalent implementation of MPC is linear MPC, which is designed for controlling systems with (close to) linear dynamics [7]. Linear MPC typically relies on quadratic or nonlinear-convex optimization solvers, which offer significantly faster computations compared to general-purpose nonlinear solvers. Moreover, these solvers can be proven to converge to the optimal solution in a finite number of iterations.

Linear MPC is a well-established and theoretically mature control approach that has been successfully applied across many applications [8, 9]. Examples include spacecraft altitude control [10], ground vehicle steering [11], and drone trajectory tracking [12].

NONLINEAR MPC

Not all systems can be adequately described by a linear model. While it is in cases possible to apply linear MPC to nonlinear systems by, for example, linearizing them around a reference trajectory [13], such methods face inherent limitations, such as reduced accuracy when the system deviates significantly from the linearization point, limited robustness to model mismatches, and potential instability in highly nonlinear or rapidly changing dynamics. In such cases, nonlinear MPC should be deployed for control purposes [14]. In this thesis, nonlinear optimization problems are formulated mainly for two reasons: to enhance optimality even when linearization is possible (see Chapter 2 of the thesis) and to tackle inherent

complexities of a problem that cannot be captured via linear formulation of the dynamics (see Chapter 5 of the thesis).

Nonlinear MPC has been applied to a wide range of autonomous problems, including multi-agent grid world exploration [15], decision-making for simultaneous localization and mapping (SLAM) [16], obstacle avoidance [17], and global path planning [18].

MPC FOR HANDLING UNCERTAINTIES

As previously discussed, the environment often influences the behavior of the controlled system. This may introduce uncertainties and lead to imperfect predictions. The most common approach to mitigate these effects is to employ a receding horizon strategy, where the optimization problem is resolved every time step using the latest available measurements from the system. While effective in various practical scenarios, classical proofs of stability derived for MPC often rely on ideal assumptions that do not account for uncertainties [6]. In the presence of significant uncertainties, the system may become unstable and thus, dedicated versions of MPC that systematically account for uncertainties become crucial. Stochastic MPC (SMPC) [19] and Robust MPC (RMPC) [20] developed based on, respectively, stochastic optimization [21] and robust optimization [22], are such versions.

STOCHASTIC MPC (SMPC)

Stochastic optimization provides a powerful framework for handling uncertainties in control systems by probabilistic modeling of disturbances. In the context of MPC, this leads to SMPC, which incorporates uncertainties through the expected value for the cumulative cost function and by including chance constraints.

While SMPC potentially offers less conservative solutions compared to robust control methods, it also introduces significant computational and modeling challenges. Online, recursive evaluation of expected values and satisfaction of probabilistic constraints typically demand significant computational resources, especially for nonlinear or high-dimensional systems. Furthermore, propagating probability distributions for nonlinear models across the prediction horizon is particularly complex and requires approximation techniques or sampling-based methods that may not scale well in real-time applications. Moreover, developing accurate stochastic models of uncertainty is often more difficult than specifying deterministic bounds, especially in environments where uncertainty is not well characterized.

Due to these challenges, this thesis focuses on developing extensions to RMPC, which — as is explained next — provides formulations that are more tractable than SMPC in online and real-time computations, while still ensuring robustness. In particular, the approaches proposed in this PhD thesis aim to mitigate the conservatism of traditional robust formulations, without incurring the computational complexity commonly associated with SMPC.

1

ROBUST MPC (RMPC)

RMPC extends MPC by explicitly recognizing the potential discrepancy between the actual future states of the system and the ones predicted by its model — raising from external disturbances or model inaccuracies — assuming this discrepancy is bounded [20]. While RMPC shares the same control objective as nominal MPC (i.e., the same MPC problem excluding uncertainty effects), guaranteeing constraint satisfaction becomes significantly more challenging. This is because constraints should hold for all possible trajectories of states, not merely a single predicted one. Achieving robustness often requires compromising optimality. Furthermore, solving the resulting robust optimization problem can be computationally demanding.

This thesis primarily focuses on a specific variant of RMPC, known as Tube-based MPC (TMPC) [23]. This approach employs a dual-controllers structure consisting of a nominal MPC and an ancillary control law. The nominal MPC governs the nominal state — the state for an idealized version of the system that evolves according to the same dynamics as the actual system, but without any uncertainties. In effect, the nominal MPC generates a reference state trajectory that the actual system is intended to follow.

To account for uncertainties, the actual system is constrained to evolve within a bounded subset of the state space, referred to as a *tube* and surrounding the nominal state trajectory. This tube contains all admissible realizations of the actual state of the system under bounded uncertainties and steered by nominal control inputs generated by the nominal MPC. The ancillary control law ensures that, during the online operation, the actual state trajectory of the system always remains within this tube. Robustness to uncertainties is ensured by conventional TMPC through designing nominal state trajectories that position the tube within areas of the state space where all constraints are respected.

The original formulation of TMPC [23] with rigid (i.e., fixed cross section area) tubes has shown to be robust, while maintaining computational efficiency compared to deterministic MPC. However, as demonstrated in [24, 25] and also in Chapter 2 of this PhD thesis, the use of rigid tubes may lead to overly conservative solutions, potentially sacrificing optimality and, even in cases, the feasibility of the problem. The implementation of a controller characterized by overly conservative design may result in increased costs, primarily due to the inefficient use of energy resources and the extension of operation time. This, in turn, impacts the overall performance of the system and the financial implications associated with the utilization of the controller.

A limitation inherent to RMPC is the assumption of bounded uncertainties, which may not hold in all environments. In such cases, ensuring satisfaction of constraints at all times may be impractical. Additionally, robust optimization methods, such as those used in TMPC, do not account for the likelihood of different uncertain realizations. Consequently, although robust approaches are computationally more tractable, their reliance on worst-case assumptions may

come at the expense of optimality.

FUZZY LOGIC (FL) IN DECISION-MAKING

An alternative approach for handling uncertainty is offered by FL [26]. This framework is particularly effective in contexts involving vague, imprecise, or incomplete concepts, commonly illustrated through linguistic terms, such as “hot” and “cold”. While such terms are intuitively comprehensible for humans, they lack universally defined boundaries.

FL extends classical (bi-valued) logic, wherein each element belongs to or does not belong to a set. In fuzzy set theory, membership is not binary, i.e., the degree of membership of an element to a set is specified through a membership function, which assigns the element a value within the continuous interval $[0, 1]$. Each fuzzy set is associated with one such function. Conventional standard logical operators, such as union and intersection, have direct counterparts in FL that behave identically when fuzzy sets boil down to crisp ones (i.e., when the membership function assigns a unique value of zero or unity to each element).

FL has been widely adopted across various domains, including fuzzy clustering [27], natural language processing [28], and fuzzy databases [29]. Among its most common applications is in Fuzzy Inference Systems (FISs), which have been successfully applied to both control [30] and system modeling [31].

Even though FL provides a promising framework for modeling environmental uncertainty, it has not yet been widely integrated within MPC to tackle uncertainties. Existing implementations of fuzzy MPC primarily explore the use of fuzzy optimization techniques to model trade-offs between competing objectives and to enhance performance [32, 33]. However, these combinations have not yet exploited the unique potential of FL to generalize the classical formulation and operations of MPC into an FL-based framework to enable more reliable handling of uncertainties, while relaxing restrictive assumptions about the boundedness of uncertainties.

In this thesis, a particular focus is given to an additional application area, i.e., to the concept of decision-making in fuzzy environments that was introduced in [34]. These constraints are treated as soft ones, meaning that small violations of constraints do not result in outright failure of the mission, but instead incur a penalty in the cost function. Authors in [34] proposed a theoretical framework for identifying optimal solutions in such settings, providing a foundation for reasoning under imprecise or flexible constraints.

1.3. RESEARCH QUESTIONS AND MAIN CONTRIBUTIONS

This thesis addresses key research questions related to the design and implementation of MPC in environments characterized by uncertainty. It proposes both extensions to the established TMPC architecture and entirely novel control architectures that fill critical gaps in current approaches. Each chapter of this thesis contributes to answering a specific research question

through both theoretical analysis and empirical validation.

KEY RESEARCH QUESTIONS

Control approaches introduced in this thesis are evaluated through two complementary perspectives. First, their theoretical properties are analyzed in terms of stability, recursive feasibility, and optimality bounds. Second, their practical utility is assessed via empirical results concerning safety, performance, and computational efficiency. Comparisons are made against contemporary state-of-the-art methods.

A fundamental requirement for any controller is implementability in the real-world. For MPC-based controllers, especially when nonlinearity and uncertainties are involved, this often hinges on the complexity of the associated optimization problem. However, it is valuable to investigate what is theoretically achievable if this limitation was removable. This helps identifying the upper bounds of performance and robustness, provides a fundamental reference to understand possibilities in an idealized setting, and serves as a benchmark, against which the performance of practical, computationally efficient versions can be compared.

In other words, if computational limitations are set aside, the following question arises:

Research Question 1:
To what extent can the boundaries be pushed to enhance optimality and robustness in the design of MPC-based systems, assuming computational complexity is not a limiting factor?

Exploring answers to *Research Question* 1 allows us to later quantify the trade-offs between performance and implementability.

Many applications, including those to nonlinear MPC, demand reductions in computational complexity, even if it comes at the cost of optimality or robustness. This has led to a body of research that focuses on possible replacements of MPC-based controllers with look-up tables of functions, in particular Explicit MPC (EMPC) [35], or with mimicry machine-learning-based algorithms, such as Neural Networks (NNs), including Approximate MPC (AMPC) [36]. These approximation methods avoid solving the associated optimization problem online.

Control requirements of systems are distinct, exhibiting varying acceptability for compromising optimality or robustness. The required reduction in computational resources may also range from modest to substantial. These give rise to the second question that will be addressed in this thesis, i.e.:

Research Question 2:
How can the computational complexity of an MPC-based controller be reduced, depending on the extent to which requirements of the controlled system tolerate compromises in optimality and/or robustness?

As discussed in Section 1.2, the applicability of RMPC may be infeasible limited in certain environments, due to invalidity of the assumption about bounded disturbances. Meanwhile, SMPC often struggles with complex systems because of the difficulty in propagating probability distributions online. This motivates the third central research question of this thesis given below:

Research Question 3:
How can uncertainty be effectively modeled when both robust and stochastic representations are inadequate, while still maintaining a balanced trade-off between performance, safety, and computational tractability?

MAIN CONTRIBUTIONS ALIGNED WITH RESEARCH QUESTIONS

The main contributions of this PhD thesis are as follows:

- **Enhancing optimality without compromising robustness via State-Dependent Dynamic Tube-based MPC (SDD-TMPC):** A novel RMPC framework is developed in Chapter 2 to reduce conservatism and risks of infeasibility, while ensuring robustness. The controller directly addresses *Research Question* 1. It designs and incorporates a dynamic tube, for which the shape of the cross section adapts to variations in the state of the system and varying, initially unknown disturbances, which are learned as a function of the system states. Unlike conventional TMPC, disturbance bounds are not assumed to always follow the worst-case realization across the environment.

  This approach guarantees constraint satisfaction, without excessive conservatism, even in highly dynamic environments, and is adaptable to different sources of uncertainty, such as to modeling errors.

  The applicability of this framework was validated on both linear and nonlinear wheeled robot systems, as published in [37].

- **Reducing computational complexity through approximation of SDD-TMPC using Spiking Neural Networks (SNNs):** Chapter 3 investigates variants of AMPC that approximate the SDD-TMPC framework using an SNN, while ensuring safety by limiting the likelihood of constraint violations. In the literature, such safety is commonly addressed using predictive safety filters [38]. In this thesis, inspired by the work in [36],

we instead model the discrepancy between the optimal control inputs — generated by SDD-TMPC — and the outputs of the machine-learning-based approximation system as *state-dependent disturbances*. This enables the design of a safe RMPC-approximating controller that retains key advantages of SDD-TMPC, such as significantly reduced conservativeness and enhanced optimality, without involving computationally intensive online optimizations.

It is crucial to note that while increasing the bounds of potential disturbances during training the SNN can improve the likelihood of safe operation, absolute safety guarantees cannot be achieved via machine-learning-based approximators.

This contribution partially answers *Research Question* 2 by presenting a controller that achieves significant reductions in computational complexity and enhanced performance, although at some cost to robustness — though this cost can be limited and kept small in practice. This contribution, published in [39], demonstrates the first use of SNNs to approximate RMPC, enabling energy efficient implementation on resource-constrained platforms, such as micro-drones [40].

- **Reducing computational complexity by decomposing the optimization problem via the Parent-Child MPC (PC-MPC) architecture:** Chapter 4 presents the PC-MPC architecture, originally introduced in [41]. This architecture decomposes the optimization problem of any MPC or TMPC system into two coupled subproblems: a Parent MPC-based system that ensures long-term feasibility and robustness, and a Child MPC-based controller that governs short-term optimality.

  The proposed hierarchical structuring can be extended to incorporate multiple hierarchical layers, where each controller acts as Parent (Child) to the layer below (above). This multi-level PC-MPC architecture is particularly well-suited for large-scale, complex systems that span large time horizons or that operate in spatially extended environments.

  This hierarchical design addresses *Research Question* 2 by providing a scalable way to reduce computational demand and to improve optimality compared to conventional TMPC, without compromising robustness.

- **Addressing uncertainty without definite bounds effectively, leveraging Fuzzy Logic (FL) within the MPC framework through Fuzzy-Logic-based MPC (FLMPC)** Chapter 5, available also in [42], introduces a novel way to represent environmental uncertainty using FL. Inspired by human decision-making, whereby uncertain environments are perceived and evaluated through qualified mental concepts (e.g., relatively safe), FLMPC constructs fuzzy maps that describe regions of the environment in terms of fuzzy variables, e.g., “very dangerous”.

  Unlike conventional approaches that rely on strict bounds, i.e., robust methods, FLMPC models uncertainties through fuzzy variables and incorporates them into the MPC

framework performing fuzzy optimization to compute control inputs. This eliminates the need for computationally intensive propagation of distributions across the prediction horizon.

This approach is especially suitable when neither robust nor stochastic methods are viable. It offers a practical trade-off between computational efficiency, safety, and performance optimality. The efficacy of FLMPC was validated through a multi-robot exploration task, where it was benchmarked against an MPC controller with a stochastic cost function, as is presented in [42]. This novel approach to modeling and exploiting uncertainties within the MPC framework directly addresses *Research Question* 3.

To clarify the structure and the inter-connections between theoretical developments and practical implementations, Figure 1.2 presents an overview of the main topics of the thesis and how they link with central research questions and chapters of the thesis. The diagram distinguishes between sections focused on theoretical contributions (represented via ovals), such as novel control architectures and formal results, and those centered on implementation (represented via rectangles), which describe problem settings, applied controllers, and evaluation results. Blue arrows indicate how theoretical developments are applied in practice, while orange arrows represent direct extensions of prior work. Background shading distinguishes the specific research question that is addressed through the given topics.

1.4. OUTLINE OF THE THESIS

The PhD thesis is structured as follows: Chapters 2–5 each focus on a specific contribution, introducing a novel control framework or architecture. They begin with a motivation and a review of related work. Due to the diversity of mathematical formulations across chapters, each chapter adopts its own notation, which is explained per chapter. Each chapter concludes by synthesizing the achieved results and highlighting limitations, as well as potential solutions to be further researched in the future. The intermediate sections that differ by chapter are organized as follows:

- Chapter 2 focuses on SDD-TMPC, with the following structure:
  - Section 2.3.1 explains what state-dependent disturbances are.
  - Section 2.3.2 explains the principles and functioning of SDD-TMPC.
  - Section 2.3.3 describes how to model environmental uncertainty using FL. The results of such modeling are presented in Section 2.5.1.
  - Section 2.3.4 provides theoretical proofs supporting stability of SDD-TMPC.
  - Sections 2.4.1 and 2.5.2 present the first case study, where SDD-TMPC is applied to a linear system influenced by environmental disturbances. The results are compared to those from standard linear MPC and linear TMPC.

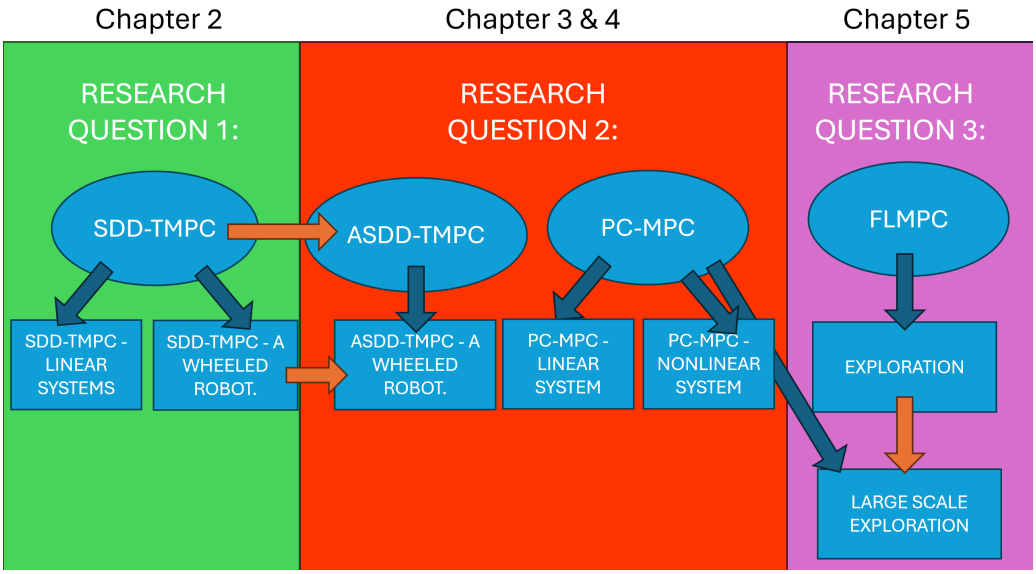


Figure 1.2: Overview of the thesis structure: Ovals represent theoretical developments, while rectangles illustrate implementation and case studies. Blue arrows show how theoretical frameworks link to case studies. Orange arrows denote direct extensions of preceding work. Background shading indicates the research questions addressed by each chapter.

  - Sections 2.4.2 and 2.5.3 describe the second case study, where SDD-TMPC controls a ground robot with a nonlinear dynamic model. Model inaccuracies are treated as state-dependent disturbances. The results are compared to those from TMPC specifically designed for this robot [43].

- Chapter 3 focuses on Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC), with the following structure:
  - Section 3.2.1 describes the types of problems, for which ASDD-TMPC is applicable.
  - Section 3.2.2 outlines the necessary modifications to SDD-TMPC before using it, in order to generate training data.
  - Section 3.2.3 details the SNN architecture and choices of hyper-parameters.
  - Section 3.2.4 discusses how the discrepancy between outputs of the SNN and the SDD-TMPC framework can be modeled as state-dependent disturbances, and how to estimate the likelihood of constraint violations.
  - Section 3.3 presents the results of training the SNN and compares the behavior of two ground robots controlled by SDD-TMPC and ASDD-TMPC, respectively.

- Chapter 4 focuses on the PC-MPC architecture, with the following structure:
  - Section 4.2 discussed the types of problems, for which the PC-MPC architecture is applicable.
  - Sections 4.3.1 and 4.3.2 explain the design of Parent MPC (P-MPC) and Child MPC (C-MPC), respectively.
  - Section 4.3.3 provides proofs of stability and recursive feasibility for the PC-MPC architecture.
  - Section 4.3.4 shows how to design the PC-MPC architecture for special cases, such as for deterministic systems.
  - Sections 4.4.1 and 4.4.2 report on both performance — with respect to the cost function — and computational efficiency, when applying the PC-MPC architecture to linear and nonlinear problems, respectively.
- Chapter 5 focuses on FLMPC, with the following structure:
  - Section 5.3.1 discusses problems domains suitable for FLMPC.
  - Section 5.3.2 describes how FLMPC operates.
  - Section 5.3.3 outlines the implementation of FLMPC for a multi-robot exploration problem.
  - Section 5.3.4 shows how to integrate FLMPC with the PC-MPC architecture.
  - Section 5.4.1 compares the performance and computation time of FLMPC with a state-of-the-art MPC that uses a stochastic cost function [44], in a multi-robot uncertain setting.
  - Section 5.4.2 evaluates the performance and computational cost of FLMPC, both with and without the PC-MPC architecture in a large-scale multi-robot exploration scenario.
- Finally, Chapter 6 summarizes the results, compares the proposed controllers, outlines suitable application scenarios, and discusses directions for future research.

REFERENCES

[1] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Autonomous Robots* 40 (2016). DOI: 10.1007/s10514-015-9479-3.

[2] L. Blackmore and D. Scharf. "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization". In: *Journal of Guidance Control and Dynamics* 33 (2010). DOI: 10.2514/1.47202.

[3] W. Chen, W. Chi, S. Ji, H. Ye, J. Liu, Y. Jia, J. Yu, and J. Cheng. "A survey of autonomous robots and multi-robot navigation: Perception, planning and collaboration". In: *Biomimetic Intelligence and Robotics* 5 (2025). DOI: 10.1016/j.birob.2024.100203.

[4] S. R. Das, P. K. Ray, A. K. Sahoo, S. Ramasubbareddy, T. S. Babu, N. M. Kumar, R. M. Elavarasan, and L. Mihet-Popa. "A Comprehensive Survey on Different Control Strategies and Applications of Active Power Filters for Power Quality Improvement". In: *Energies* 14 (2021). DOI: 10.3390/en14154589.

[5] M. Suomalainen, Y. Karayiannidis, and V. Kyrki. "A survey of robot manipulation in contact". In: *Robotics and Autonomous Systems* 156 (2022). DOI: 10.1016/j.robot.2022.104224.

[6] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, 2017. ISBN: 9780975937754.

[7] K. R. Muske and J. B. Rawlings. "Model predictive control with linear models". In: *AIChE Journal* 39 (1993). DOI: 10.1002/aic.690390208.

[8] M. Morari and J. H. Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23 (1999). DOI: 10.1016/S0098-1354(98)00301-9.

[9] D. Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50 (2014). DOI: 10.1016/j.automatica.2014.10.128.

[10] Ø. Hegrenæs, J. T. Gravdahl, and P. Tøndel. "Spacecraft attitude control using explicit model predictive control". In: *Automatica* 41 (2005). DOI: 10.1016/j.automatica.2005.06.015.

[11] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. "Predictive active steering control for autonomous vehicle systems". In: *IEEE Transactions on control systems technology* 15 (2007). DOI: 10.1109/TCST.2007.894653.

[12] A. Eskandarpour and I. Sharf. "A constrained error-based MPC for path following of quadrotor with stability analysis". In: *Nonlinear Dynamics* 99 (2020). DOI: 10.1007/s11071-019-04859-0.

[13] W. F. Lages and J. A. Vasconcelos Alves. “REAL-TIME CONTROL OF A MOBILE ROBOT USING LINEARIZED MODEL PREDICTIVE CONTROL”. In: *IFAC Proceedings Volumes* 39 (2006). DOI: 10.3182/20060912-3-DE-2911.00166.

[14] D. Mayne. “Nonlinear Model Predictive Control:Challenges and Opportunities”. In: *Nonlinear Model Predictive Control*. Birkhäuser Basel, 2000, pp. 23–44. DOI: 10.1007/978-3-0348-8407-5_2.

[15] P. Trodden and A. Richards. “Multi-Vehicle Cooperative Search Using Distributed Model Predictive Control”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit* (2008). DOI: 10.2514/6.2008-7138.

[16] C. Leung, S. Huang, and G. Dissanayake. “Active SLAM using Model Predictive Control and Attractor based Exploration”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE / IEEE Robotics and Automation Society, 2006, pp. 5026–5031. DOI: 10.1109/IROS.2006.282530.

[17] B. T. Lopez, J.-J. E. Slotine, and J. P. How. “Dynamic Tube MPC for Nonlinear Systems”. In: *2019 American Control Conference*. IEEE, 2019, pp. 2534–2549. DOI: 10.23919/ACC.2019.8814758.

[18] P. Hang, S. Huang, X. Chen, and K. K. Tan. “Path planning of collision avoidance for unmanned ground vehicles: A nonlinear model predictive control approach”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 235 (2021). DOI: 10.1177/0959651820937844.

[19] A. Mesbah. “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research”. In: *IEEE Control Systems Magazine* 36 (2016). DOI: 10.1109/MCS.2016.2602087.

[20] A. Bemporad and M. Morari. “Robust model predictive control: A survey”. In: *Robustness in identification and control*. Springer London, 1999, pp. 207–226. DOI: 10.1007/BFb0109870.

[21] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. 2nd. Springer Publishing Company, Incorporated, 2011. ISBN: 1461402360.

[22] P. De. “Robust and Adaptive Optimization”. PhD thesis. 2020. ISBN: 9798569988891.

[23] S. Raković and D. Mayne. “A SIMPLE TUBE CONTROLLER FOR EFFICIENT ROBUST MODEL PREDICTIVE CONTROL OF CONSTRAINED LINEAR DISCRETE TIME SYSTEMS SUBJECT TO BOUNDED DISTURBANCES”. In: *IFAC Proceedings Volumes* 38 (2005). DOI: 10.3182/20050703-6-CZ-1902.00440.

[24] S. V. Raković and Q. Cheng. “Homothetic tube MPC for constrained linear difference inclusions”. In: *2013 25th Chinese Control and Decision Conference (CCDC)*. IEEE, 2013, pp. 754–761. DOI: 10.1109/CCDC.2013.6561023.

[25] S. V. Raković, W. S. Levine, and B. Açikmese. “Elastic tube model predictive control”. In: *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 3594–3599. DOI: 10.1109/ACC.2016.7525471.

[26] L. Zadeh. “Fuzzy sets”. In: *Information and Control* 8 (1965). DOI: 10.1016/S0019-9958(65)90241-X.

[27] M.-S. Yang. “A survey of fuzzy clustering”. In: *Mathematical and Computer Modelling* 18 (1993). DOI: 10.1016/0895-7177(93)90202-A.

[28] J. Sun, F. Karray, O. Basir, and M. Kamel. “Natural language understanding through fuzzy logic inference and its application to speech recognition”. In: *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE’02.* IEEE, 2002, pp. 1120–1125. DOI: 10.1109/FUZZ.2002.1006661.

[29] K. Min, H. Jananthan, and J. Kepner. “Fuzzy Relational Databases via Associative Arrays”. In: *2023 IEEE MIT Undergraduate Research Technology Conference (URTC).* IEEE, 2023, pp. 1–5. DOI: 10.1109/URTC60662.2023.10534916.

[30] E. Mamdani and S. Assilian. “An experiment in linguistic synthesis with a fuzzy logic controller”. In: *International Journal of Man-Machine Studies* (1975), pp. 1–13. DOI: 10.1016/S0020-7373(75)80002-2.

[31] T. Takagi and M. Sugeno. “Fuzzy identification of systems and its applications to modeling and control”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15 (1985). DOI: 10.1109/TSMC.1985.6313399.

[32] J. da Costa Sousa and U. Kaymak. “Model predictive control using fuzzy decision functions”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31 (2001). DOI: 10.1109/3477.907564.

[33] H.-J. Zimmermann. “Fuzzy programming and linear programming with several objective functions”. In: *Fuzzy Sets and Systems* 1 (1978). DOI: 10.1016/0165-0114(78)90031-3.

[34] R. E. Bellman and L. A. Zadeh. “Decision-Making in a Fuzzy Environment”. In: *Management Science* 17 (1970). DOI: 10.1287/mnsc.17.4.B141.

[35] A. Grancharova and T. Johansen. *Explicit Nonlinear Model Predictive Control: Theory and Applications*. Vol. 429. Springer Berlin, 2012. ISBN: 978-3-642-28779-4.

[36] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. “Learning an Approximate Model Predictive Controller With Guarantees”. In: *IEEE Control Systems Letters* 2 (2018). DOI: 10.1109/LCSYS.2018.2843682.

[37] F. Surma and A. Jamshidnejad. “State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances”. In: *International Journal of Robust and Nonlinear Control* 35 (2025). DOI: 10.1002/rnc.7558.

[38] K. P. Wabersich and M. N. Zeilinger. “A predictive safety filter for learning-based control of constrained nonlinear dynamical systems”. In: *Automatica* 129 (2021). DOI: 10.1016/j.automatica.2021.109597.

[39] F. Surma and A. Jamshidnejad. “Approximate SDD-TMPC with Spiking Neural Networks: An Application to Wheeled Robots”. In: *IFAC-PapersOnLine* 58 (2024). DOI: 10.1016/j.ifacol.2024.09.050.

[40] F. Paredes-Vallés, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. C. H. E. de Croon. “Fully neuromorphic vision and control for autonomous drone flight”. In: *Science Robotics* 9 (2024). DOI: 10.1126/scirobotics.adi0591.

[41] F. Surma and A. Jamshidnejad. “Recursive feasibility without terminal constraints via parent–child MPC architecture”. In: *Results in Control and Optimization* 22 (2026). DOI: https://doi.org/10.1016/j.rico.2025.100653.

[42] F. Surma and A. Jamshidnejad. “Fuzzy-logic-based model predictive control: A paradigm integrating optimal and common-sense decision making”. In: *Applied Soft Computing* 193 (2026). DOI: https://doi.org/10.1016/j.asoc.2026.114817.

[43] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. “Robust MPC for tracking constrained unicycle robots with additive disturbances”. In: *Automatica* 90 (2018). DOI: 10.1016/j.automatica.2017.12.048.

[44] Z. Wang, J. Guo, W. Zou, and S. Li. “Cooperative search for moving targets with the ability to perceive and evade using multiple UAVs”. In: *Intelligence & Robotics* 3 (2023). DOI: 10.20517/ir.2023.30.

2

STATE-DEPENDENT DYNAMIC TUBE MPC

A NOVEL TUBE MPC METHOD WITH A FUZZY MODEL OF DISTURBANCES[1]

Most real-world systems are affected by external disturbances, which may be impossible or costly to measure. For instance, when autonomous robots move in dusty environments, the perception of their sensors is disturbed. Moreover, uneven terrains can cause ground robots to deviate from their planned trajectories. Thus, learning the external disturbances and incorporating this knowledge into future predictions in decision-making can significantly contribute to improved performance. The core idea is to learn the external disturbances that vary with the states of the system, and to incorporate this knowledge into a novel formulation for Tube-based MPC (TMPC). TMPC provides robustness to bounded disturbances, considering the known (fixed) upper bound of the disturbances; however, it does not account for the dynamics of the disturbances. This can lead to highly conservative solutions. A new dynamic version of TMPC (with proven robust stability) is proposed, called State-Dependent Dynamic Tube-based MPC (SDD-TMPC), which incorporates the dynamics of the disturbances into the decision-making of TMPC. To learn the dynamics of disturbances as a function of the system states, a fuzzy model is proposed. The performance of Model Predictive Control (MPC), TMPC, and SDD-TMPC is compared via simulations, in designed Search-and-Rescue (SaR) scenarios. The results show that, while remaining robust to bounded external disturbances, SDD-TMPC generates less conservative solutions and remains feasible in more cases, compared to TMPC.

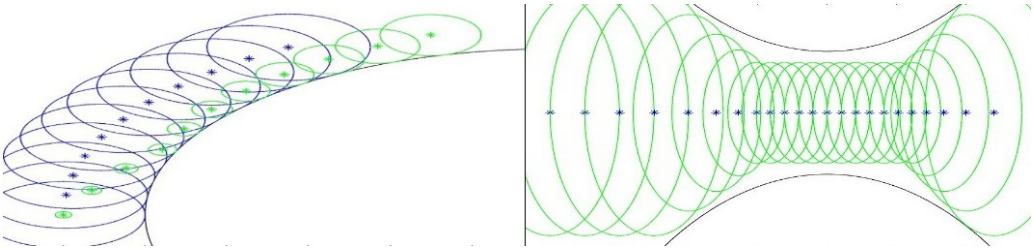


Figure 2.1: Comparison of SDD-TMPC and TMPC, where the sequences of the circular areas in both plots illustrate the evolution of the tube-based on a prediction at the current time step. **Left plot:** The tube in SDD-TMPC (shown via the smaller varying green circles) is dynamic, and always a subset of the tube of regular TMPC (shown via the larger blue circles). Thus, SDD-TMPC allows the system to perform closer to the boundaries of hard constraints, and to potentially improve the performance, without violating these constraints. **Right plot:** The nominal trajectory (shown via the blue stars) and the tube of SDD-TMPC (shown via the green varying circles) are illustrated when the states are constrained to remain within a set shown by the black boundary curves. SDD-TMPC allows, e.g., by reducing the speed of a robot, to formulate a feasible problem and provide solutions for it, and to safely move through the narrow corridor.

2.1. INTRODUCTION

MPC [2, 3] is a state-of-the-art control approach that can optimize multiple control objectives, handle state and input constraints, and provide guarantees on stability and feasibility. MPC heavily relies on a model that predicts the evolution of the states of the controlled system across a given prediction horizon. Thus, unmodeled external disturbances may lead to dangerous situations in real-life applications. For instance, SaR robots should operate autonomously in unknown environments that are prone to external disturbances [4, 5]. Therefore, for mission planning, SaR robots need control methods that next to optimize the objectives of the SaR mission (e.g., maximizing the coverage of the area in the smallest possible time) and satisfy the constraints, Also, deal with external disturbances that pose risks to the mission. Thus, MPC methods that can systematically handle the disturbances are promising for SaR robots.

In this Chapter, there is a focus on TMPC [6], which provides robustness to bounded disturbances, without significantly increasing the online computation time. TMPC generates control inputs that are composed of two parts: a nominal and an ancillary control input. The nominal states and control inputs of the system are determined by solving the nominal version of the MPC optimization (obtained by excluding external disturbances). The nominal states determine the centroids of a tube (i.e., a set of possible future states given chosen inputs and bounded disturbances) that propagates across the prediction horizon. The cross sections of

[1]This chapter is based on the publication [1], available at https://onlinelibrary.wiley.com/doi/full/10.1002/rnc.7558

the tube lie within the state space of the system, and as long as the realized states (i.e., the states in the presence of the external disturbances) at the corresponding time step remain in these cross sections, robustness is guaranteed. The ancillary controller ensures that the realized states remain inside the tube.

The nominal MPC in TMPC is not aware of the dynamics of the external disturbances. Thus, while the simplicity of the nominal optimization problem leads to a lower computation time, this may be at the cost of compromising the performance and generating overly conservative solutions, especially for systems with nonlinear dynamics and prone to dynamic disturbances. Moreover, existing MPC methods, including TMPC, do not naturally allow for the incorporation of expert knowledge. This gap should be addressed, particularly when extensive expert knowledge is available, but humans cannot or should not participate in real-time control procedures, for instance, for autonomous onboard control of SaR robots.

The main contributions of this Chapter are:

1. SDD-TMPC is introduced. The controller leverages classical TMPC to incorporate a flexible tube that is adaptively optimized online.
    - Unlike TMPC, which designs a static tube offline based on the maximum expected disturbances, SDD-TMPC allows both the size and the center of the tube to be determined by the optimization problem, taking into account the dynamics of the external disturbances as a function of the system's states. The resulting dynamic tube for SDD-TMPC is always a subset of the static tube of its counterpart TMPC.
    - Unlike TMPC, where the ancillary control inputs are generated and included outside and independent of the optimization loop, the nominal SDD-TMPC optimization problem includes within its loop the dependence of the ancillary control inputs on the nominal states that are being determined. This alteration of the nominal SDD-TMPC optimization problem, compared to TMPC, results in completely different nominal trajectories, which, based on theoretical discussions and numerical validations of the paper, significantly reduce the conservativeness and thus risk of infeasibility of SDD-TMPC.
    - More specifically, while solving its nominal problem, SDD-TMPC ensures to steer the evolution of the states (i.e., the combination of the nominal states and errors due to the disturbances) according to online estimation of the state-dependent disturbances. Therefore, a trade-off is achieved between optimizing the estimated states and reducing the impact of the resulting disturbances. Accordingly, compared to TMPC, SDD-TMPC systematically reduces the risk of the infeasibility of the constrained optimization problem, as well as the conservatism (see Figure 2.1). This results in improved performance in terms of the realized cost, with affordable computations.

- We prove the robust stability of SDD-TMPC under standard stability assumptions.

2. A set-based approach is introduced to model the dynamics of the state errors for SDD-TMPC based on the estimated dynamics of the (bounds of the) disturbances and the dynamics of the system itself. The mapping that evolves the error sets may be modeled as an analytical function or by data-based approaches (e.g., deep neural networks or fuzzy inference systems).
   - The mathematical conditions that such a mapping should generally satisfy are discussed.
   - A mapping is proposed to learn the dynamics of the state-dependent disturbances, initialized by existing expert knowledge, and using a Fuzzy Inference System (FIS). The main motivation for using a FIS is twofold: First, FIS allows, without additional costs, to direct integration of the valuable knowledge of experts (e.g., experienced SaR staff) that is usually available as linguistic data, into the controller of a system (e.g., autonomous SaR robots). Second, while other modeling approaches, e.g., Neural Network (NN), require an extensive data set to be trained, a FIS can easily be initialized with human knowledge and then be fine-tuned online to adapt to the changing environmental conditions.
3. The performance of SDD-TMPC with MPC and TMPC is compared for both linear and nonlinear systems with state-dependent external disturbances. The simulated problems correspond to autonomous robots that navigate in unknown environments. The results confirm that SDD-TMPC outperforms both MPC and TMPC by generating less conservative control inputs that still guarantee the satisfaction of the hard constraints and by resulting in an overall smaller value for the objective function (considering a minimization problem).

Note that in various cases, by proper reformulation, model inaccuracies, and disturbances on control input for nonlinear systems may also be represented as state-dependent disturbances, as is illustrated in the second case study of this Chapter and Chapter 3. Thus, SDD-TMPC can be extended to deal with uncertainties due to model mismatches and approximation errors while training NN to replace MPC.

The rest of the Chapter is structured as follows. In Section 2.2, the related work is discussed and the open challenges as well as limitations of the state-of-the-art methods are identified. In Section 2.3, the proposed approaches are explained, including SDD-TMPC, and the fuzzy model of disturbances. The proof for the stability of SDD-TMPC is also provided. In Section 2.4, the performance of MPC, TMPC, and SDD-TMPC is compared via computer simulations for a ground robot. The results of the case studies are presented in Section 2.5. Finally, Section 2.6 concludes the paper and suggests topics for future research. Mathematical notations that are frequently used in this Chapter are listed in Table 2.1.

Table 2.1: Frequently used mathematical notations and their definition

Notation	Definition	Notation	Definition
k	Time step counter in the discrete-time domain	$\mathcal{K}$	A class of continuous and strictly increasing functions starting at zero
N	Prediction horizon	$\mathcal{K}_\infty$	A subclass of $\mathcal{K}$
$\boldsymbol{z}_k$	Nominal state vector	$\mathbb{W}(\boldsymbol{x}_k)$	Set of all possible disturbances given state $\boldsymbol{x}_k$ of the system at time step k
$\boldsymbol{w}(\boldsymbol{x}_k)$	Disturbance vector when the state of the system is $\boldsymbol{x}_k$	$\mathbb{Z}^{\mathrm{f}}$	Terminal set for the nominal states
$\boldsymbol{v}_k$	Nominal input vector	$\mathbb{U}$	Admissible set for the control inputs
$\boldsymbol{u}_k$	Input vector	$\mathbb{N}$	Set of natural numbers
$\tilde{\boldsymbol{z}}_k$	Sequence of the nominal state vectors across the prediction horizon (i.e., $\{\boldsymbol{z}_{k+1},\ldots,\boldsymbol{z}_{k+N}\}$) estimated at time step k	$\mathbb{R}$	Set of real numbers
$\tilde{\boldsymbol{x}}_k$	Sequence of the state vectors across the prediction horizon (i.e., $\{\boldsymbol{x}_{k+1},\ldots,\boldsymbol{x}_{k+N}\}$) estimated at time step k	$\bar{\mathbb{E}}_k$	Error set estimated for time step k, assuming an autonomous evolution of the error set (excluding the influence of the disturbances at previous time step $k-1$)
$\tilde{\boldsymbol{v}}_k$	Sequence of the nominal input vectors across the prediction horizon (i.e., $\{\boldsymbol{v}_k,\ldots,\boldsymbol{v}_{k+N-1}\}$) estimated at time step k	$\mathbb{E}_k$	Set of all possible errors between the real and the nominal state for time step k
$\tilde{\boldsymbol{u}}_k$	Sequence of the input vectors across the prediction horizon (i.e., $\{\boldsymbol{u}_k,\ldots,\boldsymbol{u}_{k+N-1}\}$) estimated at time step k	$\mathbb{T}_k^{\max}$	Tube of regular robust TMPC for prediction horizon $\{k+1,\ldots,k+N\}$ with a fixed cross section area $\mathbb{E}^{\max}$
$\pi(\cdot)$	Control policy of SDD-TMPC	$\mathbb{T}_k$	Tube of SDD-TMPC for prediction horizon $\{k+1,\ldots,k+N\}$ with a dynamic cross section area $\mathbb{E}_i$ for $i=k+1,\ldots,k+N$
$\varepsilon(\cdot)$	Autonomous error dynamic function, which determines $\bar{\mathbb{E}}_k$ based on $\mathbb{E}_{k-1}$	$\oplus$	Minkowski summation
$(\cdot)_{i\|k}$	prediction of a dynamic variable for time step i, given the measurements at time step k	$\ominus$	Minkowski difference

2.2. RELATED WORK

2.2.1. MODELING VIA FUZZY INFERENCE SYSTEM

Fuzzy Inference System (FIS) can approximate any nonlinear function, when the inputs of the function are bounded [7]. Various supervised learning algorithms, e.g., gradient descent [8],

learning from example [9], and genetic algorithm Genetic Algorithm (GA) [10] have been proposed for the FIS to learn such functions from data. One of the most common FISs used in various applications, including robotics, is Takagi-Sugeno-Kang Fuzzy Inference System (TSK-FIS). TSK-FISs can be updated online in a stable way using, e.g., Reinforcement Learning (RL) (see applications to robot navigation [11], balancing a bipedal robot [12], controlling a continuum robot [13]). To the best my knowledge, FISs have never been used to model the dynamics of external disturbances, especially for providing robustness for model-based control approaches, e.g., MPC.

2.2.2. TUBE MODEL PREDICTIVE CONTROL

One of the most attractive Robust MPC (RMPC) methods is robust Tube-based MPC (TMPC) [6], where a robust control strategy that is designed offline is used to keep the state trajectory within an invariant tube, the center-line of which is a nominal trajectory of the states that is determined online. The nominal state trajectory can be determined for the system whenever the controller has a perfect model of the system and there are no disturbances. Although the nominal states are predictable in such cases, the actual states cannot be known in advance because the external disturbances are unknown. However, since the disturbances are assumed to be bounded and thus the maximum error between the actual and the nominal states can be determined, it is possible to compute a sequence $\mathbb{T}_k^{\max}$ of N sets per time step k, where each set $\{z_{i|k}\} \oplus \mathbb{E}^{\max}$ within this sequence includes all the possible actual states for a given time step i within the prediction window $\{k+1,\ldots,k+N\}$, and N is the prediction horizon of TMPC. Then the state constraints are guaranteed to be satisfied within the prediction window, if all the states within sequence $\mathbb{T}_k^{\max}$ satisfy the constraints. To prevent the actual states from deviating significantly from the corresponding nominal states, an ancillary control law, e.g., an error state feedback controller [6], another MPC controller [14], or a sliding mode control [15], is used to reduce the error between the actual and the nominal state.

There are two common ways of designing and implementing TMPC [16]:

- The first approach involves determining a single error set (i.e., tube) for the entire prediction window based on the error dynamics and the selected ancillary control law, where this error set serves as a positive robust invariant set. In other words, if the current error belongs to this positive robust invariant set, then all future errors remain within the set. This approach benefits from low online computational complexity since the positive robust invariant set can be computed offline. Moreover, tighter constraints can be imposed on the nominal set, such that if the center of the positive robust invariant set is inside the tightened constraints, then the actual state remains in the original feasible set. However, using this approach may result in more restrictive control inputs.

- The second approach involves computing the error set per time step within the prediction window. This approach results in a sequence of regions, known as reachable sets, which

are the smallest sets of possible states for a system prone to external disturbances that guarantee that the states remain in these sets for all time steps. This approach may yield less conservative solutions for TMPC but requires additional online computations compared to the first approach.

Some variations of TMPC for nonlinear systems have been proposed. Nonlinear TMPC is much more challenging than linear TMPC, particularly because it is not trivial to choose an ancillary control law and design a tube. One way to reduce the computational complexity of nonlinear TMPC is by using parameterized TMPC, such as the approach in [17], which reduces the required online optimization into a linear programming problem.

Another alternative is to employ ellipsoids as tubes, instead of polytopic sets (see, e.g., [18]). Some examples of nonlinear TMPC can be found in [14, 15, 19–21].

The first implementation of RMPC (although not TMPC) when state-dependent disturbances exist includes [22]. In [23], the optimization problem of RMPC has been handled as a 2nd-order cone program. This approach, however, can only be used for linear systems subject to a certain group of additive disturbances (i.e., disturbances defined as the summation of an independent component from a polytope and a state, and input-dependent component bounded via a non-convex inequality). Similarly, in [24] RMPC has been discussed for a special class of state-dependent disturbances. However, SDD-TMPC is not limited to any type of nonlinearity. Authors of [15] propose a stable TMPC for an agent that should avoid obstacles in an environment with disturbances that are proportional to the square of its velocity. The proposed method, however, is restricted to a sliding-mode ancillary control law and thus can only be used for feedback-linearizable or minimum-phase systems. In [25], a dynamic tube is used and is parameterized as a sublevel set of incremental Lyapunov functions. This method is extended in [26], where chance constraints and state or input-dependent disturbances are included. This paper has the most similarities with SDD-TMPC, since it uses a dynamic tube that evolves in time, and tightens the constraints online. Using a sub-level set of incremental Lyapunov function reduces the conservativeness with a small increase in parameters and equations (and probably in computations) compared to other approaches in the literature. The parameterization, however, requires finding an incremental Lyapunov function and restricts the shape of the tube, which potentially leads to conservative decisions. In [19], a NN has been used to learn the dynamics of the tube of TMPC. The NN, however, cannot be initialized without an extensive dataset, and thus another controller should be used until a sufficient number of data is gathered. Moreover, no proof has been provided that the trajectory of the states remains inside the tube. In [20], a Gaussian process is used to learn the disturbance set. The stability of the algorithm for linear systems has been proven. However, this method requires heavy offline computations, and discretization of the state space, and may thus become intractable for systems with large state spaces.

SDD-TMPC is proven to be stable under standard assumptions. SDD-TMPC can be used

in unknown and time-varying environments, being initially provided with intuitive human knowledge about similar environments, using fuzzy rules, and afterward being updated based on newly collected real-life data from the environment. Unlike most state-of-the-art methods, SDD-TMPC is not limited to any specific class of nonlinearity and does not restrict the choice of the ancillary control law. Moreover, unlike state-of-the-art methods that assume the bounds or the probability distribution of the disturbances are known, for SDD-TMPC it is possible to learn the disturbances.

2.3. SDD-TMPC: IDEA, FORMULATION, AND STABILITY

In this Section, the problem is described and formulated, and the proposed methods for State-Dependent Dynamic Tube-based MPC (SDD-TMPC) are discussed.

2.3.1. PROBLEM STATEMENT

A dynamic system is considered such that it is described in discrete time with the state vector $\boldsymbol{x}_k$ and control input $\boldsymbol{u}_k$, and is affected by additive state-dependent external disturbances $\boldsymbol{w}(\boldsymbol{x}_k)$ at time step $k \in \mathbb{N}$, where $\boldsymbol{x}_k \in \mathbb{X} \subseteq \mathbb{R}^n$, $\boldsymbol{u}_k \in \mathbb{U} \subseteq \mathbb{R}^m$, and $\boldsymbol{w}(\boldsymbol{x}_k) \in \mathbb{W}(\boldsymbol{x}_k) \subseteq \mathbb{R}^n$. While the admissible sets $\mathbb{X}$ and $\mathbb{U}$ for, respectively, the state and the control input are static, The admissible set $\mathbb{W}(\boldsymbol{x}_k)$ of the external disturbances is allowed to vary as a function of the state. Learning (an approximation of) the admissible set $\mathbb{W}(\boldsymbol{x}_k)$, such that it bounds the external disturbances per state, instead of considering a set that bounds all potential external disturbances for the entire admissible set $\mathbb{X}$ in time, introduces dynamics and reduces conservativeness into SDD-TMPC.

The dynamic system is given by:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}(\boldsymbol{x}_k) \tag{2.1}$$

where $f(\cdot)$ is a time-invariant time-discrete nonlinear Lipschitz function.

It is assumed that the states are measured by perfect sensors, i.e., per time step k the value of the state $\boldsymbol{x}_k$ is perfectly known, whereas the external disturbances are unknown and may take any arbitrary value within $\mathbb{W}(\boldsymbol{x}_k)$. The aim is to control the dynamics of system (2.1), such that a given cost function $J(\cdot)$ is minimized across a prediction horizon of size N, while it is guaranteed for the controlled system that for all admissible external disturbances the hard constraints are always satisfied. Thus, for time step k there is:

$$\begin{aligned} J^*(\boldsymbol{x}_k) = \min_{\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{u}}_k} \; & J(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{u}}_k) \\ \text{s.t. for } & i = k, \ldots, k+N-1 \\ & \boldsymbol{x}_{i+1|k} = f(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}) + \boldsymbol{w}(\boldsymbol{x}_{i|k}) \\ & \boldsymbol{x}_{k|k} = \boldsymbol{x}_k \end{aligned} \tag{2.2}$$

$$\boldsymbol{x}_{i+1|k} \in \mathbb{X}$$
$$\boldsymbol{u}_{i|k} \in \mathbb{U}$$
$$\boldsymbol{w}(\boldsymbol{x}_{i|k}) \in \mathbb{W}(\boldsymbol{x}_{i|k})$$

with $\tilde{\boldsymbol{x}}_k = [\boldsymbol{x}_{k+1|k}^\top, \ldots, \boldsymbol{x}_{k+N|k}^\top]^\top$, $\boldsymbol{x}_{i|k}$ for $i > k$ the value of $\boldsymbol{x}_i$ predicted at time step k, $\tilde{\boldsymbol{u}}_k = [\boldsymbol{u}_{k|k}^\top, \ldots, \boldsymbol{u}_{k+N-1|k}^\top]^\top$, $\boldsymbol{u}_{i|k}$ for $i \geq k$ the value of $\boldsymbol{u}_i$ determined at time step k, and $J^*(\boldsymbol{x}_k)$ an optimal value for the cost function obtained by solving the constrained minimization problem. The cost function $J(\cdot)$ is composed of a stage cost that is accumulated across the prediction horizon $\{k, \ldots, k+N-1\}$, and a terminal cost that is computed at the terminal time step $k+N$. The cost function is continuous and finite for $\boldsymbol{x}_i \in \mathbb{X}$, $\forall i \in \{k, \ldots, k+N\}$.

2.3.2. STATE-DEPENDENT DYNAMIC TUBE MPC

TMPC works based on the assumption of bounded external disturbances, where the boundary set $\mathbb{W}^{\max}$ for the disturbances is known and given [6]. A main challenge for SDD-TMPC in solving (2.2) per time step k is that the dynamic set $\mathbb{W}(\boldsymbol{x}_i)$ (for $i = k, \ldots, k+N-1$) is not known (neither for the current time step nor for the future time steps) and should thus be estimated. The link between the known static set $\mathbb{W}^{\max}$ and the dynamic set $\mathbb{W}(\boldsymbol{x}_i)$ of disturbances that should be estimated by SDD-TMPC for every prediction horizon $\{k+1, \ldots, k+N\}$ is given by:

$$\bigcup_{i=k,\ldots,k+N-1} \mathbb{W}(\boldsymbol{x}_i) \subseteq \mathbb{W}^{\max} \tag{2.3}$$

$\mathbb{W}(\boldsymbol{x}_i)$ is assumed to always contain the origin. Regular TMPC uses $\mathbb{W}^{\max}$ to determine a sequence of N control inputs and a tube $\mathbb{T}_k^{\max}$ for the entire prediction window per time step k, for which the cross section $\mathbb{E}^{\max}$ remains unchanged. The tube $\mathbb{T}_k^{\max}$ is a robust positive invariant set. The main difference between TMPC and SDD-TMPC is in the formulation of their nominal MPC. This allows SDD-TMPC to make use of the dynamics of the external disturbances, to generate per time step k a tube $\mathbb{T}_k$ with a generally time-varying cross section $\mathbb{E}_i$ for $i = k+1, \ldots, k+N$ across the prediction horizon that improves the control performance by generating less conservative solutions. Note that the set $\mathbb{T}_k$ is a robust control invariant set, i.e., there is at least one control policy that prevents the state trajectory from leaving the tube. The nominal problem of SDD-TMPC is given by:

$$V^*(\boldsymbol{x}_k, \boldsymbol{z}_k) = \min_{\tilde{\boldsymbol{z}}_k, \tilde{\boldsymbol{v}}_k, \mathbb{T}_k} \sum_{i=k}^{k+N-1} l(\boldsymbol{z}_{i|k}, \boldsymbol{v}_{i|k}) + V_{\mathrm{f}}(\boldsymbol{z}_{k+N|k}) \tag{2.4a}$$

$$\text{s.t. for } i = k+1, \ldots, k+N\text{:}$$

$$\boldsymbol{z}_{i|k} = f(\boldsymbol{z}_{i-1|k}, \boldsymbol{v}_{i-1|k}) \tag{2.4b}$$

$$\bar{\mathbb{E}}_{i|k} = \varepsilon(\mathbb{E}_{i-1|k}) \tag{2.4c}$$

$$\boldsymbol{\Theta}_{i-1|k} = f^{\text{dis}}\left(\{\boldsymbol{z}_{i-1|k}\} \oplus \mathbb{E}_{i-1|k}\right) \tag{2.4d}$$

$$\mathbb{E}_{i|k} = \bar{\mathbb{E}}_{i|k} \oplus \bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\Theta}_{i-1|k}) \tag{2.4e}$$

$$\mathbb{E}_k = \{\boldsymbol{x}_k - \boldsymbol{z}_k\} \tag{2.4f}$$

$$\{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k} \subseteq \mathbb{X} \tag{2.4g}$$

$$\boldsymbol{z}_{N|k} \in \mathbb{Z}^{\text{f}} \tag{2.4h}$$

$$\mathbb{T}_k = \left\{\{z_{k+1|k}\} \oplus \mathbb{E}_{k+1|k}, \ldots, \{z_{N|k}\} \oplus \mathbb{E}_{k+N|k}\right\} \tag{2.4i}$$

$$\pi\left(\{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k}, \boldsymbol{v}_{i|k}, \boldsymbol{z}_{i|k}\right) \subseteq \mathbb{U} \tag{2.4j}$$

In (2.4a), $l(\cdot)$ is the stage cost, $V_{\text{f}}(\cdot)$ is the terminal cost, and $\tilde{\boldsymbol{z}}_k$ and $\tilde{\boldsymbol{v}}_k$ are the trajectory/sequence of, respectively, the nominal states and the nominal control inputs within the prediction window $\{k+1,\ldots,k+N\}$. The realized states of the system follow the nominal state sequence when there are no external disturbances. The nominal states $\boldsymbol{z}_{i|k}$ are predicted at time step k for time step $i = k+1,\ldots,k+N$ according to (2.4b). Moreover, $\oplus$ represents the Minkowski addition [27], which for every two given sets A and B is defined by:

$$\mathbb{A} \oplus \mathbb{B} = \{a+b : a \in \mathbb{A}, b \in \mathbb{B}\} \tag{2.5}$$

The error set $\mathbb{E}_{i|k}$ (for $i = k+1,\ldots,k+N$) for SDD-TMPC is computed in three steps: First, in (2.4c) the system uses the autonomous error dynamic function $\varepsilon(\cdot)$ to estimate the evolved error set, without considering the influence of the state-dependent disturbances on the errors. The autonomous error dynamic function is determined via state-of-the-art methods and based on the nominal model $f(\cdot)$ of the system dynamics and the policy $\pi(\cdot)$, assuming no external disturbances exist. Note that the error vector is defined in the prediction window by $\boldsymbol{e}_{i|k} = \boldsymbol{x}_{i|k} - \boldsymbol{z}_{i|k}$ for $i = k+1,\ldots,k+N$. For details on how $\varepsilon(\cdot)$ can be determined, see, e.g., [6], for linear TMPC and [15] for nonlinear TMPC. Second, in (2.4d) a mapping, f^{dis}, is used to determine, based on the estimated nominal states (see (2.4b)) and the error set corresponding to the previous time step, a vector of parameters $\boldsymbol{\Theta}_{i-1|k}$. This vector is then used in (2.4e) to determine an admissible set $\bar{\mathbb{W}}_{i-1|k}$ (i.e. an outer approximation of $\mathbb{W}_{i-1|k}$) per time step i ($i = k+1,\ldots,k+N$) that contains all the possible disturbances at time step $i-1$ that can result in all the possible states at time step i. From a practical point of view, both the error sets $\mathbb{E}_{k+1|k}$ and the vector of parameters $\boldsymbol{\Theta}_{i-1|k}$ are just ancillary parameters that may be computed using decision variables and constraints. Note that mathematically, f^{dis} can in general be represented by any mappings (e.g., an analytical function or a deep neural network) that properly formulate the existing knowledge about the state-dependent disturbances, as long as the mapping meets the conditions explained in Section 2.3.3. The disturbance model (i.e., f^{dis}) in (2.4c) is in general developed offline, based on the existing general knowledge that

corresponds to the expected disturbance levels with the states of the environment, thus the states of the robot. Such prior information may be available as quantified data or in linguistic terms (e.g., via expert SaR staff). Accordingly, the most suitable ways for modeling (e.g., machine learning, Fuzzy Logic (FL)-based modeling, etc.) is used. Third, the sets obtained via the previous two steps (i.e., via (2.4c) and (2.4d)) are combined in (2.4e). In other words, the set $\mathbb{E}_{i|k}$ of all possible errors for time step i is obtained by combining the autonomously evolved error set and the set of all possible disturbances from the previous time step that can result in further errors. Based on Definition 1 given next, $\mathbb{E}_{i|k}$ for $i = k+1,\ldots,k+N$ is an estimation of a robust forward reachable set for $\mathbb{E}_{i-1|k}$.

Definition 1. Consider the autonomous system that is formulated via (2.4c)-(2.4e). A one-step robust reachable set $\mathbb{R}_\varepsilon$ from the set of errors $\mathbb{E}_{i-1|k}$ that is estimated for time step $i-1$ (with $i = k+1,\ldots,k+N$) is defined based on [28], via:

$$\mathbb{R}_\varepsilon(\mathbb{E}_{i-1|k}, \bar{\mathbb{W}}_{i-1|k}) = \Big\{ \boldsymbol{e} \in \big(\mathbb{X} \ominus \{\boldsymbol{z}_{i|k}\}\big) \,|\, \exists \boldsymbol{e}_{i-1|k} \in \mathbb{E}_{i-1|k}, \exists \boldsymbol{w}(\boldsymbol{x}_{i-1|k}) \in \bar{\mathbb{W}}_{i-1|k} : \boldsymbol{e} = \varepsilon(\boldsymbol{e}_{i-1|k}) + \boldsymbol{w}(\boldsymbol{x}_{i-1|k}) \Big\}$$

The three steps explained above for estimation of the error set $\mathbb{E}_{i|k}$ (for $i = k+1,\ldots,k+N$) are illustrated in Figure 2.2. Constraint (2.4f) initializes the error set per time step when the nominal SDD-TMPC is solved. Constraint (2.4g) assures that the realized states, estimated online based on the nominal state and the error values, remain inside the admissible state set $\mathbb{X}$. Finally, (2.4h) is a terminal constraint with $\mathbb{Z}^{\mathrm{f}} \subseteq \mathbb{X}$, where $\mathbb{Z}^{\mathrm{f}}$ is a control invariant set for the nominal system that guarantees the recursive feasibility.

The dynamic tube $\mathbb{T}_k$ of SDD-TMPC for time step k is given by (2.4i). The optimization variables of (2.4) are the nominal state sequence $\tilde{\boldsymbol{z}}_k$, the nominal control input sequence $\tilde{\boldsymbol{v}}_k$, and the SDD-TMPC tube $\mathbb{T}_k$ (i.e., an ordered set including the influence of the error sets across the entire prediction horizon). By including the tube as an optimization variable for the nominal SDD-TMPC problem, the optimizer determines solutions that foresee the dynamics of the error in the generation of the online nominal state trajectory (i.e., the center-line of tube $\mathbb{T}_k$). Therefore, the nominal state trajectory of SDD-TMPC is in general different from that of regular TMPC (see Figure 2.1). Moreover, the SDD-TMPC policy $\pi : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{U}$, which receives the nominal state trajectory of SDD-TMPC as input, is incorporated within the optimization loop of SDD-TMPC via constraint (2.4j), as well as in the estimation of the autonomous error dynamic function as explained before. The policy $\pi(\cdot)$ of SDD-TMPC, which combines the nominal SDD-TMPC inputs and ancillary control inputs, may be generated such that the closed-loop system is stabilized (see, e.g., [14, 15]). Constraint (2.4j) enforces the generated control inputs to remain inside the admissible control input set $\mathbb{U}$ for all possible realizations of the state. Thus, the policy also plays a role in the computation of the online nominal state trajectory that is determined via SDD-TMPC. Note that for computing the policy in (2.4j), where the input of the function is a set, Definition 2 given next is used.

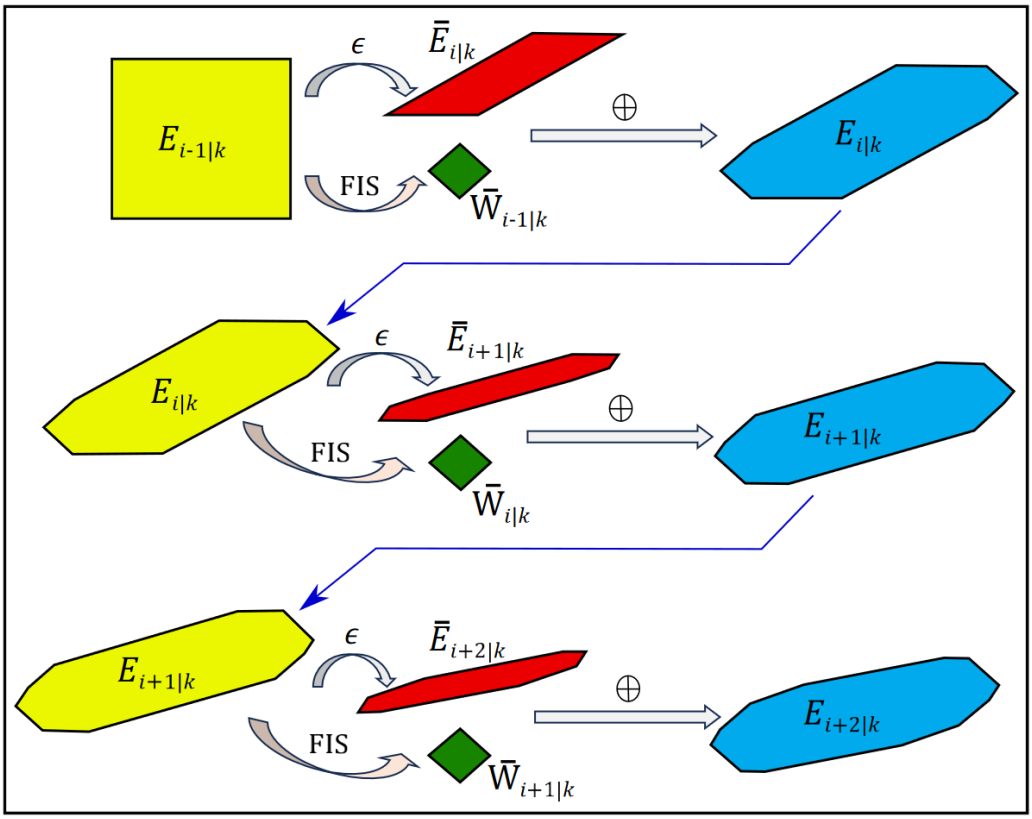


Figure 2.2: Illustration of the development of the error set $\mathbb{E}_{\cdot|k}$ for SDD-TMPC via (2.4c)-(2.4e) in 2 dimensions: Each row corresponds to one prediction time step (illustrated in this figure for time steps i, $i+1$, $i+2$). The 2-dimensional sets in the first column (shown in yellow) illustrate the error set $\mathbb{E}_{\cdot-1|k}$ that has been determined at the previous time step. By propagating this error set through the autonomous error dynamics (i.e., mapping ε) via (2.4c), the 2-dimensional sets $\bar{\mathbb{E}}_{\cdot|k}$ at the top side of the second column of each row (shown in red) are generated. Moreover, the 2-dimensional sets $\bar{\mathbb{W}}_{\cdot-1|k}$ at the bottom part of the second column in each row (shown in green) are obtained via the disturbance model (mapping $f^{\text{dis}}(\cdot)$) using (2.4d) where the input of the mapping is the error set that is centered around the nominal state trajectory. Finally, the red and green 2-dimensional sets of the second column per row are combined using Minkowski summation to obtain the error set $\mathbb{E}_{\cdot|k}$ via (2.4e). This set is then used as the starting set for the next time step (i.e., at the start of the next row).

Definition 2. Whenever a function $g(\cdot)$ takes a set $\mathbb{S}$ as input, then the output $\mathbb{O}$ is also a set defined by $\mathbb{O} := \{g(i) | i \in \mathbb{S}\}$, when $g(i)$ itself is not a set, and by $\bigcup_{i\in\mathbb{S}} g(i)$ when $g(i)$ itself is a set. Thus, if $\mathbb{S}_1 \subseteq \mathbb{S}_2$, then $g(\mathbb{S}_1) \subseteq g(\mathbb{S}_2)$.

State-of-the-art TMPC methods, however, solve the nominal MPC problem without incorporating the policy $\pi(\cdot)$. Only after determining the nominal state trajectory via an optimization procedure, the policy of TMPC is used outside of the optimization loop to generate the actual control input $\boldsymbol{u}_k$.

In summary, the inclusion of the error evolution model into the optimization procedure of SDD-TMPC, which is done via incorporating the dynamic tube of SDD-TMPC in the optimization variables, as well as including the policy of SDD-TMPC into the constraints, is expected to result in solutions for SDD-TMPC that are less conservative than the solutions of TMPC.

The optimization problem (2.4) is, in general, nonlinear and non-convex. It is common to assume convexity for the admissible sets $\mathbb{X}$ and $\mathbb{U}$ for the states and the control inputs [6, 20], but this is not the case in some applications (e.g., in collision avoidance [15, 28], which is relevant for SaR). Therefore, to solve the non-convex, nonlinear optimization problem, solvers such as GA [10], Particle Swarm (PS) [29], and Sequential Quadratic Programming (SQP) [30] may be used. In particular, to address the issues that are raised due to non-convexity, especially falling into local optima (due to the gradient-based nature of the optimizer or the limited number of iterations for the optimization), one may use multi-start optimization.

2.3.3. MODELING THE DYNAMICS OF THE EXTERNAL DISTURBANCES

The predicted and actual states are assumed to be different, and that the difference is bounded, but the bound is state-dependent. This difference can be the result of a real external force that affects the system differently depending on its state. For example, when a robot moves in an environment, the external force may vary in different parts of the environment. This difference can also be the result of an inaccurate model, and the inaccuracy can be state-dependent. For example, it is well known that when a system is linearized around a certain state, a model is very accurate around that state, but becomes less accurate as it moves away. Both of these examples are shown in the case studies.

The goal is to use a disturbance model (f^{dis}) to model the set of external disturbances per time step $i-1$, where i belongs to the prediction horizon $\{k+1,\dots,k+N\}$, as a function of the system states, i.e. to approximate $\mathbb{W}(\boldsymbol{x}_{i-1|k})$ given in (2.3) by $\bar{\mathbb{W}}_{i-1|k}$ using (2.4d). For the type of the set, an ellipsoid or a polytope is considered; it is assumed that the origin always belongs to the set. Ellipsoids and polytopes are the most commonly used sets in the related literature [6, 18] and –when parameterized– are bounded if these parameters are bounded.

Next, the set is parameterized and it is shown by $\bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\Theta}_{i-1|k})$, where $\boldsymbol{\Theta}_{i-1|k}$ is the vector of all parameters at time step $i-1$. Therefore, the problem of approximating the disturbance

set is translated into the problem of determining the vector $\boldsymbol{\Theta}_{i-1|k}$. This vector is returned by the mapping $f^{\text{dis}}(\cdot)$, which takes as input the corresponding state of the system (see (2.4d)). Mathematically, f^{dis} can, in general, be represented by any mappings (e.g., an analytical function or a deep neural network) that properly formulate the existing knowledge about the state-dependent disturbances, as long as the stability conditions are met by the mapping.

STABILITY CONDITION

To ensure robustness for SDD-TMPC to all values of the external disturbances, i.e., to ensure that (2.3) holds, the parameter vector $\boldsymbol{\Theta}_{i-1|k}$ should be determined such that the actual disturbance set $\mathbb{W}(\boldsymbol{x}_{i-1|k})$ for all time steps $i \in \{k+1,\dots,k+N\}$ is a subset of the approximate disturbance set $\bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\Theta}_{i-1|k})$, which itself should be a subset of $\bar{\mathbb{W}}^{\max}$. The formulation of the robust control problem is based on the assumption of the existence of $\mathbb{W}^{\max}$, which bounds all possible disturbances. Thus, the evolution of the disturbance set generated by the above $f^{\text{dis}}(\cdot)$ is stable if the approximate disturbance set $\bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\Theta}_{i-1|k})$ is bounded by $\mathbb{W}^{\max}$ for all $i \in \{k+1,\dots,k+N\}$.

TAKAGI-SUGENO-KANG FIS FOR SAR ROBOTICS

The rule base of the TSK-FIS of SDD-TMPC is composed of rules with the formulation given by (2.6), where $\widetilde{X}^l$ is a fuzzy set that mathematically represents a linguistic term that describes the input variable x:

$$\text{Rule } l: \text{ IF } x \text{ is } \widetilde{X}^l, \text{ THEN } y^l \text{ is } h^l(x) \tag{2.6}$$

The output generated by each rule concerns only one element of the parameter vector $\boldsymbol{\Theta}_{i-1|k}$. More specifically, for $i \in \{k+1,\dots,k+N\}$, the input x is replaced by $\boldsymbol{x}_{i-1|k}$, and the output y^l is replaced by an element $\theta^l_{i-1|k}$ of the parameter vector. Note that more than one rule in the rule base may generate a candidate value for this element of the parameter vector. Therefore, the superscript l is used to show that the value is generated for this element via the l^{th} rule within the set of all L rules that generate a candidate value for this element. Moreover, $h^l(\cdot)$ shows a generally nonlinear mapping from the input space to the output space (i.e., from the admissible set of the state variables of the dynamical system to the admissible set for element $\theta_{i-1|k}$ of the vector $\boldsymbol{\Theta}_{i-1|k}$). Then the final value for the parameter is computed by:

$$\theta_{i-1|k} = \frac{\Sigma_{l=1}^{L} \mu^l(\boldsymbol{x}_{i-1|k}) h^l(\boldsymbol{x}_{i-1|k})}{\Sigma_{l=1}^{L} \mu^l(\boldsymbol{x}_{i-1|k})} \tag{2.7}$$

where $\mu^l(\cdot)$ is the membership function of fuzzy set $\widetilde{X}^l$ and L is the number of the rules in the rule base that generate a value for element $\theta_{i-1|k}$.

The approximate set $\bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\theta}_{i-1|k})$ is bounded, whenever all elements of vector $\boldsymbol{\Theta}_{i-1|k}$ are bounded (based on the assumption of an ellipsoid or polytope set or any other set that satisfies the condition of bounded output for bounded input). The membership functions in

(2.7) are restricted to the interval $[0,1]$ by definition, implying that to keep the elements of $\boldsymbol{\Theta}_{i-1|k}$ bounded, the functions $h^l(\cdot)$ (with $l \in \{1,\cdots,L\}$) should be formulated such that for all $\boldsymbol{x}_{i-1|k} \in \mathbb{X}$ the function remains bounded. Finally, to ensure that the approximate set $\bar{\mathbb{W}}_{i-1|k}(\boldsymbol{\Theta}_{i-1|k})$ that is designed to be bounded, is also a subset of $\mathbb{W}^{\max}$, this set is defined as the union of the output of the TSK-FIS and $\mathbb{W}^{\max}$. Therefore, the proposed TSK-FIS satisfies the stability conditions.

2.3.4. STABILITY OF SDD-TMPC

In this section, the stability of SDD-TMPC is discussed, and discussions are based on the following assumptions:

1. There exists $\mathbb{T}_k^{\max} = \{\{z_{k+1|k}\} \oplus \mathbb{E}^{\max}, \ldots, \{z_{N|k}\} \oplus \mathbb{E}^{\max}\}$ that propagates across the prediction horizon at time step k and is positive robust invariant under the policy $\pi(\cdot) : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{U}$, i.e., when the system follows policy $\pi(\cdot)$, the state remains inside this set.

2. There exists a terminal control invariant set $\mathbb{Z}^{\mathrm{f}}$ for the nominal system, such that $\mathbb{Z}^{\mathrm{f}} \oplus \mathbb{E}^{\max} \subseteq \mathbb{X}$.

3. There exists a control law $\kappa : \mathbb{Z}^{\mathrm{f}} \to \mathbb{U}$ for the nominal system, such that for $\boldsymbol{z}_k \in \mathbb{Z}^{\mathrm{f}}$ and $i = k, \ldots, k+N-1$:
   - $f(\boldsymbol{z}_{i|k}, \kappa(\boldsymbol{z}_{i|k})) \in \mathbb{Z}^{\mathrm{f}}$
   - $V_{\mathrm{f}}(f(\boldsymbol{z}_{i|k}, \kappa(\boldsymbol{z}_{i|k}))) - V_{\mathrm{f}}(\boldsymbol{z}_{i|k}) \leq -l(\boldsymbol{z}_{i|k}, \kappa(\boldsymbol{z}_{i|k}))$ (where $V_{\mathrm{f}}(\cdot)$ is the terminal cost as is given in (2.4a))
   - $\pi(\boldsymbol{x}_{i|k}, \kappa(\boldsymbol{z}_{i|k}), \boldsymbol{z}_{i|k}) \in \mathbb{U}$, $\boldsymbol{x}_{i|k} \in \{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}^{\max}$

4. There exists $\mathcal{K}_\infty$ functions $\alpha_1(\cdot)$ and $\alpha_{\mathrm{f}}(\cdot)$ that satisfy the following inequalities (note that a function belongs to class $\mathcal{K}$, if it is continuous, zero at zero, and strictly increasing; and a function belongs to class $\mathcal{K}_\infty$, if it is in class K and is unbounded [2])):
   - $l(\boldsymbol{z}_{i|k}, \boldsymbol{v}_{i|k}) \geq \alpha_1(|\boldsymbol{z}_{i|k}|) \quad \forall \boldsymbol{z}_{i|k} \in \mathbb{X}, \forall \boldsymbol{v}_{i|k} \in \mathbb{U}$ (where $l(\cdot)$ is the stage cost as is given in (2.4a))
   - $V_{\mathrm{f}}(\boldsymbol{z}_{i|k}) \leq \alpha_{\mathrm{f}}(|\boldsymbol{z}_{i|k}|) \quad \forall \boldsymbol{z}_{i|k} \in \mathbb{Z}^{\mathrm{f}}$

Assumption 1 implies that a stabilizable TMPC law can be determined for the system (i.e., the error does not go to infinity). Moreover, $\mathbb{T}_k^{\max}$ is the tube with a constant cross section that is used in TMPC. Assumption 2 (existence of a nominal invariant set) is a standard assumption in TMPC literature [6]. The first two items of Assumption 3 and Assumption 4 are standard assumptions in MPC literature [2] that are required to use the optimal cost as a Lyapunov function. The third item of Assumption 3 indicates that there exists a control law in the terminal set, such that the input constraints are satisfied.

First, some theorems that are used in the discussions on stability are given.

Theorem 1. If $\mathbb{C} \subseteq \mathbb{A}$, then $\mathbb{C} \oplus \mathbb{B} \subseteq \mathbb{A} \oplus \mathbb{B}$.

Proof. Each element in $\mathbb{C} \oplus \mathbb{B}$ is given by $c+b$, where $c \in \mathbb{C}$ and $b \in \mathbb{B}$. Since $\mathbb{C} \subseteq \mathbb{A}$, then $c \in \mathbb{A}$. Thus, from the definition of Minkowski addition (2.5), there is $(c+b) \in \mathbb{A} \oplus \mathbb{B}$. □

Theorem 2. If $\mathbb{C} \subseteq \mathbb{A}$ and $\mathbb{D} \subseteq \mathbb{B}$, then $\mathbb{C} \oplus \mathbb{D} \subseteq \mathbb{A} \oplus \mathbb{B}$.

Proof. From the previous theorem, there is $\mathbb{C} \oplus \mathbb{B} \subseteq \mathbb{A} \oplus \mathbb{B}$ and $\mathbb{C} \oplus \mathbb{D} \subseteq \mathbb{C} \oplus \mathbb{B}$, which together imply that $\mathbb{C} \oplus \mathbb{D} \subseteq \mathbb{A} \oplus \mathbb{B}$. □

Theorem 3. If $\mathbb{E}_k \subseteq \mathbb{E}^{\max}$ and the system follows the SDD-TMPC policy $\pi(\cdot)$, then $\mathbb{E}_{i|k} \subseteq \mathbb{E}^{\max}$ for $i = k+1, \ldots, k+N$, where $\{z_{i|k}\} \oplus \mathbb{E}_{i|k}$ are the elements of the tube $\mathbb{T}_k$ of SDD-TMPC.

Proof. By contradiction, if the theorem is not true, i.e., if there exists $\mathbb{E}_{i|k}$ for $i = k+1, \ldots, k+N$, such that $\mathbb{E}_{i|k} \not\subseteq \mathbb{E}^{\max}$ then there is at least one element $\boldsymbol{e}_{i|k} = \boldsymbol{x}_{i|k} - \boldsymbol{z}_{i|k}$ corresponding to a possible realization $\boldsymbol{x}_{i|k}$ of the state at time step i, such that $\boldsymbol{e}_{i|k} \in \mathbb{E}_{i|k}$, but $\boldsymbol{e}_{i|k} \notin \mathbb{E}_{\max,i|k}$. Therefore, for the corresponding external disturbances, the state leaves tube $\mathbb{T}^{\max}$, which is not possible because $\mathbb{T}^{\max}$ is a robust positive invariant set for the policy $\pi(\cdot)$ according to Assumption 1. □

Theorem 4. The set $\Phi := \{\{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k} \big| \boldsymbol{z}_{i|k} \in \mathbb{Z}^{\mathrm{f}},\ \mathbb{E}_{i|k} \subseteq \mathbb{E}^{\max},\ i = k+1, \ldots, k+N\}$ is a set robust positive invariant set for the SDD-TMPC inputs $\pi(\boldsymbol{x}_{i|k}, \kappa(\boldsymbol{z}_{i|k}), \boldsymbol{z}_{i|k})$, for $i = k, \ldots, k+N-1$. (Note that according to [6], a set robust positive invariant set is defined as a set of sets. Since the terminal real state $x_{k|N+k}$ should be bounded, I need to constrain both the nominal state and the potential errors; thus, a set of tubes (i.e., Φ) is needed.)

Proof. From Assumption 3 it follows that if $\boldsymbol{z}_k \in \mathbb{Z}^{\mathrm{f}}$, then $\boldsymbol{z}_{i|k} \in \mathbb{Z}^{\mathrm{f}}$ for $i = k+1, \ldots, k+N$. Moreover, from Theorem 3 if $\mathbb{E}_k \subseteq \mathbb{E}^{\max}$, then $\mathbb{E}_{i|k} \subseteq \mathbb{E}^{\max}$ for $i = k+1, \ldots, k+N$. Therefore, from the definition of Φ, there is $\{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k} \in \Phi$, which proves the theorem. □

Theorem 5. If the sequence of nominal control inputs $\tilde{\boldsymbol{v}}_k = [\boldsymbol{v}_{k|k}^\top, \boldsymbol{v}_{k+1|k}^\top, \ldots, \boldsymbol{v}_{k+N-1|k}^\top]^\top$ of SDD-TMPC is feasible for the admissible pair $(\boldsymbol{z}_k, \boldsymbol{x}_k)$, i.e., using these inputs and policy $\pi(\cdot)$, the controlled system satisfies $\boldsymbol{x}_{i|k} \in \mathbb{X}$ for $i = k+1, \ldots, k+N$, then the shifted sequence of nominal control inputs $\tilde{\boldsymbol{v}}_{k+1} = [\boldsymbol{v}_{k+1|k}^\top, \boldsymbol{v}_{k+2|k}^\top, \ldots, \boldsymbol{v}_{k+N-1|k}^\top, \kappa^\top(\boldsymbol{z}_{k+N|k})]^\top$ is feasible for the admissible pair $(\boldsymbol{z}_{k+1}, \boldsymbol{x}_{k+1})$ under the SDD-TMPC policy $\pi(\cdot)$.

Proof. The nominal system is deterministic, and at time step $k+1$, for the first $N-1$ time steps, the same nominal control inputs as for time step k are applied. Thus one can write $\boldsymbol{z}_{i|k+1} = \boldsymbol{z}_{i|k}$ for $i = k+1, \ldots, k+N$. The error set $\mathbb{E}_{k+1|k}$ contains all the possible errors of $\boldsymbol{x}_{k+1|k}$ with respect to $\boldsymbol{z}_{k+1|k}$. At time step $k+1$, however, $\mathbb{E}_{k+1|k+1}$ contains only one

element of $\mathbb{E}_{k+1|k}$ based on the realized state under the SDD-TMPC policy. Thus, there is $\mathbb{E}_{k+1|k+1} \subseteq \mathbb{E}_{k+1|k}$.

Since $\boldsymbol{z}_{k+1|k+1} = \boldsymbol{z}_{k+1|k}$ and $\mathbb{E}_{k+1|k+1} \subseteq \mathbb{E}_{k+1|k}$, from Theorem 2 there is $\{\boldsymbol{z}_{k+1|k+1}\} \oplus \mathbb{E}_{k+1|k+1} \subseteq \{\boldsymbol{z}_{k+1|k}\} \oplus \mathbb{E}_{k+1|k}$. From (2.4d)-(2.4e), the next error set is derived based on a mapping from all elements of the current error set added to the nominal state. Based on the last two statements, Definition 2, and Theorem 2, it is concluded that $\mathbb{E}_{k+2|k+1} \subseteq \mathbb{E}_{k+2|k}$. Similarly, in an iterative way, it can be shown that $\mathbb{E}_{i|k+1} \subseteq \mathbb{E}_{i|k}$ for $i = k+1, \ldots, k+N$.

The sequence $\tilde{\boldsymbol{v}}_k$ is assumed to be feasible for the admissible pair $(\boldsymbol{z}_k, \boldsymbol{x}_k)$. Thus the elements $\{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k}$ for $i = k+1, \ldots, k+N-1$ belong to the admissible state set. Now since $z_{i|k+1} = z_{i|k}$ and $\mathbb{E}_{i|k+1} \subseteq \mathbb{E}_{i|k}$ for $i = k+1, \ldots, k+N$, based on Theorem 4, the elements $\{\boldsymbol{z}_{i|k+1}\} \oplus \mathbb{E}_{i|k+1}$ for $i = k+1, \ldots, k+N-1$ belong to the admissible state set. Finally, since at time step $k+N$ the SDD-TMPC input is determined based on the nominal control input $\kappa(\boldsymbol{z}_{k+N|k})$ (or equivalently, based on $\kappa(\boldsymbol{z}_{k+N|k+1})$), and since $\mathbb{E}_{k+N|k+1} \subseteq \mathbb{E}_{k+N|k} \subseteq \mathbb{E}^{\max}$, from Theorem 4 it is concluded that $\{\boldsymbol{z}_{k+N|k+1}\} \oplus \mathbb{E}_{k+N|k+1}$ belongs to the admissible state set. □

Theorem 6. The optimal cost function $V^*(\cdot)$ in (2.4a) is a Lyapunov function for system (2.1) (which implies stability).

Proof. Given Assumption 4, the optimal cost function $V^*(\cdot)$ is lower bounded by $\alpha_1(\cdot)$. Moreover, an upper bound can be found for $V^*(\cdot)$, since it is continuous and finite for $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{X}$. The optimal sequence of nominal control inputs corresponding to the optimal cost $V^*(\boldsymbol{x}_k, \boldsymbol{z}_k)$ is given by $\boldsymbol{v}_k^* = [\boldsymbol{v}^{*\top}_{k|k}, \boldsymbol{v}^{*\top}_{k+1|k}, \ldots, \boldsymbol{v}^{*\top}_{k+N-1|k}]^\top$. Based on Theorem 5, for the next control time step the sequence $\tilde{\boldsymbol{v}}_{k+1} = [\boldsymbol{v}^{*\top}_{k+1|k}, \boldsymbol{v}^{*\top}_{k+2|k+1}, \ldots, \boldsymbol{v}^{*\top}_{k+N-1|k}, \kappa^\top(\boldsymbol{z}_{N|k})]^\top$ is feasible for the admissible pair $(\boldsymbol{z}_{k+1}, \boldsymbol{x}_{k+1})$ under the SDD-TMPC policy $\pi(\cdot)$. The difference between $V^*(\boldsymbol{x}_k, \boldsymbol{z}_k)$ and the cost under sequence $\tilde{\boldsymbol{v}}_{k+1}$ is $l(\boldsymbol{z}_{k|k}, \boldsymbol{v}_{k|k}) + V^{\mathrm{f}}(\boldsymbol{z}_{k+N|k}) - l(\boldsymbol{z}_{k+N|k+1}, \kappa(\boldsymbol{z}_{k+N|k+1})) - V^{\mathrm{f}}(f(\boldsymbol{z}_{k+N|k+1}, \kappa(\boldsymbol{z}_{k+N|k}))$. From Assumption 3, this difference is positive, implying that the optimal cost $V^*(\boldsymbol{x}_k, \boldsymbol{z}_k)$ is larger than the cost under $\tilde{\boldsymbol{v}}_{k+1}$, which itself is larger than or equal to the optimal cost $V^*(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1})$. Therefore, there is $V^*(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}) < V^*(\boldsymbol{x}_k, \boldsymbol{z}_k)$. □

2.4. CASE STUDIES

In this section, the following two numerical case studies are provided as they have been designed for a SaR robot:

- **Case study 1:** An SDD-TMPC controller is developed for a linear model of the robot, which is impacted by state-dependent disturbances. With this experiment, the unique capabilities of SDD-TMPC are shown in comparison with MPC and TMPC, to provide robustness with respect to the disturbances, while improving the feasibility and reducing the conservativeness. The codes for this experiment have been published in [31].

Table 2.2: Frequently used mathematical notations of the case study and their definition

Notation	Definition	Notation	Definition
$[A^{\text{ineq}}_{n_1\times n_2}, b^{\text{ineq}}_{n_1\times 1}]$	Pair of matrices describing a polytope in n_2 dimensions with n_1 inequalities	m	meters
$p_{x,k}$	Component of robot position parallel to x axis at time step k	$\frac{\text{m}}{\text{s}}$	meters per second
$p_{y,k}$	Component of robot position parallel to y axis at time step k	$\frac{\text{m}^2}{\text{s}}$	meters per squared second
$v_{x,k}$	Component of robot velocity parallel to x axis at time step k	$r^{\max}(\boldsymbol{x}_k)$	The major of the ellipsoidal disturbance set given current state
$v_{y,k}$	Component of robot velocity parallel to y axis at time step k	$r^{\min}(\boldsymbol{x}_k)$	The minor of the ellipsoidal disturbance set given current state
$a_{x,k}$	Component of robot acceleration parallel to x axis at time step k	T^{s}	Sampling time
$a_{y,k}$	Component of robot acceleration parallel to y axis at time step k	$I_{n\times n}$	Identity matrix with dimension n

- **Case study 2:** To showcase the performance of SDD-TMPC for nonlinear systems, an SDD-TMPC controller is developed for a nonlinear reference tracking problem, which has previously been addressed in [32] using nonlinear TMPC. While the robot steered via nonlinear TMPC suffers from a large rise time, with SDD-TMPC, the robot achieves a better performance by using less conservative control inputs. Consequently, the robot safely moves faster, still guaranteeing robustness to disturbances. Thus, the results indicate a decreased rise time using SDD-TMPC, compared to nonlinear TMPC, without violating the constraints. The code has been published in [33].

The results of these case studies, where the performance of SDD-TMPC is compared with MPC and TMPC, are presented and discussed in Section 2.5. Table 2.2 includes the mathematical notations that are frequently used in this section. The parameters and values used in the case studies are given in Appendix A1.

2.4.1. CASE STUDY 1

SIMULATION SETUP

A holonomic ground robot is simulated with the following discrete-time state-space

representation for its kinematics:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & T^{\mathrm{s}} & 0 \\ 0 & 1 & 0 & T^{\mathrm{s}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ T^{\mathrm{s}} & 0 \\ 0 & T^{\mathrm{s}} \end{bmatrix} \boldsymbol{u}_k + \begin{bmatrix} \boldsymbol{w}_k \\ 0_{2\times 1} \end{bmatrix} \tag{2.8}$$

where the state $\boldsymbol{x}_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^\top$ and control input $\boldsymbol{u}_k = [a_{x,k}, a_{y,k}]^\top$ vectors include, respectively, the position and velocity, and the acceleration of the robot in the x and y directions, T^{s} is the discretization sampling time and $\boldsymbol{w}_k$ is the 2-dimensional vector of the external disturbances that influence the position of the robot. It is assumed that the disturbances affect the position only (similarly as in [32]). The set $\mathbb{W}(\boldsymbol{x}_k)$ of external disturbances when the state of the robot is $\boldsymbol{x}_k$ is a two-dimensional ellipse, with its major axis parallel to the direction of the movement of the robot. The magnitude of the disturbances is nonlinearly dependent on the state of the robot, i.e.:

$$r^{\max}(\boldsymbol{x}_k) = 0.202\sqrt{v_{x,k}^2 + v_{y,k}^2} + r^{\min}(\boldsymbol{x}_k) \tag{2.9a}$$

$$r^{\min}(\boldsymbol{x}_k) = 0.225\beta^2(\boldsymbol{x}_k)\sqrt[4]{v_{x,k}^2 + v_{y,k}^2} \tag{2.9b}$$

with $r^{\max}(\boldsymbol{x}_k)$ and $r_{\min}(\boldsymbol{x}_k)$ the length of the, respectively, major and minor axes of the ellipse. The external disturbances may vary across the environment; thus, the parameter $\beta(\boldsymbol{x}_k)$, an indication of the ground slipperiness, is defined to include this variation in the simulations. It is assumed that the robot has perfect knowledge of its state and $\beta(\boldsymbol{x}_k)$, but has no information on (2.9). The constraints on the position of the robot vary in different simulations, while the constraints on the inputs and the velocity of the robot are always the following:

$$|v_{x,k}| < 2\frac{\mathrm{m}}{\mathrm{s}},\ |v_{y,k}| < 2\frac{\mathrm{m}}{\mathrm{s}},\ |a_{x,k}| < 5\frac{\mathrm{m}}{\mathrm{s}^2},\ |a_{y,k}| < 5\frac{\mathrm{m}}{\mathrm{s}^2} \tag{2.10}$$

FORMULATING SDD-TMPC FOR PATH PLANNING

For path planning of the SaR robot, an SDD-TMPC problem with a quadratic cost function is given by:

$$V(\boldsymbol{z}_k, \boldsymbol{v}_k) = \sum_{i=k}^{k+N-1} \left(\|\boldsymbol{z}_{i|k}\|_Q^2 + \|\boldsymbol{v}_{i|k}\|_R^2 \right) + \|\boldsymbol{z}_{k+N|k}\|_F^2 \tag{2.11}$$

A common choice for the ancillary control input is a linear state feedback, which for time step k is given by:

$$\boldsymbol{u}_k = K\boldsymbol{e}_k + \boldsymbol{v}_k \tag{2.12}$$

where $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{z}_k$. For determining K, a Linear Quadratic Regulator (LQR) method is used (the corresponding matrices are given in Appendix A1), such that the ancillary control law

more aggressively reduces the position. Thus, the influence of the state-dependent external disturbances is more significantly reduced. The error $\boldsymbol{e}_k$ evolves according to:

$$\boldsymbol{e}_{k+1} = (A+BK)\boldsymbol{e}_k + \left[\boldsymbol{w}_k^\top | 0_{1\times 2}\right]^\top \tag{2.13}$$

with $A_{4\times 4}$ and $B_{4\times 2}$ the dynamic and input matrices in (2.8). Thus the equivalent of (2.4c) and (2.4j) for this case study are:

$$\bar{\mathbb{E}}_{i+1|k} = (A+BK)\mathbb{E}_{i|k}, \qquad i = k,\dots,k+N-1 \tag{2.14}$$

$$K\mathbb{E}_{i|k} \oplus \{\boldsymbol{v}_{i|k}\} \subseteq \mathbb{U}, \qquad i = k,\dots,k+N-1 \tag{2.15}$$

Definition 2 is used to compute (2.14) and (2.15). To reduce the computation time, and since the external disturbance set changes slowly with the states, (2.4d) is simplified by only considering the centroid of the tubes as input:

$$\bar{\mathbb{W}}_{i|k} = f^{\text{dis}}\left(\boldsymbol{z}_{i|k} + (A+BK)^{i-k}\boldsymbol{e}_k\right) \tag{2.16}$$

To determine the terminal admissible set $\mathbb{Z}^{\text{f}}$ (a positive invariant polytope) for the nominal state, it is assumed that the system follows an LQR control law $\kappa(\cdot)$ beyond the prediction horizon. Thus, the terminal cost in (2.11) is the limit of the cost for $k \to \infty$, when the system follows the LQR control law $\kappa(\cdot)$, and is determined by solving the following Riccati equation, using, e.g., Matlab dlyap command [34]:

$$F = (A+BK)^T F (A+BK) + Q_\kappa + K^T R_\kappa K \tag{2.17}$$

For details on the derivation of (2.17), see [2]. Control policy $\kappa(\cdot)$ should be designed such that Assumption 3 holds.

LEARNING THE EXTERNAL DISTURBANCE SETS USING A FIS

A FIS can describe the dynamics of a (generally nonlinear) system via rules that are formulated via linguistic terms (e.g., very high), where this nonlinear mapping is interpretable– unlike, e.g., NN [35]. The rules of a FIS can be generated based on experimental data gathered before the beginning of a mission or can be deduced directly from expert knowledge available in human language. In this case study, two TSK-FIS are considered, one for approximating the major and one for the minor in (2.9) per time step k, with their output functions defined by:

$$r^m(\boldsymbol{x}_k) = c_1^m v_k^2 + c_2^m \beta^2(\boldsymbol{x}_k) + c_3^m v_k \beta(\boldsymbol{x}_k) + c_4^m v_k + c_5^m \beta(\boldsymbol{x}_k) + c_6^m \tag{2.18}$$

with $v_k^2 = v_{x,k}^2 + v_{y,k}^2$, $m \in \{\max, \min\}$, and $\boldsymbol{\theta}^m = [c_1^m \ \dots \ c_6^m]^\top$.

However, the Minkowski summation of two ellipses does not result in an ellipse. To solve this problem, $\mathbb{W}(\boldsymbol{x}_k)$ is approximated by $\bar{\mathbb{W}}_k$, which is a polytopic set defined by $\{[x,y]^\top : A_{8\times 2}^{\text{ineq}}[x,y]^\top \leq \boldsymbol{b}_{8\times 1}^{\text{ineq}}\}$ with a fixed number of edges that encounters the resulting

ellipse, and $\bar{\mathbb{W}}_k$ is used as the disturbance set at time step k. For polytope sets, unlike ellipses, the Minkowski addition can be computed without approximation, and the set remains a polytope afterward.

IMPLEMENTATION OF SDD-TMPC

The following 4 scenarios, relevant for SaR, are considered. Each scenario assesses one key property of SDD-TMPC. The scenarios are simulated several times to show the behavior of the controllers under different disturbances.

Scenario 1: Closely following a reference path The main aim of the comparison among MPC, TMPC, and SDD-TMPC in scenario 1 is to assess how fast a robot that is controlled by each of these approaches reaches a given destination, following a reference path (i.e., an ordered set of given positions). As soon as the (nominal) state of the robot reaches a distance of 0.3 m from its reference position, the next reference position from the path is followed by the controller. Scenario 1 resembles a real-life SaR situation when a robot should precisely travel along a given path in the presence of external disturbances (e.g., to avoid hazards that exist in the close vicinity of the robot). Scenario 1 is simulated 3 times, considering the following situations:

1. The robot is not affected by external disturbances.
2. The external disturbances help in attracting the robot to the current reference position.
3. The external disturbances result in repelling the robot from the current reference position.

SaR robots should often move close to obstacles. In such cases, it should be guaranteed that no crashes occur, whereas overly conservative decisions significantly slow down the mission. In scenario 2, the robot should move from position $[2, 0.01]^\top$ to position $[8.5, 0.01]^\top$. Whenever the vertical position of the robot falls below $p_y = 0$, a crash with an obstacle (e.g., a wall extended across $p_y = 0$) occurs. Moreover, the value of $\beta(\boldsymbol{x}_k)$ is determined via:

$$\beta(\boldsymbol{x}_k) = \frac{1}{|5 - p_{x,k}| + 1} \tag{2.19}$$

Scenario 2 is simulated 3 times, considering the following situations:

1. The robot is not affected by external disturbances.
2. The robot is pushed upwards by external disturbances.
3. The robot is pushed downwards by external disturbances.

Scenario 3: Approaching an unreachable target SaR robots may need to move in very narrow corridors, avoiding crashes into the walls. TMPC may become infeasible in such cases due to conservativeness. Scenario 3 simulates such an infeasible problem for TMPC. The robot starting at position $[5.5,0]^\top$ with a zero initial speed should reach position $[11,0]^\top$ in a narrowing corridor. To avoid crashes into the walls, the following hard constraints are defined on the robot's position for all time steps k during the simulation:

$$p_{x,k} + 8p_{y,k} < 10; \quad p_{x,k} - 8p_{y,k} < 10 \tag{2.20}$$

Note that all coordinates are given in m. While TMPC is infeasible already at the initial position of the robot (the farthest position where TMPC generates a feasible tube is $p_x = 4.59$ m, whereas the width of the corridor is about 0.68 m (see Figure 2.6)), it is investigated how far the robot moves forward using SDD-TMPC
Scenario 3 is simulated twice, considering the following situations:

1. The robot is not affected by external disturbances.
2. The robot is pushed (relative to the speed value) upwards via external disturbances.

Scenario 4: Ability to make high-level optimal decisions Next to robustness, SDD-TMPC should improve the performance by making high-level optimal decisions. In scenario 4, I consider a case where the robot has to move from position $[2.3,3]^\top$ to position $[3.25,3.05]^\top$, with an obstacle on its way (see Figure 2.9). The robot initially has a horizontal speed of $v_x = 1 \frac{\text{m}}{\text{s}}$, and should select an initial vertical speed of either $v_y = 1.2 \frac{\text{m}}{\text{s}}$ (i.e., moving upwards) or $v_y = -1.2 \frac{\text{m}}{\text{s}}$ (i.e., moving downwards).

2.4.2. CASE STUDY 2

In [32], two controllers, nonlinear TMPC and another nonlinear RMPC, have been used for a nonlinear reference tracking problem. A similar experiment using SDD-TMPC is simulated. The only differences concern:

- Disturbance generator: In [32], only the boundaries of the disturbances are given.
- Solver: Despite using the same algorithm (interior point) from Matlab's optimization toolbox, due to the non-convex nature of the optimization problem, It cannot be ensured that the same solutions are identified.
- Tube initialization: In [32], the solver chooses the initial nominal position per time step, provided that the measured position remains within the tube. The starting nominal direction is, however, not given. The solver is allowed to freely choose the direction.
- Discretization method: In [32], the system is discretized using the ICLOCS toolbox [36]. Matlab's c2d function, Zero-Order Hold [37], and Runge-Kutta 4 algorithm to discretize the linear and nonlinear systems are used.

- Terminal state sets
- Hardware

Two unicycle robots are simulated: a leader, which is assumed to remain unaffected by the external disturbances, and a follower, which is controlled with implemented controllers and is affected by the disturbances. A unicycle robot is a circular robot, steered at time instant t by the velocities v_t^{l} and v_t^{r} of its, respectively, left and right wheels. The head of the robot is its front point, located on the main axis of the robot that is perpendicular to the axis of the wheels of the robot. The inputs to the robot at time instant t are the linear velocity v_t and the angular velocity ω_t, given by:

$$v_t = (v_t^{\mathrm{l}} + v_t^{\mathrm{r}})/2, \qquad \omega_t = (v_t^{\mathrm{r}} - v_t^{\mathrm{l}})/(2\rho) \tag{2.21}$$

where ρ represents the radius of the robot. Moreover, the admissible set of the inputs is defined by:

$$\mathbb{U} := \left\{ [v, \omega]^T \in \mathbb{R}^2 \middle| \frac{|v|}{v^{\max}} + \frac{\rho|\omega|}{v^{\max}} \leq 1 \right\} \tag{2.22}$$

where $v^{\max}$ is the maximum absolute velocity of the wheels.

The leader robot utilizes constant reference inputs v^{R} and ω^{R}. The state $\boldsymbol{x}_t^{\mathrm{R}}$ of the leader robot per time instant t is characterized by the position $(p_{x,t}^{\mathrm{R}}, p_{y,t}^{\mathrm{R}})$ of its center and by the orientation ψ_t^{R} of the robot. The state evolution, in the continuous time domain, for the leader robot at time instant t is determined by:

$$\begin{bmatrix} \dot{p}_{x,t}^{\mathrm{R}} \\ \dot{p}_{y,t}^{\mathrm{R}} \\ \dot{\psi}_t^{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} v^{\mathrm{R}} \cos(\psi_t^{\mathrm{R}}) \\ v^{\mathrm{R}} \sin(\psi_t^{\mathrm{R}}) \\ \omega^{\mathrm{R}} \end{bmatrix} \tag{2.23}$$

The input $\boldsymbol{u}_t$ to the follower robot at time instant t consists of its linear v_t and angular ω_t velocities, and its state $\boldsymbol{x}_t$ includes the position $(p_{x,t}, p_{y,t})$ of its head and the robot orientation ψ_t. The position is impacted by the unknown disturbance vector $\boldsymbol{w}_t = [w_{x,t}, w_{y,t}]^T$, where $\sqrt{w_{x,t}^2 + w_{y,t}^2} < \eta$. The state evolution for the follower robot is given by:

$$\dot{\boldsymbol{x}}_t = \begin{bmatrix} \dot{p}_{x,t} \\ \dot{p}_{y,t} \\ \dot{\psi}_t \end{bmatrix} = \begin{bmatrix} v_t \cos(\psi_t) + \rho\,\omega_t \sin(\psi_t) \\ v_t \sin(\psi_t) + \rho\,\omega_t \cos(\psi_t) \\ \omega_t \end{bmatrix} + \begin{bmatrix} w_{x,t} \\ w_{y,t} \\ 0 \end{bmatrix} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_t) \tag{2.24}$$

FORMULATING SDD-TMPC

The nonlinear ancillary control law used for SDD-TMPC is the same as the one in [32]:

$$\boldsymbol{u}_t = \begin{bmatrix} \cos(\boldsymbol{x}_t[3]) & -\rho \sin(\boldsymbol{x}_t[3]) \\ \sin(\boldsymbol{x}_t[3]) & \rho \cos(\boldsymbol{x}_t[3]) \end{bmatrix}^{-1} \left(\begin{bmatrix} \cos(\boldsymbol{z}_t[3]) & -\rho \sin(\boldsymbol{z}_t[3]) \\ \sin(\boldsymbol{z}_t[3]) & \rho \cos(\boldsymbol{z}_t[3]) \end{bmatrix} \boldsymbol{v}_t - K^{\mathrm{e}} \boldsymbol{e}_t[1:2] \right) \tag{2.25}$$

where K^{e} is a constant design parameter and the notation $\boldsymbol{e}_t[1:2]$ implies elements 1 and 2 of vector $\boldsymbol{e}_t$. This ancillary control law linearizes the dynamics of the position error of the controlled system (see Appendix A2 for details). The dynamics of the position and direction errors are formulated via:

$$\dot{\boldsymbol{e}}_t[1:2] = K^{\mathrm{e}}\boldsymbol{e}_t[1:2] + [w_x, w_y]^T, \qquad \dot{\boldsymbol{e}}_t[3] = -\frac{1}{\rho}\boldsymbol{e}_t[3]\boldsymbol{v}_t[1] + w_t^{\mathrm{e}} \tag{2.26}$$

where the description, derivation, and lower and upper bounds of w_t^{e} (which is state-dependent and contains the nonlinear terms in the directional error evolution equation) are given in detail in Appendix A3. The above equations are given in the continuous time domain. At the end, the derived equations may be discretized in time.

In this case study, to speed up the computations, SDD-TMPC uses a box to represent the error set, which is larger than the real error set. However, since the disturbance set is a box and the error dynamics of the variables are independent, the error set is also a box set for the entire optimization iterations. This means that the entire error set can be described by 3 times as many state variables, which makes the computational complexity linear. In the previous case study, where the smallest possible sets are used, the computational complexity is growing exponentially with the size of the state vector and prediction horizon.

The nonlinear optimization problem (in the discrete time domain) solved by SDD-TMPC per time step is:

$$V^*(\boldsymbol{x}_k, \boldsymbol{z}_k) = \min_{\tilde{\boldsymbol{z}}_k, \tilde{\boldsymbol{v}}_k, \mathbb{T}_k} \sum_{i=k}^{k+N-1} \left(\left\| \boldsymbol{z}^{\mathrm{r}}_{i|k} \right\|^2_Q + \left\| \boldsymbol{u}^{\mathrm{r}}_{i|k} \right\|^2_R \right) + \left\| \boldsymbol{z}^{\mathrm{r}}_{k+N|k} \right\|^2_F \tag{2.27a}$$

$$\text{s.t. for } i = k+1, \ldots, k+N\text{:}$$

$$\boldsymbol{z}_{i+1|k} = f^{\mathrm{d}}(\boldsymbol{z}_{i|k}, \boldsymbol{v}_{i|k}) \tag{2.27b}$$

$$\mathbb{T}_k = \{\{z_{k+1|k}\} \oplus \mathbb{E}_{k+1|k}, \ldots, \{z_{k+N|k}\} \oplus \mathbb{E}_{k+N|k}\} \tag{2.27c}$$

$$\bar{\mathbb{E}}_{i|k} = \mathrm{diag}\left(K^{\mathrm{e,d}}, K^{\mathrm{e,d}}, K^{\theta,\mathrm{d}}\right) \mathbb{E}_{i-1|k} \tag{2.27d}$$

$$\mathbb{E}_{i|k} = \bar{\mathbb{E}}_{i|k} \oplus \mathbb{W}_{i-1|k}(\mathbb{E}_{i-1|k}, \boldsymbol{z}_{i-1|k}, \boldsymbol{v}_{i-1|k}) \tag{2.27e}$$

$$\boldsymbol{x}_k, \boldsymbol{z}_k \text{ are given} \tag{2.27f}$$

$$\mathbb{E}_k = \{\boldsymbol{x}_k - \boldsymbol{z}_k\} \tag{2.27g}$$

$$v_{i|k} \in \mathbb{V}(\mathbb{E}_{i|k}) \tag{2.27h}$$

$$\boldsymbol{z}^{\mathrm{r}}_{k+N|k} \in \mathbb{Z}^{\mathrm{f}} \tag{2.27i}$$

where $f^{\mathrm{d}}(\cdot)$ is the discretized dynamic function that is determined after time-discretization of (2.24), and $K^{\mathrm{e,d}}$ and $K^{\theta,\mathrm{d}}$ are coefficients that are multiplied by, respectively, discretized errors $\boldsymbol{e}_i[1]$ (or $\boldsymbol{e}_i[2]$) and $\boldsymbol{e}_i[3]$, after discretization of the error dynamics equation (2.26), and are

explained in more detail in Appendix A2. For control input constraint tightening (2.27h), the following set in the continuous time is proposed:

$$\mathbb{V}(\mathbb{E}_t) := \min_{\boldsymbol{e}_t[3]\in\mathbb{E}_t[3]} (\lambda(\boldsymbol{e}_t[3]) \mathbb{U} \ominus \begin{bmatrix} -\cos(\boldsymbol{z}_t[3]) & -\sin(\boldsymbol{z}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{z}_t[3]) & -\frac{1}{\rho}\cos(\boldsymbol{z}_t[3]) \end{bmatrix} K^{\mathrm{e}}(\mathbb{E}_t[1] \times \mathbb{E}_t[2]) \tag{2.28}$$

The details on the derivation of this tightened set are given in Appendix A4.

2.5. RESULTS AND DISCUSSION

All simulations are implemented in MATLAB 2022b on the same computer with an 11th Gen Intel(R) Core(TM) i7-1185G7 with a 3GHz frequency. In the simulations, the solution of TMPC is used as a warm start for SDD-TMPC, and in case TMPC is infeasible, the shifted trajectory from Theorem 5 is used.

2.5.1. CASE STUDY 1: TRAINING THE FUZZY INFERENCE SYSTEMS

Table 2.3: Results for the trained FISs that estimate the lengths of the major and minor axes of the disturbance ellipse sets using the test data. The error values are normalized by dividing by the maximum ground truth value.

	FIS: major estimation	FIS: minor estimation
Cases with negative errors (%)	2.64	3.01
Mean value of negative errors	7.8×10^{-4}	1.93×10^{-3}
Mean value of positive errors	1.02×10^{-2}	2.32×10^{-2}
Maximum value of negative errors	5.89×10^{-3}	2.56×10^{-2}
Maximum value of positive errors	5.94×10^{-2}	2.45×10^{-1}

First, the results of training the FISs are given for case study 1. 1240 input-output pairs are generated, based on the ground truth ellipsoids given by (2.9). This data set is divided into a training and a validation set, with a proportion of 660 : 580. For the test set, a large data set with 100000 pairs of input-output are generated using (2.9) to extensively test the trained FISs. Given the training and the validation errors, the best results are achieved using 5 fuzzy sets per input, which resulted in 25 fuzzy rules with outputs that are described by (2.18). Note that compared to different expressions for the output of the rules, the expression given by (2.18) does not result in under- or over-fitting. GA has been used to train the FISs by minimizing the mean square error, with an additional penalty for negative errors.

The results obtained for the test set for the two FISs that estimate the lengths of the major and minor axes of the ellipse sets for the external disturbances are summarized in Table 2.3, where the error values have been normalized. In general, both FISs are able to approximate the ground truth with mostly positive-valued errors. In case negative-valued errors exist, a penalty for small positive errors may be considered to ensure to eliminate all these negative-valued errors as well.

To illustrate the impact on performance, a one-step simulation was performed twice using two SDD-TMPC controllers: one using FIS and the other utilizing ground truth values. In this simulation, the controllers were utilized to control the problem outlined in Scenario 1, but with a horizon equal to 20. As illustrated in Figure 2.3, there is a discrepancy between the tubes of both SDD-TMPCs. Note that, even if the utilization of FIS does not result in perfect approximation, FIS, having been trained with a greater degree of conservativeness, can be relied upon to ensure the safety of SDD-TMPC. Note that the results presented in Section 2.5.2 demonstrate that, in scenarios where conservative tube generation with FIS is used, SDD-TMPC still outperforms TMPC.

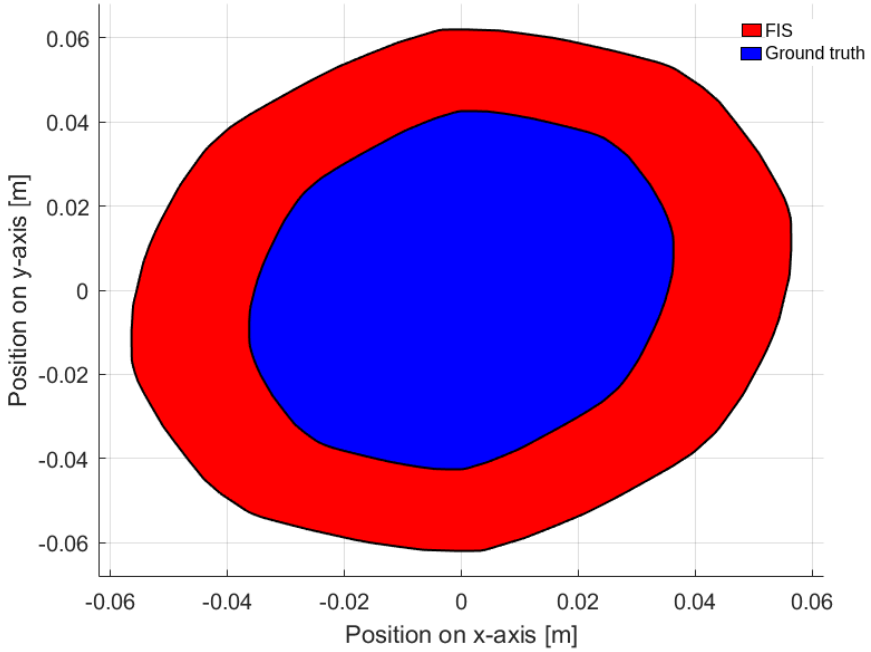


Figure 2.3: The projection of tubes on the position at time step 20 is represented by the colors red and blue, which correspond to tubes generated with FIS and the ground truth function, respectively.

2.5.2. Case Study 1: Comparison of MPC, TMPC, SDD-TMPC

In the simulations, MPC (2.29), TMPC (2.30), and SDD-TMPC (2.31) are implemented, and their performance, with respect to minimizing the cost function, satisfying the hard constraints, and computation time are compared. To solve the optimization problems, the quadratic programming solver from MATLAB's Optimization toolbox is used for MPC and TMPC, whereas for SDD-TMPC (which solves a non-convex nonlinear optimization problem), PS

[29] from MATLAB Global optimization toolbox is implemented. Unless stated otherwise, the optimization procedure for SDD-TMPC is terminated after 60 iterations, which takes about 4 minutes to run and provides a balanced trade-off between the computation time and the accuracy of the solutions. In comparison, MPC and TMPC solve their quadratic problems in milliseconds. The initial nominal state is assumed to be equal to the real state. Terminal constraint and cost are not used in scenarios 2 and 3, respectively, for computational efficiency and since the goal is infeasible. To compare the convergence of the optimization algorithms, in a sample problem, the robot moves to the zero state, starting from state $[0.35, 0.65, 0, 0]^\top$. Table 2.4 shows the optimal costs for MPC, TMPC, and SDD-TMPC. Since SDD-TMPC solves a non-convex, nonlinear optimization problem, it may not be possible to find a global minimum. Therefore, the changes in the value of the cost by iteration of the PS algorithm are given in the table (the number of iterations is given in parentheses). The rate of these changes depends strongly on how constrained the problem is (compare the third and sixth rows of the table). As expected, the cost values corresponding to TMPC and MPC are, respectively, an upper and a lower bound for the cost of SDD-TMPC.

$$V^*(\boldsymbol{x}_k) = \min_{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k} \sum_{i=k}^{k+N-1} \left(\|\boldsymbol{x}_{i|k}\|_Q^2 + \|\boldsymbol{u}_{i|k}\|_R^2 \right) + \|\boldsymbol{x}_{k+N|k}\|_F^2 \tag{2.29}$$
$$\text{s.t. for } i = k+1, \dots, k+N : \boldsymbol{x}_{i|k} = A\boldsymbol{x}_{i-1|k} + B\boldsymbol{u}_{i-1|k},$$
$$\boldsymbol{x}_k \text{ is given}, \quad \boldsymbol{x}_{i|k} \in \mathbb{X}, \quad \boldsymbol{u}_{i|k} \in \mathbb{U}, \, \boldsymbol{x}_{N|k} \in \mathbb{Z}^{\mathrm{f}}$$

$$V^*(\boldsymbol{z}_k) = \min_{\bar{\boldsymbol{z}}_k, \bar{\boldsymbol{v}}_k} \sum_{i=k}^{k+N-1} \left(\|\boldsymbol{z}_{i|k}\|_Q^2 + \|\boldsymbol{v}_{i|k}\|_R^2 \right) + \|\boldsymbol{z}_{k+N|k}\|_F^2 \tag{2.30}$$
$$\text{s.t. for } i = k+1, \dots, k+N : \boldsymbol{z}_{i|k} = A\boldsymbol{z}_{i-1|k} + B\boldsymbol{v}_{i-1|k}, \quad \boldsymbol{z}_k \text{ is given},$$
$$\boldsymbol{z}_{i|k} \in \mathbb{X} \ominus \mathbb{E}_{\max}, \quad \boldsymbol{v}_{i|k} \in \mathbb{U} \ominus \mathbb{E}_{\max}, \quad \boldsymbol{z}_{N|k} \in \mathbb{Z}^{\mathrm{f}}$$

$$V^*(\boldsymbol{x}_k, \boldsymbol{z}_k) = \min_{\bar{\boldsymbol{z}}_k, \bar{\boldsymbol{v}}_k, \mathbb{T}_k} \sum_{i=k}^{k+N-1} \left(\|\boldsymbol{z}_{i|k}\|_Q^2 + \|\boldsymbol{v}_{i|k}\|_R^2 \right) + \|\boldsymbol{z}_{k+N|k}\|_F^2 \tag{2.31}$$
$$\text{s.t. for } i = k+1, \dots, k+N : \quad \boldsymbol{z}_{i|k} = A\boldsymbol{z}_{i-1|k} + B\boldsymbol{v}_{i-1|k},$$
$$\mathbb{T}_k = \{\{z_{k+1|k}\} \oplus \mathbb{E}_{k+1|k}, \dots, \{z_{N|k}\} \oplus \mathbb{E}_{k+N|k}\}, \quad \bar{\mathbb{E}}_{i|k} = (A+BK)\mathbb{E}_{i-1|k},$$
$$\bar{\mathbb{W}}_{i-1|k} = \mathrm{FIS}\left(\boldsymbol{z}_{i|k} + (A+BK)^{i-k}(\boldsymbol{x}_k - \boldsymbol{z}_k)\right), \quad \mathbb{E}_{i|k} = \bar{\mathbb{E}}_{i|k} \oplus \bar{\mathbb{W}}_{i-1|k},$$
$$\mathbb{E}_k = \{\boldsymbol{x}_k - \boldsymbol{z}_k\}, \{\boldsymbol{z}_{i|k}\} \oplus \mathbb{E}_{i|k} \subseteq \mathbb{X}, \quad K\mathbb{E}_{i|k} \oplus \{\boldsymbol{v}_{i|k}\} \subseteq \mathbb{U},$$
$$\boldsymbol{z}_{N|k} \in \mathbb{Z}^{\mathrm{f}} \quad \boldsymbol{x}_k, \boldsymbol{z}_k \text{ are given}$$

Remark. For determining $\mathbb{Z}^{\mathrm{f}}$ the procedure is started by computing the Minkowski difference of $\mathbb{X}$ and $\mathbb{E}^{\max}$, to generate a candidate admissible terminal set, shown by $\mathbb{Z}^{\mathrm{c}}$. This set is then

transformed into the set $\mathbb{Z}^d$ through the system dynamics, using the terminal control law. In case the resulting set $\mathbb{Z}^d$ is a subset of $\mathbb{Z}^c$, then $\mathbb{Z}^c$ is the admissible terminal set for the nominal states. Otherwise, the intersection of $\mathbb{Z}^c$ and $\mathbb{Z}^d$ is the new candidate admissible terminal set for the nominal states, and the procedure is repeated until the condition $\mathbb{Z}^d \subseteq \mathbb{Z}^c$ is satisfied.

Table 2.4: Comparing the cost values for MPC, TMPC, and different numbers of iterations of PS, which solves SDD-TMPC when the robot should reach the zero state from state $[0.35, 0.65, 0, 0]^\top$ (no stage position constraints).

Without terminal constraints and terminal cost						
MPC	TMPC	PS(40)	PS(80)	PS(120)	PS(160)	PS(200)
339.23	379.76	340	339.6	339.6	339.6	339.6
With terminal constraints and terminal cost						
MPC	TMPC	PS(40)	PS(80)	PS(120)	PS(160)	PS(200)
402.91	781.32	667.05	597.10	540.48	531.28	528.31

Scenario 1: The mission time of the robot for the 3 MPC-based controllers is represented in Table 2.5: For MPC, the mission time is strongly affected by the external disturbances (see the significant variations in the mission time in different cases), whereas TMPC and SDD-TMPC provide more consistent results independent of the realized disturbances. TMPC needs more time than SDD-TMPC to finish the mission. In fact, to keep the states within the tube of TMPC (determined for the worst disturbance case), the nominal acceleration and speed of the robot have to remain within 60% and 50% of their maximum allowed values.

Table 2.5: The mission time required by the robot in scenario 1 to closely track the entire reference path. "No terminal" and "Terminal" show whether terminal cost and constraints are added to the optimization problem or not.

	Mission time [s] (No terminal)			Mission time [s] (Terminal)		
	MPC	TMPC	SDD-TMPC	MPC	TMPC	SDD-TMPC
Case 1	10.8	12.9	10.7	12.5	13.3	11.5
Case 2	7.8	12.9	10.7	8.6	13.3	11.3
Case 3	18	12.9	10.7	19.8	13.3	11.4

Furthermore, the extent of the sacrifices made by both TMPC and SDD-TMPC to ensure robustness is demonstrated in Figure 2.4, which presents the tube of TMPC and the evolution of the dynamic tube. Given the nature of the tubes as four-dimensional sets, Figure 2.4 shows two-dimensional projections. In the presented example, both controllers were assigned the task of performing a one-step simulation at the beginning of Scenario 1. However, the horizons were set to 20 to more clearly illustrate the evolution of the dynamic tube. It has been demonstrated that the dynamic tube is consistently smaller than the tube of TMPC. Furthermore, the dynamic tube exhibits reactivity. When the system is predicted to be close to

the destination, it is logical to reduce speed. The phenomenon is characterized by a reduced degree of uncertainty, and the dynamic tube responds by lowering its size. As demonstrated in Scenario 3, the controller can reduce the tube intentionally.

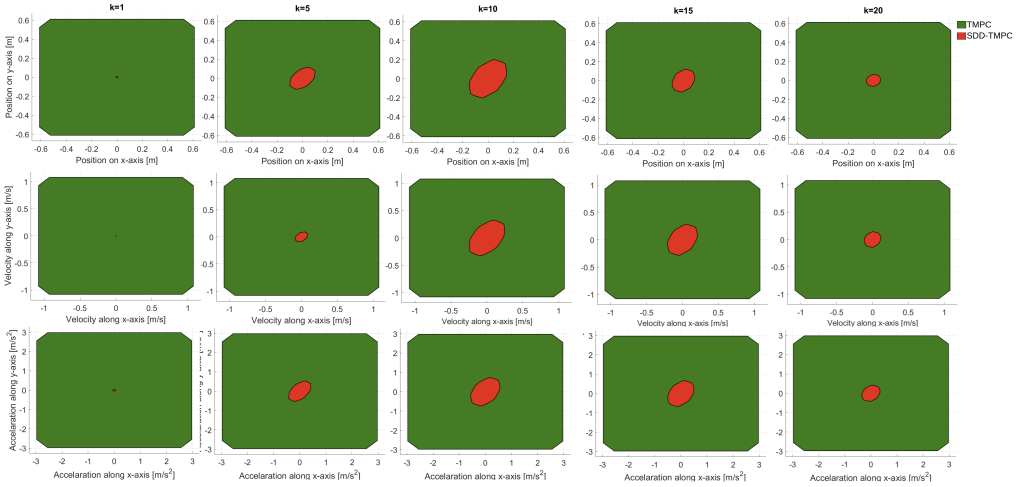


Figure 2.4: The green and red colors represent tubes of the TMPC and the SDD-TMPC, respectively. The first and second rows illustrate projections of the 4-dimensional tubes on the position and velocity, respectively. The third row of the table illustrates the tubes projected on the input space. Each column displays the progression of the tubes over time following 1, 5, 10, 15, and 20 steps.

Scenario 2: The results of the simulations for scenario 2 are shown in Figures 2.5: MPC steers the robot close to the wall, and when the robot is pushed downwards due to external disturbances, it collides with the wall. TMPC provides a safe, collision-free trajectory for the robot, moving relatively far from the wall. The initial nominal state of the robot with TMPC is considered far enough from the wall to avoid infeasibility (i.e., initial tube colliding with the wall). From the nominal and realized trajectories of the robot, when controlled via SDD-TMPC, it moves much closer to the wall compared to TMPC, but never crashes into it. Note that from (2.4), the nominal states of SDD-TMPC are determined based on an estimation of the external disturbances. Thus, the nominal trajectories vary with the disturbances (the more slippery the ground, the more careful the actions of SDD-TMPC).

Scenario 3: To reduce the risks of collision with the wall, the prediction horizon for this scenario is 6 (i.e., larger than in other scenarios), since the target is outside of the feasible state set and there is no terminal constraint. In Figure 2.7, the evolution of the position and velocity of the robot when no external disturbances exist is shown, using SDD-TMPC to steer the robot. Since there is no wall in a close neighborhood of the robot, it first moves faster and then slows down in time (thanks to the larger prediction horizon) as the corridor narrows

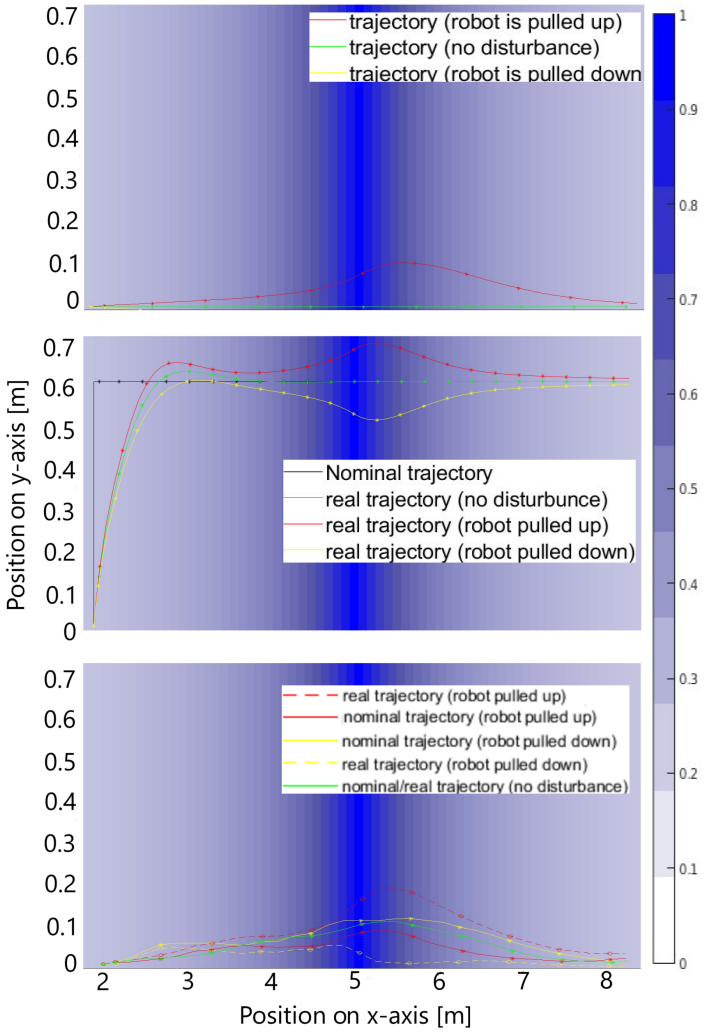


Figure 2.5: Scenario 2: Trajectories of the robot position when controlled by MPC (top plot), TMPC (middle plot) and SDD-TMPC (bottom plot), in case 1 (in green), 2 (in red), and 3 (in yellow). Note that the spectrum of blue corresponds to the values of $\beta(\boldsymbol{x}_k)$ for all realized states $\boldsymbol{x}_k$ of the robot during the simulation. In the middle plot, the black trajectory corresponds to the nominal case. For the bottom plot, the corresponding nominal and real trajectories are shown in the same color but with solid and dashed lines, respectively.

down. In fact, with SDD-TMPC the robot can move further through the corridor, compared to when TMPC is used. In case external disturbances push the robot upwards relative to its speed value, a similar pattern of behavior for the robot is observed, although the increase in the speed is less significant than the case without external disturbances. Figure 2.8 shows the trajectories for the position of the robot for both cases, i.e., with and without external disturbances: SDD-TMPC changes the nominal vertical position (in the presence of external disturbances), which, in a vacuum, raises the cost (because of deviating from the reference trajectory). However, SDD-TMPC makes this decision for the nominal trajectory of the robot, because it foresees that it significantly cancels out the impact of the external disturbances for the realized trajectory $p_{y,k}$ of the robot via the ancillary control input. Such behavior cannot be obtained via regular MPC and TMPC.

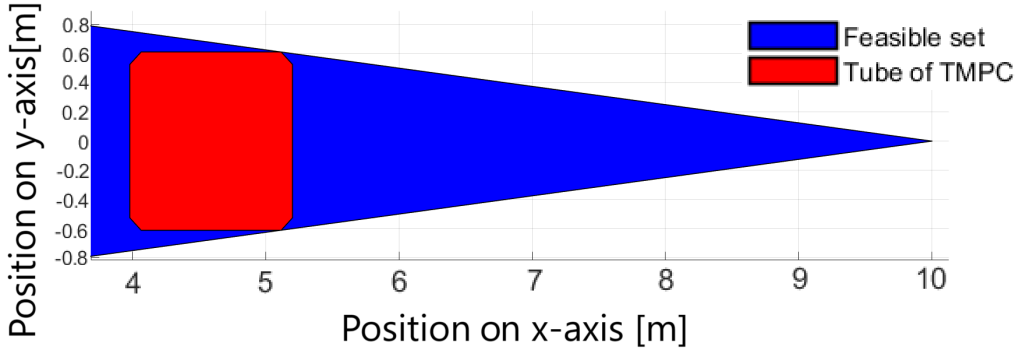


Figure 2.6: Scenario 3: The feasible state set $\mathbb{X}$ (in blue) and the tube of TMPC (in red).

Scenario 4: When the robot moves upwards, the slipperiness coefficient is larger and thus, the robot is prone to larger disturbances, compared to when it moves downwards. The larger external disturbances results in a longer realized path (due to an increased distance from the obstacle) for the robot. Since MPC and TMPC do not take into consideration the dynamics of the external disturbances, they estimate smaller costs when v_y is positive, whereas SDD-TMPC returns a lower cost (772) for the trajectory with smaller disturbances that corresponds to an initial vertical speed of $v_y = -1 \frac{\text{m}}{\text{s}}$, compared to the cost (917) for the other trajectory.

2.5.3. CASE STUDY 2: TRANSIENT RESPONSE BEHAVIOR

The leader and follower robots are simulated for 100 time steps, based on [38], with $v^{\max} = 0.13 \frac{\text{m}}{\text{s}}$ and $\rho = 0.0267\text{m}$. The aim is to maintain a constant distance $[-0.1, -0.1]^T$ (measured in the local coordinates of the leader robot) between the robots, in the local coordinate frame of the leader robot (the origin of this local frame coincides with the position of the center of the leader robot and the x-axis is parallel to the heading of the robot). The

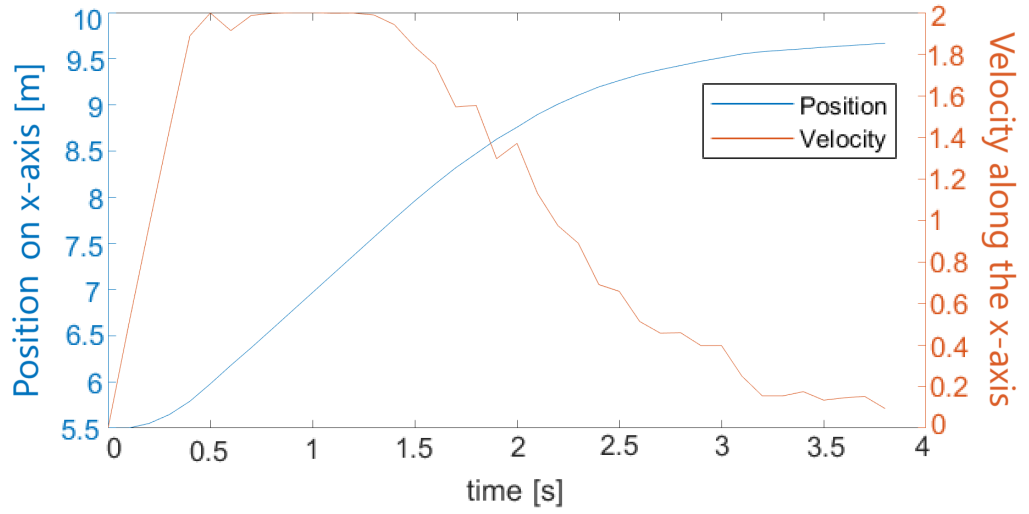


Figure 2.7: Scenario 3: The evolution of the position p_x (blue) and the velocity v_x (orange) of the robot, when SDD-TMPC is used and no external disturbances exist.

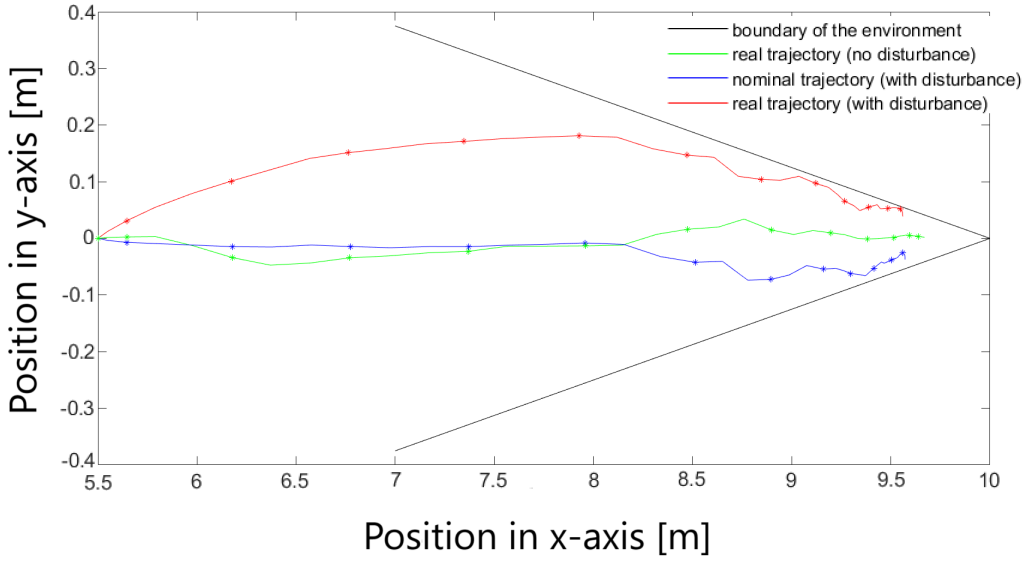


Figure 2.8: Scenario 3: Comparison of the trajectories of the position of the robot using SDD-TMPC, when no external disturbances affect the robot (green curve), and when external disturbances push the robot upwards relative to its speed (the blue curve for the nominal trajectory and the orange curve for the realized trajectory). Note that the black lines represent the boundaries of the corridor.

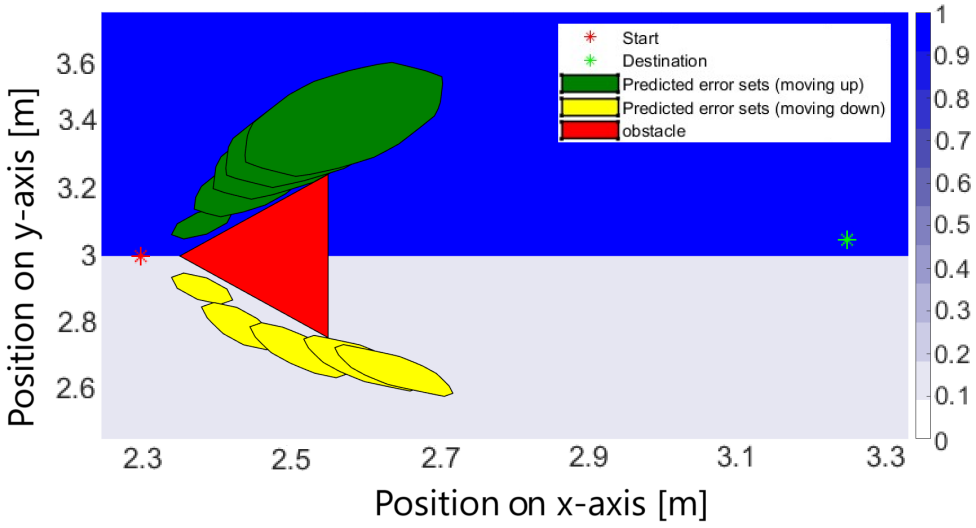


Figure 2.9: Scenario 4: The robot has to move from position $[2.3, 3]^\top$ to position $[3.25, 3.05]^\top$, while avoiding an obstacle (illustrated by the triangular red shape). The trajectories of the position projections of the tubes for the two cases are shown in green (above the obstacle) and yellow (below the obstacle) colors.

initial states of the leader and follower robots are, respectively, $[0, 0, \frac{\pi}{3}]$ and $[0.4, -0.2, -\frac{\pi}{2}]^T$. The performance of SDD-TMPC and a nonlinear TMPC are compared (see Appendix A2 for the formulation) for steering the follower robot. Since the control framework is discrete time, Matlab Ode45 [39] is used to determine the evolved states per discrete time step, using the system of differential equations (2.23) and (2.24). To solve one optimization step, TMPC needs 0.2s, while SDD-TMPC needs about 20s. This is an improvement compared to the previous case study, where it takes 4 minutes for SDD-TMPC per optimization iteration for a twice smaller prediction horizon.

Figure 2.10 shows the path generated by these controllers. Both controllers are able to reach and follow the reference trajectory despite the disturbances. Figure 2.11 shows that the absolute value of the directional error is actually less than 0.6 rad, which confirms the assumption of small angles. Figure 2.12 shows the transient responses, where SDD-TMPC reaches the desired position in 10% less time compared to nonlinear TMPC. From Figure 2.13, once both controllers achieve the steady-state, their performance is nearly equivalent. This is because SDD-TMPC is authorized to employ substantially larger nominal inputs, as shown in Figure 2.14. Even though SDD-TMPC returns significantly larger inputs (above 30%), especially in the early stages, the actual input constraints are never violated (Figure 2.15) and, compared to nonlinear TMPC, the actual input is usually 20% larger, during the first 3s of the mission, compared to the input of TMPC. This is particularly intriguing because the nominal

input of SDD-TMPC violates the original constraints $\mathbb{U}$. Nonetheless, SDD-TMPC anticipates the behavior of the ancillary controller and allows it to marginally violate the constraints, since the actual control input does not breach them. These results are noteworthy, since in [32], nonlinear TMPC is compared to another RMPC, where nonlinear TMPC exhibits an improved steady-state performance, but has a larger transient time. With SDD-TMPC, however, the large rise time is successfully eliminated, whereas it maintains an equivalent steady-state performance.

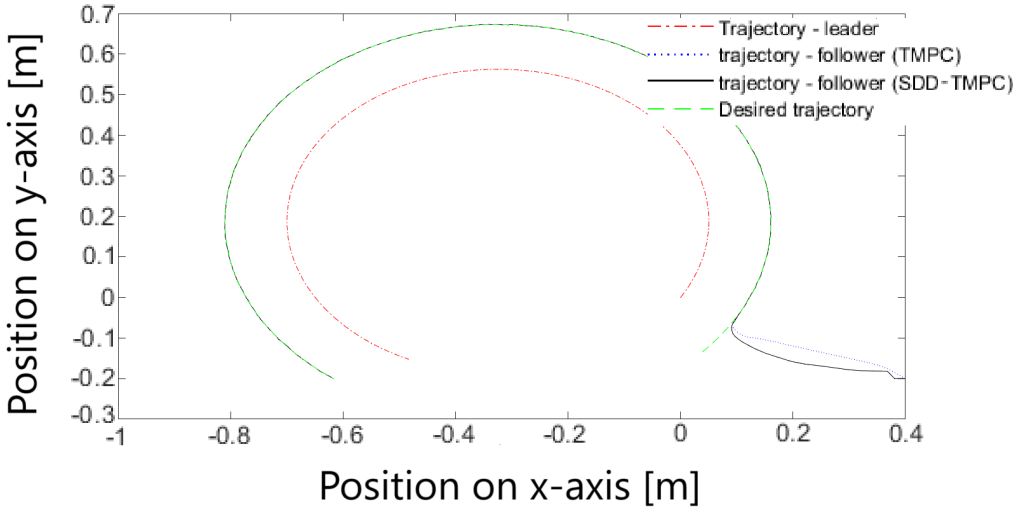


Figure 2.10: The leader trajectory, the desired trajectory for the follower, and the realized trajectories when the follower is steered via SDD-TMPC and via regular TMPC.

2.6. CONCLUSION AND FUTURE WORK

A dynamic version of TMPC, called SDD-TMPC, is proposed that uses a disturbance model (f^{dis}) to learn the dynamics of state-dependent external disturbances and to incorporate these dynamics into its future decisions. A FIS is used as an example of f^{dis}. It is generated offline based on a historical dataset. In the future, expert knowledge can be used to derive the rules, and a RL approach can be integrated into the FIS for online tuning. I also show an alternative approach by deriving state-dependent bounds by comparing real and simplified models in Appendix A3.

The stability of SDD-TMPC is also proven. The performance of SDD-TMPC is compared to regular MPC and TMPC for steering an autonomous robot in various scenarios that include obstacles and external disturbances. SDD-TMPC and TMPC show robustness to state-dependent disturbances, whereas SDD-TMPC compromises the optimality less than

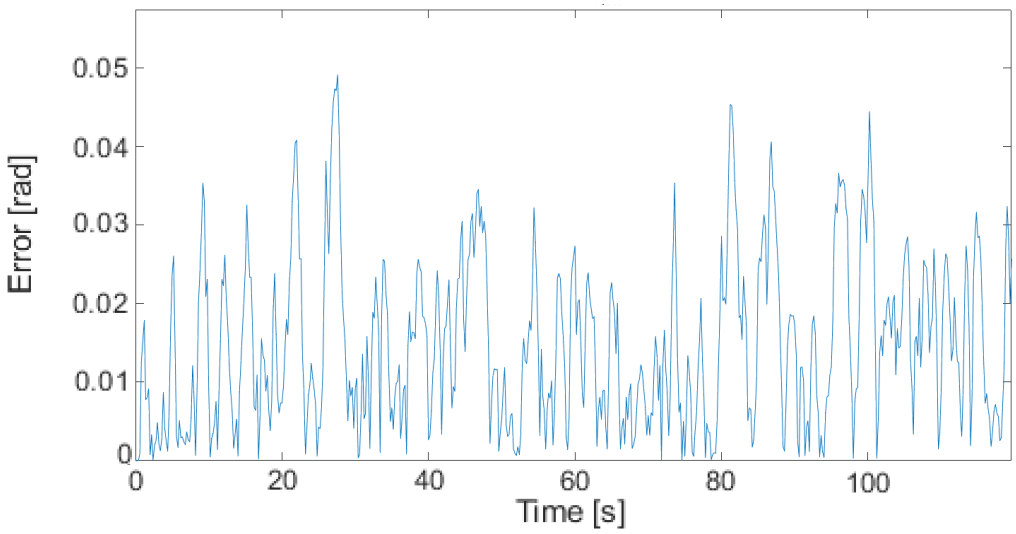


Figure 2.11: Directional error (i.e., difference between the nominal and actual heading for SDD-TMPC.)

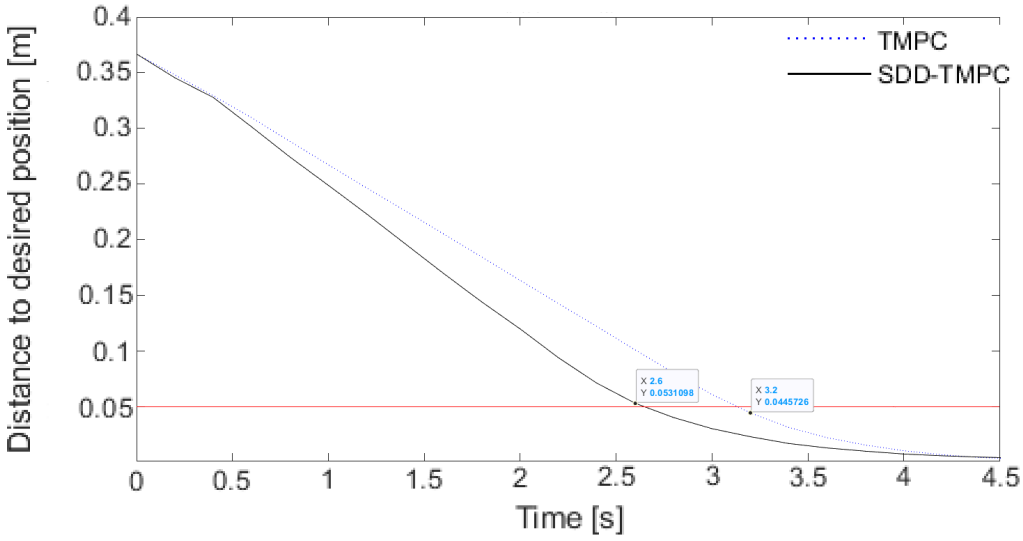


Figure 2.12: Distance between the desired and the transient positions using SDD-TMPC and TMPC.

TMPC. Thus, SDD-TMPC can reach states that are inaccessible for TMPC, resulting in reduced mission time (e.g., via a larger velocity or moving closer to the obstacles). In Table 2.6, a summarized comparison between TMPC and SDD-TMPC is included, based on

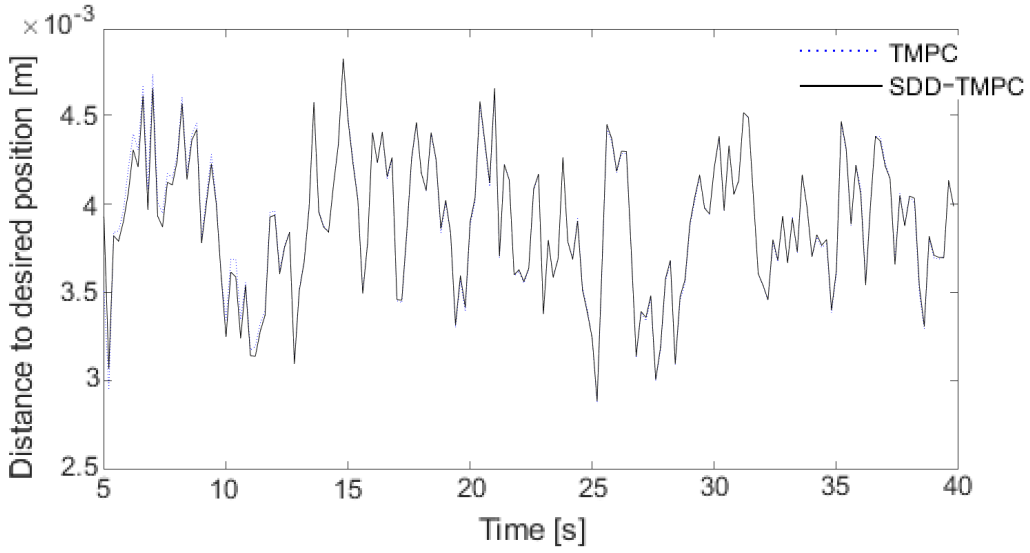


Figure 2.13: Difference between the desired and the steady-state positions using SDD-TMPC and TMPC.

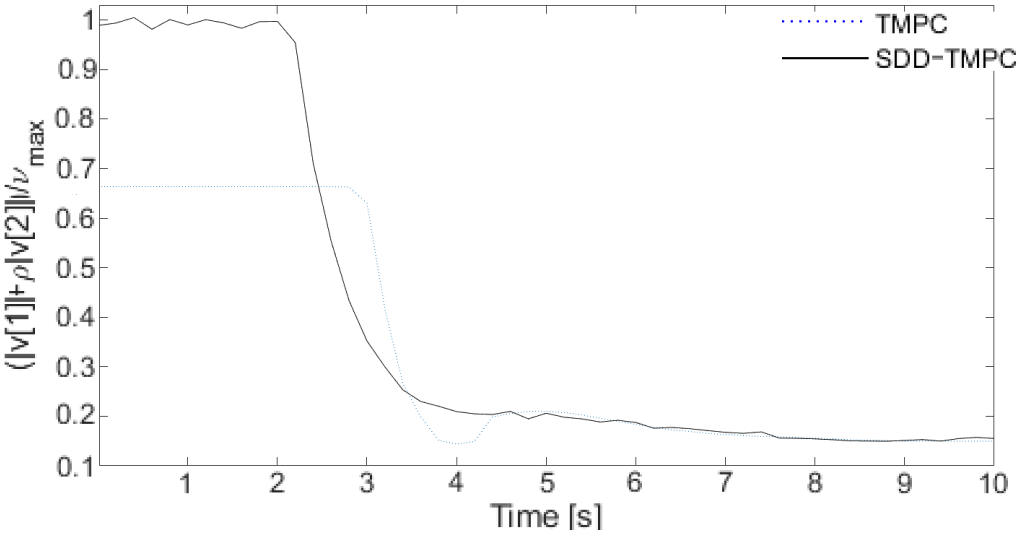


Figure 2.14: Nominal control input over time for SDD-TMPC and for TMPC

the theoretical discussions and results of the numerical experiments given in this paper.

A main challenge of SDD-TMPC is the time required to solve online the optimization problem, which does not scale well with the number of states. Moreover, if polytopes are used

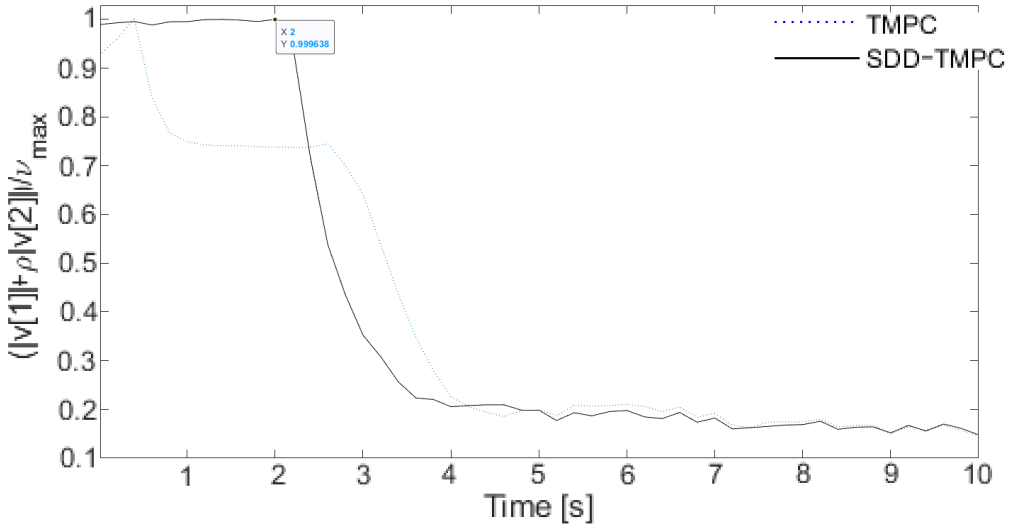


Figure 2.15: Control input over time for SDD-TMPC and for TMPC.

Table 2.6: Comparison between SDD-TMPC and TMPC based on the results of the case studies

Criteria	Optimality	Computation time	Robustness
Outperforming controller	SDD-TMPC (reducing the final value of the cost function by up to 33% in case study 1 and reducing the time to reach steady-state by 18% in case study 2).	TMPC (with TMPC operating in the range of milliseconds and SDD-TMPC in the range of a few minutes for Case study 1, whereas this computation time has already been reduced to seconds for SDD-TMPC in Case study 2)	Both controllers are the same

to describe the error sets, the complexity of the optimization problem does not scale well with the size of the prediction horizon. In Chapter 3, the online computation time of SDD-TMPC is improved by providing approximate versions for SDD-TMPC, e.g., by learning the nonlinear control policy using a NN, as has been done for MPC in [40] and for TMPC in [41]. The main research challenges for such an approximation will involve the proper choice of the learning method and enhancing the guarantees of the trained system for satisfying the hard constraints and ensuring stability. An alternative way to reduce required computational resources would be to divide the optimization problem of SDD-TMPC into sub-problems. One way to do this will be shown in Chapter 4. Another interesting topic concerns comparing SDD-TMPC and a similar controller that includes the dynamic tube within the cost, similarly to [19].

An additional interesting topic for future research is to replace the FIS with other approximators, including state-of-the-art state estimators for estimating an augmented state

vector that includes the external disturbances. This allows to implement SDD-TMPC for various real-life applications, including systems for which no reliable intuitive knowledge is available, but such state estimators already exist.

Finally, SDD-TMPC will be implemented and assessed for large-scale SaR scenarios.

REFERENCES

[1] F. Surma and A. Jamshidnejad. “State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances”. In: *International Journal of Robust and Nonlinear Control* 35 (2025). DOI: 10.1002/rnc.7558.

[2] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, 2017. ISBN: 9780975937754.

[3] M. Forbes, R. Patwardhan, H. Hamadah, and B. Gopaluni. “Model Predictive Control in Industry: Challenges and Opportunities”. In: *IFAC-PapersOnLine* 48 (2015). DOI: 10.1016/j.ifacol.2015.09.022.

[4] Y. Liu and G. Nejat. “Robotic Urban Search and Rescue: A Survey from the Control Perspective”. In: *Journal of Intelligent & Robotic Systems* 72 (2013). DOI: 10.1007/s10846-013-9822-x.

[5] J. P. Queralta *et al.* “Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision”. In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.3030190.

[6] S. Raković and D. Mayne. “A SIMPLE TUBE CONTROLLER FOR EFFICIENT ROBUST MODEL PREDICTIVE CONTROL OF CONSTRAINED LINEAR DISCRETE TIME SYSTEMS SUBJECT TO BOUNDED DISTURBANCES”. In: *IFAC Proceedings Volumes* 38 (2005). DOI: 10.3182/20050703-6-CZ-1902.00440.

[7] B. Kosko. “Fuzzy systems as universal approximators”. In: *IEEE Transactions on Computers* 43 (1994). DOI: 10.1109/12.324566.

[8] L.-X. Wang and J. Mendel. “Back-propagation fuzzy system as nonlinear dynamic system identifiers”. In: *IEEE International Conference on Fuzzy Systems*. 1992, pp. 1409–1418. DOI: 10.1109/FUZZY.1992.258711.

[9] L. Wang and J. Mendel. “Generating fuzzy rules by learning from examples”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 22 (1992). DOI: 10.1109/21.199466.

[10] F. Herrera, M. Lozano, and J. Verdegay. “Tuning fuzzy logic controllers by genetic algorithms”. In: *International Journal of Approximate Reasoning* 12 (1995). DOI: 10.1016/0888-613X(94)00033-Y.

[11] F. Fathinezhad, V. Derhami, and M. Rezaeian. “Supervised fuzzy reinforcement learning for robot navigation”. In: *Applied Soft Computing* 40 (2016). DOI: 10.1016/j.asoc.2015.11.030.

[12] C. Zhou and Q. Meng. “Dynamic balance of a biped robot using fuzzy reinforcement learning agents”. In: *Fuzzy Sets and Systems* 134 (2003), pp. 169–187.

[13] M. Goharimanesh, A. Mehrkish, and F. Janabi-Sharifi. “A Fuzzy Reinforcement Learning Approach for Continuum Robot Control”. In: *Journal of Intelligent & Robotic Systems* 100 (2020). DOI: 10.1007/s10846-020-01237-6.

[14] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi. “Tube-based robust nonlinear model predictive control”. In: *International Journal of Robust and Nonlinear Control* 21 (2011). DOI: 10.1002/rnc.1758.

[15] B. T. Lopez, J.-J. E. Slotine, and J. P. How. “Dynamic Tube MPC for Nonlinear Systems”. In: *2019 American Control Conference*. IEEE, 2019, pp. 2534–2549. DOI: 10.23919/ACC.2019.8814758.

[16] R. Gonzalez, M. Fiacchini, T. Alamo, J. Guzman, and F. Rodriguez. “Online robust tube-based MPC for time-varying systems: a practical approach”. In: *International Journal of Control* 84 (2011). DOI: 10.1080/00207179.2011.594093.

[17] S. V. Raković, B. Kouvaritakis, M. Cannon, and C. Panos. “Fully parameterized tube model predictive control”. In: *International Journal of Robust and Nonlinear Control* 22 (2012). DOI: 10.1002/rnc.2825.

[18] A. Parsi, A. Iannelli, and R. S. Smith. “Scalable tube model predictive control of uncertain linear systems using ellipsoidal sets”. In: *International Journal of Robust and Nonlinear Control* 119 (2022). DOI: 10.1002/rnc.6485.

[19] D. Fan, A. Agha, and E. Theodorou. “Deep Learning Tubes for Tube MPC”. In: *Robotics Science and Systems*. Robotics: Science and Systems Foundation, 2020. DOI: 10.15607/RSS.2020.XVI.087.

[20] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer. “Learning-Based Robust Model Predictive Control with State-Dependent Uncertainty”. In: *6th IFAC Conference on Nonlinear Model Predictive Control* 51 (2018). DOI: 10.1016/j.ifacol.2018.11.052.

[21] A. D. Bonzanini, D. B. Graves, and A. Mesbah. “Learning-Based SMPC for Reference Tracking Under State-Dependent Uncertainty: An Application to Atmospheric Pressure Plasma Jets for Plasma Medicine”. In: *IEEE Transactions on Control Systems Technology* 30 (2022). DOI: 10.1109/TCST.2021.3069825.

[22] G. Pin, D. M. Raimondo, L. Magni, and T. Parisini. “Robust Model Predictive Control of Nonlinear Systems With Bounded and State-Dependent Uncertainties”. In: *IEEE Transactions on Automatic Control* 54 (2009). DOI: 10.1109/TAC.2009.2020641.

[23] D. Malyuta, B. Açikmeşe, and M. Cacan. “Robust Model Predictive Control for Linear Systems with State and Input Dependent Uncertainties”. In: *American Control Conference*. IEEE, 2019, pp. 1145–1151. DOI: 10.23919/ACC.2019.8815343.

[24] P. Falugi and D. Q. Mayne. “Getting Robustness Against Unstructured Uncertainty: A Tube-Based MPC Approach”. In: *IEEE Transactions on Automatic Control* 59 (2014). DOI: 10.1109/TAC.2013.2287727.

[25] J. Köhler, M. A. Müller, and F. Allgöwer. “A novel constraint tightening approach for nonlinear robust model predictive control”. In: *Annual American Control Conference*. IEEE, 2018, pp. 728–734. DOI: 10.23919/ACC.2018.8431892.

[26] H. Schlüter and F. Allgöwer. “A Constraint-Tightening Approach to Nonlinear Stochastic Model Predictive Control under General Bounded Disturbances”. In: *21st IFAC World Congress* 53 (2020). DOI: 10.1016/j.ifacol.2020.12.518.

[27] S. Rakovic, E. Kerrigan, K. Kouramas, and D. Mayne. “Invariant approximations of the minimal robust positively Invariant set”. In: *IEEE Transactions on Automatic Control* 50 (2005). DOI: 10.1109/TAC.2005.843854.

[28] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. “A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles”. In: *Vehicle System Dynamics, International Journal of Vehicle Mechanics and Mobility* 52 (2014). DOI: 10.1080/00423114.2014.902537.

[29] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. IEEE, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[30] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. The MIT Press, 2019. ISBN: 0262039427.

[31] F. Surma. *Implementation of State-Dependent dynamic tube model predective control*. 2023. DOI: 10.4121/82150c4b-eea2-46f4-8a47-fcb6bf3d8e3d.v1.

[32] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. “Robust MPC for tracking constrained unicycle robots with additive disturbances”. In: *Automatica* 90 (2018). DOI: 10.1016/j.automatica.2017.12.048.

[33] F. Surma. *Application of SDDTMPC to control a unicycle*. 2023. DOI: 10.4121/3f1b28ee-2eca-4bac-bc2e-5ae2d2db4a5f.

[34] *Solve discrete-time Lyapunov equations*. URL: https://nl.mathworks.com/help/control/ref/dlyap.html (visited on 10/19/2023).

[35] T. J. Ross. *Fuzzy Logic with Engineering Applications*. 4th. Wiley, 2016. ISBN: 978-1-119-23586-6.

[36] *CLOCS2: Your One-Stop-Shop Solution for Optimization Based Control in Matlab/Simulink*. URL: http://www.ee.ic.ac.uk/ICLOCS/Overview.html (visited on 10/19/2023).

[37] *Continuous-Discrete Conversion Methods*. URL: https://nl.mathworks.com/help/ident/ug/continuous-discrete-conversion-methods.html#bs78nig-2 (visited on 09/22/2023).

[38] P. Gonçalves, P. Torres, C. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. “The e-puck, a Robot Designed for Education in Engineering”. In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* 1 (2009). URL: https://e-puck.gctronic.com/index.php?Itemid=36&id=34&option=com_content&view=article.

[39] *Solve nonstiff differential equations — medium order method*. URL: https://www.mathworks.com/help/matlab/ref/ode45.html (visited on 10/19/2023).

[40] Y. Cao and R. B. Gopaluni. “Deep Neural Network Approximation of Nonlinear Model Predictive Control”. In: *21st IFAC World Congress* 53 (2020), pp. 11319–11324. DOI: 10.1016/j.ifacol.2020.12.538.

[41] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How. "Demonstration-Efficient Guided Policy Search via Imitation of Robust Tubce MPC". In: *International Conference on Robotics and Automation*. IEEE, 2022, pp. 462–468. DOI: 10.1109/ICRA46639.2022.9812122.

[42] H. E. T. Yiqi Gao Andrew Gray and F. Borrelli. "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles". In: *Vehicle System Dynamics* 52 (2014). DOI: 10.1080/00423114.2014.902537.

APPENDIX A1: PARAMETERS/VALUES USED IN THE CASE STUDIES

This appendix includes all the parameters and values that have been used in the case studies. In particular, for case study 1, there is:

- Discretization sampling time: $T^{\mathrm{s}} = 0.1$ s
- Prediction horizon: $N = 5$
- Cost matrices used in (2.11): $Q = \mathrm{diag}(100, 100, 1, 1), \quad R = I_{2\times 2}$,
$$F = \begin{bmatrix} 805 & 0 & -571 & 0 \\ 0 & 805 & 0 & -571 \\ -571 & 0 & 655 & 0 \\ 0 & -571 & 0 & 655 \end{bmatrix}$$
- Gain matrix used in (2.12): $K = \begin{bmatrix} 7{,}98 & 0 & 4.42 & 0 \\ 0 & 7{,}98 & 0 & 4.42 \end{bmatrix}$
- Matrices used to determine matrix K in (2.12) via LQR: $Q_K = \mathrm{diag}(100, 100, 0.1, 0.1), \quad R_K = I_{2\times 2}$
- Matrices Q_κ and R_κ used in (2.17): $Q_\kappa = \mathrm{diag}(10, 10, 1, 1), \quad R_\kappa = I_{2\times 1}$
- Ground slipperiness coefficient: $\beta = 1$ (unless otherwise states)

Different cost matrices are used for:

- ancillary control law to more aggressively minimize the position state, which more strongly reduces the influence of external position disturbances in then the original cost matrices.
- terminal control law to reduce the size of the terminal set.

For case study 2, the following values are used:

- Discretization sampling time: $T^{\mathrm{s}} = 0.2$ s
- Prediction horizon: $N = 10$

- Maximum value of the velocity of the robot wheels: $v^{\max} = 0.13$ m/s
- Maximum value of the position disturbance vector: $\eta = 0.004$ m
- Radius of the leader and follower robots: $\rho = 0.0267$ m
- velocity of the leader robot: $v^{\mathrm{R}} = 0.015\frac{\mathrm{m}}{\mathrm{s}}$, $\omega^{\mathrm{R}} = 0.04\frac{\mathrm{rad}}{\mathrm{s}}$
- Cost matrices used in (2.32a): Q=diag(0.2,0.2,0), R=diag(0.4,0.4), F=diag(0.5,0.5)
- Gain used in (2.26): K^{e}=2.3
- Gain used in (2.33): $K^{\mathrm{e,d}}$=0.63
- Gain used in (2.37): $\bar{K} = 0.12$
- $\mathbb{E}^{\max} = \{\boldsymbol{e} \in \mathbb{R}^3 | \max(|\boldsymbol{e}[1:2]|) \leq 0.0022\}$
- $\mathbb{V} = 0.6636\mathbb{U}$
- $\mathbb{Z}^{\mathrm{f}} = \{\boldsymbol{z}^{\mathrm{r}} \in \mathbb{R}^3 | |z^{\mathrm{r}}[1]| + |z^{\mathrm{r}}[2]| \leq 0.542\}$.

APPENDIX A2: NONLINEAR TMPC FORMULATION FOR CASE STUDY 2

To control the motion of the follower robot, TMPC solves the following problem per iteration:

$$V^*(\boldsymbol{x}_k) = \min_{\tilde{\boldsymbol{z}}_k, \tilde{\boldsymbol{v}}_k} \sum_{i=k}^{k+N-1} \left(\left\| \boldsymbol{z}^{\mathrm{r}}_{i|k} \right\|^2_Q + \left\| \boldsymbol{v}^{\mathrm{r}}_{i|k} \right\|^2_R \right) + \left\| \boldsymbol{z}^{\mathrm{r}}_{k+N|k} \right\|^2_F \tag{2.32a}$$

$$\text{s.t. for } i = k+1, \ldots, k+N\text{:}$$

$$\boldsymbol{x}_k \in \{\boldsymbol{z}_k\} \oplus \mathbb{E}^{\max} \tag{2.32b}$$

$$\boldsymbol{z}_{i+1|k} = f^{\mathrm{d}}(\boldsymbol{z}_{i|k}, \boldsymbol{v}_{i|k}) \tag{2.32c}$$

$$\boldsymbol{v}_{i|k} \in \mathbb{V}, \quad \boldsymbol{z}^{\mathrm{r}}_{i+N|k} \in \mathbb{Z}^{\mathrm{f}} \tag{2.32d}$$

$$\boldsymbol{z}^{r}_{i|k}[1,2] = \mathrm{Rot}(-\boldsymbol{z}_{i|k}[3])(\boldsymbol{x}^{\mathrm{R}}_i[1,2] - \boldsymbol{z}_{i|k}[1:2]) + \mathrm{Rot}(\boldsymbol{z}^{\mathrm{r}}_{i|k}[3])\boldsymbol{p}_{\mathrm{d}} \tag{2.32e}$$

$$\boldsymbol{z}^{\mathrm{r}}_{i|k}[3] = \boldsymbol{x}^{\mathrm{R}}_i[3] - \boldsymbol{z}_{i|k}[3] \tag{2.32f}$$

$$\boldsymbol{v}^{\mathrm{r}}_{i|k} = \begin{bmatrix} -\boldsymbol{v}_{i|k}[1] + (v_{\mathrm{R}} - p^{\mathrm{d}}_x \omega^{\mathrm{R}}) \cos(\boldsymbol{z}^{\mathrm{r}}_{i|k}[3]) - p^{\mathrm{d}}_x \omega^{\mathrm{R}} \sin(\boldsymbol{z}^{\mathrm{r}}_{i|k}[3]) \\ -\rho\, \boldsymbol{v}_{i|k}[2] + (v_{\mathrm{R}} - p^{\mathrm{d}}_x \omega^{\mathrm{R}}) \sin(\boldsymbol{z}^{\mathrm{r}}_{i|k}[3]) - p^{\mathrm{d}}_y \omega^{\mathrm{R}} \cos(\boldsymbol{z}^{\mathrm{r}}_{i|k}[3]) \end{bmatrix} \tag{2.32g}$$

where (2.32b) states that the initial nominal state $\boldsymbol{z}_k$ of the follower robot, which is a decision variable, should be determined such that the error between the measured state $\boldsymbol{x}_k$ and the nominal state $\boldsymbol{z}_k$ of the follower robot belongs to set $\mathbb{E}^{\max}$. Moreover, according to (2.32c), the nominal states of the follower robot evolve according to the discretized function $f^{\mathrm{d}}(\cdot)$,

which is determined after time-discretization of (2.24) and by putting $\boldsymbol{w}_k = 0$ for all time steps. Constraint (2.32d) enforces all the calculated nominal inputs to fall within set $\mathbb{V}$, which should be constructed, such that if $\boldsymbol{v}_{i|k} \in \mathbb{V}$, then for the actual input $\boldsymbol{u}_{i|k} \in \mathbb{U}$ for a given ancillary control law. Finally, $\mathbb{Z}^{\mathrm{f}}$ is the terminal set of the states for the follower robot. The cost function $V(\cdot)$ is computed using $\boldsymbol{z}^{\mathrm{r}}_{i|k}$, which is the nominal state of the follower robot within the coordinate system of the leader robot, while the original nominal state $\boldsymbol{z}_{i|k}$ is given in the global coordinates. The transition from the global coordinate system to the local coordinate system of the leader robot is done via (2.32e)-(2.32g), where $\mathrm{Rot}(\cdot)$ is a function that returns a 2-dimensional rotation matrix with respect to the global coordinates, given an input angle. Whenever square brackets are used after a vector, if the bracket contains one scalar, the notation refers to the element corresponding to that scalar of the vector. In case the square brackets include 'scalar 1:scalar 2', the notation refers to elements 'scalar 1' to 'scalar 2' (including these elements) of the vector.

For the nonlinear ancillary control law given at time instant t via (2.25), the dynamics of the position error for the controlled system is described by the first equation in (2.26). After discretization, the dynamics of the position error for time step $i \in \{k, \ldots, k+N-1\}$ is given by:

$$\boldsymbol{e}_{i+1}[1:2] = K^{\mathrm{e,d}}\boldsymbol{e}_i[1:2] + T^{\mathrm{s}}[w_x, w_y]^T \tag{2.33}$$

where T^{s} represents the sampling time and $K^{\mathrm{e,d}}$ is calculated based on the zero-order hold approach [37], to ensure equivalence between the continuous-time and discrete-time formulations.

The cross section $\mathbb{E}^{\max}$ of the tube for nonlinear TMPC, set $\mathbb{V}$ of admissible values for the nominal control inputs, and the terminal control law (i.e., the control law that is implemented beyond the prediction horizon) are defined by:

$$\mathbb{E}^{\max} = \left\{ \boldsymbol{e} \in \mathbb{R}^3 \middle| \max(|\boldsymbol{e}[1:2]|) < \frac{T^{\mathrm{s}}\eta}{1-K^{\mathrm{e,d}}} \right\} \tag{2.34}$$

$$\mathbb{V} = \left\{ \boldsymbol{v} \in \mathbb{R}^2 \middle| \left[\frac{1}{v^{\max}}, \frac{\rho}{v^{\max}} \right] |\boldsymbol{v}| < \frac{\sqrt{2}}{2} - \frac{\eta\sqrt{2}}{v^{\max}} \right\} \tag{2.35}$$

$$\boldsymbol{u}_k^{\mathrm{T}} = \begin{bmatrix} \bar{K}\boldsymbol{z}^{\mathrm{r}}_{k+N}[1] + (v^{\mathrm{R}} - p^{\mathrm{d}}\omega^{\mathrm{R}})\cos(\boldsymbol{z}^{\mathrm{r}}_{k+N}[3]) - p^{\mathrm{d}}_x\omega^{\mathrm{R}}\sin(\boldsymbol{z}^{\mathrm{r}}_{k+N}[3]) \\ \frac{1}{\rho}(\bar{k}\boldsymbol{z}^{\mathrm{r}}_{k+N}[2] + (v^{\mathrm{R}} - p^{\mathrm{d}}_x\omega^{R})\sin(\boldsymbol{z}^{\mathrm{r}}_{k+N}[3]) - p^{\mathrm{d}}_y\omega^{R}\cos(\boldsymbol{z}^{\mathrm{r}}_{k+N}[3])) \end{bmatrix} \tag{2.36}$$

where $\bar{K}$ is a constant. Finally, the terminal admissible set for the position of the follower robots is given by:

$$\mathbb{Z}^{\mathrm{f}} =: \left\{ \boldsymbol{z}^{\mathrm{r}} \in \mathbb{R}^3 \middle| \bar{K}(|\boldsymbol{z}^{\mathrm{r}}[1]| + |\boldsymbol{z}^{\mathrm{r}}[2]|) < \frac{v^{\max}\sqrt{2}}{2} - \eta\sqrt{2} - \sqrt{2}\left\| \begin{bmatrix} 1 & -p^{\mathrm{d}}_x \\ 0 & p^{\mathrm{d}}_y \end{bmatrix} \right\| \right\} \tag{2.37}$$

APPENDIX A3: DIRECTIONAL ERROR DYNAMICS & LOWER AND UPPER BOUNDS

In case study 2, for SDD-TMPC the error sets are approximated via box sets, which may slightly increase the conservatism, but significantly reduce the computation time. Let $\mathbb{E}_i[\ell]$ denote the projection of the error set $\mathbb{E}_i$ onto the ℓ^{th} dimension of the state space (thus for this case study $\ell = 1,2,3$). Since these are box sets, $\mathbb{E}_i$ is the cross product over the projections of this set on all the dimensions of the state space. Based on (2.33), for $\mathbb{W} = \{w \in \mathbb{R} | w \leq T^{\text{s}}\eta\}$ and $\ell = 1,2$, for the projection of the error set the following equality is true:

$$\mathbb{E}_{i+1}[\ell] = K^{\text{e,d}}\mathbb{E}_i[\ell] \oplus \mathbb{W} \tag{2.38}$$

To obtain the error dynamics for the third dimension of the state space (i.e., the robot direction), from the definition of this error and (2.24), there is:

$$\dot{\boldsymbol{e}}_t[3] = \dot{\boldsymbol{x}}_t[3] - \dot{\boldsymbol{z}}_t[3] = \boldsymbol{u}_t[2] - \boldsymbol{v}_t[2] \tag{2.39a}$$

which, together with (2.25), and after applying the trigonometric identities $\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$ and $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$, as well as substituting $\boldsymbol{x}_t[3] = \boldsymbol{z}_t[3] + \boldsymbol{e}_t[3]$, results in:

$$\dot{\boldsymbol{e}}_t[3] = -\frac{1}{\rho}\sin(\boldsymbol{e}_t[3])\boldsymbol{v}_t[1] + (\cos(\boldsymbol{e}_t[3]) - 1)\boldsymbol{v}_t[2] - \frac{K^{\text{e}}}{\rho}\sin(\boldsymbol{x}_t[3])\boldsymbol{e}_t[1] - \frac{K^{\text{e}}}{\rho}\cos(\boldsymbol{x}_t[3])\boldsymbol{e}_t[2] \tag{2.39b}$$

Note that (2.39b) is nonlinear and thus, computing all possible errors per time instant via (2.39b) is, in general, computationally demanding. Moreover, (2.39b) does not necessarily result in a polytopic error set per time instant. Thus, as explained in [42], the dynamic equation in (2.39b) is rewritten as a linear evolutionary equation for $\boldsymbol{e}_t[3]$, where the nonlinear terms are treated as state-dependent disturbances, i.e.:

$$\dot{\boldsymbol{e}}_t[3] = -\frac{1}{\rho}\boldsymbol{e}_t[3]\boldsymbol{v}_t[1] + w_t^{\text{e}} \tag{2.39c}$$

where $w_t^{\text{e}} = \sum_{i=1}^{4} w_t^{\text{e},i}$, with $w_t^{\text{e},1} = -\frac{1}{\rho}\left(\sin(\boldsymbol{e}_t[3]) - \boldsymbol{e}_t[3]\right)\boldsymbol{v}_t[1]$ and $w_t^{\text{e},2} = \left(\cos(\boldsymbol{e}_t[3]) - 1\right)\boldsymbol{v}_t[2]$ and $w_t^{\text{e},3} = -\frac{K^{\text{e}}}{\rho}\sin(\boldsymbol{x}_t[3])\boldsymbol{e}_t[1]$ and $w_t^{\text{e},4} = -\frac{K^{\text{e}}}{\rho}\cos(\boldsymbol{x}_t[3])\boldsymbol{e}_t[2]$. This approach has proven to be accurate for small deviations in the third dimension of the state space (in this case the heading/direction of the robot). Henceforth, it is assumed that the absolute values of $\boldsymbol{e}_t[3]$ remain relatively low (i.e., $|\boldsymbol{e}_t[3]| < \pi/4$).

For determining the lower and upper bounds of the external disturbances, a lower bound and an upper bound per term $w_t^{\text{e},i}$ for $i = 1,2,3,4$ are found. Since $\boldsymbol{e}_t[3] < \pi/4$, it can be written:

$$\begin{aligned} w_t^{\text{e},1,\min} &= -\tfrac{1}{\rho}\Big(\sin(\min(\mathbb{E}_t[3])) - \min(\mathbb{E}_t[3])\Big)\boldsymbol{v}_t[1] \\ w_t^{\text{e},1,\max} &= -\tfrac{1}{\rho}\Big(\sin(\max(\mathbb{E}_t[3])) - \max(\mathbb{E}_t[3])\Big)\boldsymbol{v}_t[1] \end{aligned} \qquad \text{for } \boldsymbol{v}_t[1] \geq 0 \tag{2.40a}$$

$$\begin{aligned} w_t^{\mathrm{e},1,\min} &= -\tfrac{1}{\rho}\Big(\sin(\max(\mathbb{E}_t[3])) - \max(\mathbb{E}_t[3])\Big)\boldsymbol{v}_t[1] \\ w_t^{\mathrm{e},1,\max} &= -\tfrac{1}{\rho}\Big(\sin(\min(\mathbb{E}_t[3])) - \min(\mathbb{E}_t[3])\Big)\boldsymbol{v}_t[1] \end{aligned} \quad \text{for } \boldsymbol{v}_t[1] \le 0 \tag{2.40b}$$

I can bound $w_t^{\mathrm{e},2}$ considering that $\cos(\boldsymbol{e}_t[3]) - 1$ is a descending function for the given range of $\boldsymbol{e}_t[3]$:

$$\begin{aligned} w_t^{\mathrm{e},2,\min} &= \Big(\cos(|\max(\mathbb{E}_t[3])|) - 1\Big)\max(\boldsymbol{v}_t[2], 0) \\ w_t^{\mathrm{e},2,\max} &= \Big(\cos(|\min(\mathbb{E}_t[3])|) - 1\Big)\min(\boldsymbol{v}_t[2], 0) \end{aligned} \tag{2.40c}$$

Finally, for $w_t^{\mathrm{e},3}$ and $w_t^{\mathrm{e},4}$ it can be written:

$$\begin{aligned} w_t^{\mathrm{e},3,\min} &= \min_{(\boldsymbol{e}_t[1],\boldsymbol{e}_t[3])\in\mathbb{E}_t[1]\times\mathbb{E}_t[3]} \Big(-\frac{K^{\mathrm{e}}}{\rho}\sin(\boldsymbol{z}_t[3]+\boldsymbol{e}_t[3])\boldsymbol{e}_t[1]\Big) \\ w_t^{\mathrm{e},3,\max} &= \max_{(\boldsymbol{e}_t[1],\boldsymbol{e}_t[3])\in\mathbb{E}_t[1]\times\mathbb{E}_t[3]} \Big(-\frac{K^{\mathrm{e}}}{\rho}\sin(\boldsymbol{z}_t[3]+\boldsymbol{e}_t[3])\boldsymbol{e}_t[1]\Big) \end{aligned} \tag{2.40d}$$

$$\begin{aligned} w_t^{\mathrm{e},4,\min} &= \min_{(\boldsymbol{e}_t[2],\boldsymbol{e}_t[3])\in\mathbb{E}_t[2]\times\mathbb{E}_t[3]} \Big(-\frac{K^{\mathrm{e}}}{\rho}\cos(\boldsymbol{z}_t[3]+\boldsymbol{e}_t[3])\boldsymbol{e}_t[2]\Big) \\ w_t^{\mathrm{e},4,\max} &= \max_{(\boldsymbol{e}_t[2],\boldsymbol{e}_t[3])\in\mathbb{E}_t[2]\times\mathbb{E}_t[3]} \Big(-\frac{K^{\mathrm{e}}}{\rho}\cos(\boldsymbol{z}_t[3]+\boldsymbol{e}_t[3])\boldsymbol{e}_t[2]\Big) \end{aligned} \tag{2.40e}$$

Therefore, the admissible set of the external disturbances for (2.39c) is defined via:

$$\mathbb{W}_t[3] := \Big\{ w_t^{\mathrm{e}} \in \mathbb{R} \Big| \sum_{i=1}^{4} w_t^{\mathrm{e},i,\min} \le w_t^{\mathrm{e}} \le \sum_{i=1}^{4} w_t^{\mathrm{e},i,\min} \Big\} \tag{2.40f}$$

For the discrete-time framework of the problem, (2.40f) is estimated for the discrete time steps.

APPENDIX A4: INPUT CONSTRAINT TIGHTENING

Here, the approach that has been used to tighten the input constraints of SDD-TMPC online for case study 2 is explained. Considering the ancillary control law that is formulated by (2.25), imposing the hard constraint $\boldsymbol{u}_t \in \mathbb{U}$, using the equality $\boldsymbol{x}_t[3] = \boldsymbol{z}_t[3] + \boldsymbol{e}_t[3]$, multiplying both sides of the hard constraints by $\begin{bmatrix} \cos(\boldsymbol{e}_t[3]) & -\rho\sin(\boldsymbol{e}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{e}_t[3]) & \cos(\boldsymbol{e}_t[3]) \end{bmatrix}$ from left, and considering all possible combinations of $\boldsymbol{e}_t[1]$, $\boldsymbol{e}_t[2]$, and $\boldsymbol{e}_t[3]$, the following condition is obtained:

$$\begin{aligned} &\{\boldsymbol{v}_t\} \oplus \begin{bmatrix} -\cos(\boldsymbol{z}_t[3]) & -\sin(\boldsymbol{z}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{z}_t[3]) & -\frac{1}{\rho}\cos(\boldsymbol{z}_t[3]) \end{bmatrix} K^{\mathrm{e}}(\mathbb{E}_t[1]\times\mathbb{E}_t[2]) \subseteq \\ &\bigcap_{\boldsymbol{e}[3]\in\mathbb{E}_t[3]} \left(\begin{bmatrix} \cos(\boldsymbol{e}_t[3]) & -\rho\sin(\boldsymbol{e}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{e}_t[3]) & \cos(\boldsymbol{e}_t[3]) \end{bmatrix} \mathbb{U} \right) = \mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3]) \end{aligned} \tag{2.41}$$

Note that multiplication of a matrix by a set that contains vector elements means that the mapping corresponding to that matrix is implemented on each vector element of the matrix. On the left-hand side of (2.41), the Minkowski addition of the nominal control input and a linear transformation of the error set is given, whereas on the right-hand side, a nonlinear

transformation of the admissible set of control inputs, i.e., $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$, should be computed. The resulting set $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$ does not necessarily correspond to a polytope. Moreover, since set $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$ is time-varying and based on (2.41) is in general rotated per time instant, the exact computation of (2.41) per time instant is complex or may become computationally intractable.

Picking an arbitrary element of $\mathbb{U}$, e.g., $[u[1],u[2]]^T$, after the mapping $\begin{bmatrix} \cos(\boldsymbol{e}_t[3]) & -\rho\sin(\boldsymbol{e}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{e}_t[3]) & \cos(\boldsymbol{e}_t[3]) \end{bmatrix}$ is performed on this vector, a new vector is obtained $\left[\cos(\boldsymbol{e}_t[3])u[1]-\rho\sin(\boldsymbol{e}_t[3])u[2], \frac{1}{\rho}\sin(\boldsymbol{e}_t[3])u[1]+\cos(\boldsymbol{e}_t[3])u[2]\right]^T$, which belongs to the admissible set $\mathbb{U}$ of inputs in case based on (2.22), it satisfies the following condition:

$$\frac{\left|\cos(\boldsymbol{e}_t[3])u[1]-\rho\sin(\boldsymbol{e}_t[3])u[2]\right|}{v^{\max}}+\frac{\left|\sin(\boldsymbol{e}_t[3])u[1]+\rho\cos(\boldsymbol{e}_t[3])u[2]\right|}{v^{\max}}\leq 1$$

Therefore, in general for each $\boldsymbol{e}_t[3]\in\mathbb{E}_t[3]$ the right-hand side term of (2.41), i.e., $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$, can be defined by the following quadrangle:

$$\begin{bmatrix} \frac{\cos(\boldsymbol{e}_t[3])+\sin(\boldsymbol{e}_t[3])}{v^{\max}} & \frac{-\rho\sin(\boldsymbol{e}_t[3])+\rho\cos(\boldsymbol{e}_t[3])}{v^{\max}} \\ \frac{-\cos(\boldsymbol{e}_t[3])+\sin(\boldsymbol{e}_t[3])}{v^{\max}} & \frac{\rho\sin(\boldsymbol{e}_t[3])+\rho\cos(\boldsymbol{e}_t[3])}{v^{\max}} \\ \frac{-\cos(\boldsymbol{e}_t[3])-\sin(\boldsymbol{e}_t[3])}{v^{\max}} & \frac{\rho\sin(\boldsymbol{e}_t[3])-\rho\cos(\boldsymbol{e}_t[3])}{v^{\max}} \\ \frac{\cos(\boldsymbol{e}_t[3])-\sin(\boldsymbol{e}_t[3])}{v^{\max}} & \frac{-\rho\sin(\boldsymbol{e}_t[3])-\rho\cos(\boldsymbol{e}_t[3])}{v^{\max}} \end{bmatrix}\boldsymbol{u}_t\leq\begin{bmatrix}1\\1\\1\\1\end{bmatrix} \tag{2.42}$$

Figure 2.16 (see the red background quadrangle) illustrates an example of such a quadrangle, when $\boldsymbol{e}_t[3]=0.3473$. The intersection of all such quadrangles for the entire range of $\boldsymbol{e}_t[3]$, in this case when $-0.3473\leq\boldsymbol{e}_t[3]\leq 0.3473$, generates $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$ according to (2.41). This set is illustrated via the black middle-ground polygon in Figure 2.16. To tackle the computational complexity associated with (2.41) and to ensure that the resulting $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$ is always represented via a polytope, the quadrangles corresponding to $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$ (i.e., the red background quadrangle in Figure 2.16) are replaced with the largest subset of $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$, for which the diameters are parallel to the x and y axes (see the blue foreground quadrangle in Figure 2.16). In other words, $\mathbb{V}^{\mathrm{NL}}(\cdot)$ is replaced with the multiplication of the original set $\mathbb{U}$ and a positive, state-dependent scalar function $\lambda(\cdot)$, i.e.:

$$\mathbb{V}^{\mathrm{NL}}(\cdot)=\lambda(\cdot)\mathbb{U} \tag{2.43}$$

This approach is similar to the constraint tightening method used in [32], although there a constant value is used instead of a scalar function $\lambda(\cdot)$.

To determine this subset, $\boldsymbol{u}_t[1]$ is set to 0, which based on (2.42) results in:

$$\begin{bmatrix}\boldsymbol{u}_t[2]\\ \boldsymbol{u}_t[2]\\ -\boldsymbol{u}_t[2]\\ -\boldsymbol{u}_t[2]\end{bmatrix}\leq\begin{bmatrix}\frac{v^{\max}}{\rho(\cos(\boldsymbol{e}_t[3])-\sin(\boldsymbol{e}_t[3]))}\\ \frac{v^{\max}}{\rho(\cos(\boldsymbol{e}_t[3])+\sin(\boldsymbol{e}_t[3]))}\\ \frac{v^{\max}}{\rho(\cos(\boldsymbol{e}_t[3])-\sin(\boldsymbol{e}_t[3]))}\\ \frac{v^{\max}}{\rho(\cos(\boldsymbol{e}_t[3])+\sin(\boldsymbol{e}_t[3]))}\end{bmatrix} \tag{2.44}$$

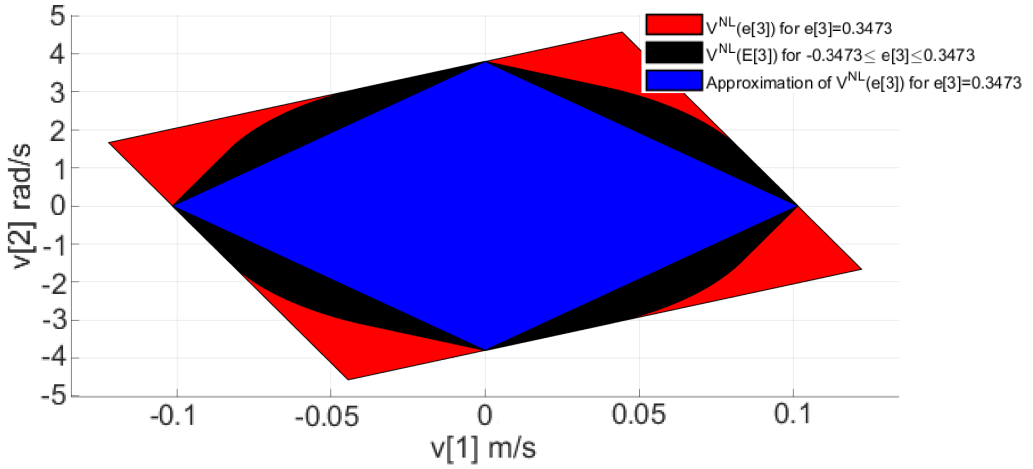


Figure 2.16: Illustration of an example quadrangle (the red background quadrangle) that represents $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$ when $\boldsymbol{e}_t[3] = 0.3473$. The intersection of all such quadrangles for the entire range of $\boldsymbol{e}_t[3]$, i.e., for $-0.3473 \leq \boldsymbol{e}_t[3] \leq 0.3473$, generates $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t[3])$ (the black middle-ground polygon). The simplified version of $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$, i.e., the largest subset of $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$ with its diameters parallel to the x and y axes, is shown via the blue foreground quadrangle.

Since $|\boldsymbol{e}_t[3]| < \pi/4$, both '$\cos(\boldsymbol{e}_t[3]) - \sin(\boldsymbol{e}_t[3])$' and '$\cos(\boldsymbol{e}_t[3]) + \sin(\boldsymbol{e}_t[3])$' result in positive values. Thus, (2.44) can be reformulated via:

$$|\boldsymbol{u}_t[2]| \leq \frac{v^{\max}}{\rho} \lambda\left(\boldsymbol{e}_t[3]\right) \tag{2.45}$$

where $\lambda(\cdot)$ is defined via:

$$\lambda(\cdot) = \frac{1}{\cos(\cdot) + |\sin(\cdot)|} \tag{2.46}$$

Similarly, if $\boldsymbol{u}_t[2]$ is set to 0, from (2.42) it is obtained:

$$\begin{bmatrix} \boldsymbol{u}_t[1] \\ \boldsymbol{u}_t[1] \\ -\boldsymbol{u}_t[1] \\ -\boldsymbol{u}_t[1] \end{bmatrix} \leq \begin{bmatrix} \frac{v^{\max}}{\cos(\boldsymbol{e}_t[3]) - \sin(\boldsymbol{e}_t[3])} \\ \frac{v^{\max}}{\cos(\boldsymbol{e}_t[3]) + \sin(\boldsymbol{e}_t[3])} \\ \frac{v^{\max}}{\cos(\boldsymbol{e}_t[3]) - \sin(\boldsymbol{e}_t[3])} \\ \frac{v^{\max}}{\cos(\boldsymbol{e}_t[3]) + \sin(\boldsymbol{e}_t[3])} \end{bmatrix} \tag{2.47}$$

which is equivalent to the following equation:

$$|\boldsymbol{u}_t[1]| \leq v^{\max} \lambda\left(\boldsymbol{e}_t[3]\right) \tag{2.48}$$

From (2.45) and (2.48), vectors $\lambda(\boldsymbol{e}_t[3]) \left[0, \frac{v^{\max}}{\rho}\right]^T$, $\lambda(\boldsymbol{e}_t[3]) \left[0, -\frac{v^{\max}}{\rho}\right]^T$, $\lambda(\boldsymbol{e}_t[3]) [v^{\max}, 0]^T$, and $\lambda(\boldsymbol{e}_t[3]) [-v^{\max}, 0]^T$ are obtained as the corners of the polytope that represents the largest

subset of $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3])$ (see, e.g., the blue foreground quadrangle in Figure 2.16). All other points of this polytope are obtained by a convex combination of these vectors. In this case, the original set $\mathbb{U}$ is simplified to a polytope with it corners given by $[v^{\max},0]^T$, $\left[0,-\frac{v^{\max}}{\rho}\right]^T$, $[-v^{\max},0]$, and $\left[0,\frac{v^{\max}}{\rho}\right]^T$, which satisfies $\mathbb{V}^{\mathrm{NL}}(\boldsymbol{e}_t[3]) = \lambda(\boldsymbol{e}_t[3])\mathbb{U}$, i.e., satisfies (2.43).

Figure 2.17 shows the curve corresponding to the function $\lambda(\cdot)$. From the expression of $\lambda(\cdot)$ given by (2.46) and as is shown in Figure 2.17, for negative values of $\boldsymbol{e}_t[3]$, $\lambda(\cdot)$ shows an ascending behavior, where the maximum of $\lambda(\cdot)$ is unity, which occurs for $\boldsymbol{e}_t[3] = 0$. For positive values of $\boldsymbol{e}_t[3]$, $\lambda(\cdot)$ shows a descending behavior. Moreover, $\lambda(\cdot)$ is an even function, which implies that its representative curve is symmetric with respect to the vertical axis (see Figure 2.17). Thus, for tightening the constraints online, from (2.41) and (2.43), the admissible set $\mathbb{V}(\mathbb{E}_t)$ is defined for the nominal control inputs as a function of the error set via:

$$\mathbb{V}(\mathbb{E}_t) := \min_{\boldsymbol{e}_t[3]\in\mathbb{E}_t[3]} (\lambda(\boldsymbol{e}_t[3])\mathbb{U} \ominus \begin{bmatrix} -\cos(\boldsymbol{z}_t[3]) & -\sin(\boldsymbol{z}_t[3]) \\ \frac{1}{\rho}\sin(\boldsymbol{z}_t[3]) & -\frac{1}{\rho}\cos(\boldsymbol{z}_t[3]) \end{bmatrix} K^{\mathrm{e}}(\mathbb{E}_t[1]\times\mathbb{E}_t[2]) \tag{2.49}$$

For the design of TMPC in [32], the admissible set of the nominal control inputs is generated by scaling set $\mathbb{U}$ using a constant value 0.6636. As shown in Figure 2.17, set $\mathbb{V}(\mathbb{E}_t)$ defined via (2.49) is consistently larger. Consequently, the original design is more conservative than the proposed approach. Note that to further reduce the conservatism, $\mathbb{V}^{\mathrm{NL}}(\mathbb{E}_t)$ and thus $\mathbb{V}(\mathbb{E}_t)$ may be approximated via polytopes that in general have more vertices than 4.

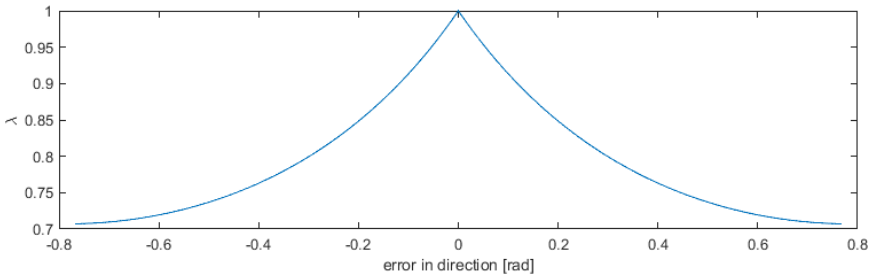


Figure 2.17: Illustration of the curve representing function $\lambda(\cdot)$ versus the error in the third dimension of the state space.

3

APPROXIMATE SDD-TMPC WITH SPIKING NEURAL NETWORKS:

AN APPLICATION TO WHEELED ROBOTS[1]

Model Predictive Control (MPC) optimizes an objective function within a prediction window under constraints. In the presence of bounded disturbances, robust versions are used. State-Dependent Dynamic Tube-based MPC (SDD-TMPC) is potentially able to outperform SOTA approaches, but solving its optimization problem online is computationally expensive. An efficient approximation method, such as Neural Network (NN), can be substituted to accelerate the online computation. There are discrepancies between the control inputs due to the approximation. It is proposed to model them as bounded state-dependent disturbances to robustly control nonlinear wheeled robots. A Spiking Neural Network (SNN) is considered to ensure that small robots could use it.

3.1. INTRODUCTION

Robust versions of Model Predictive Control (MPC) ensure that, despite bounded disturbances, the constraints are satisfied. Since Robust MPC (RMPC) is designed for the worst-case disturbance scenarios, the performance is compromised. As explained in the introduction, a common RMPC method is Tube-based MPC (TMPC), which uses a tube, i.e., a set that includes all possible future states that satisfy the constraints despite the disturbances. Usually, the shape of the tube is designed offline, whereas TMPC determines online the center of the tube, as well as an ancillary control law that keeps the actual states of the system within the tube.

State-Dependent Dynamic Tube-based MPC (SDD-TMPC) is introduced in Chapter 2 to significantly improve the performance and feasibility of the robust control problem, compared

[1]This chapter is based on the publication [1] available at https://www.sciencedirect.com/science/article/pii/S2405896324014290

to regular TMPC. SDD-TMPC determines a dynamic tube that accounts for the dynamics of the state-dependent disturbances. A main limitation of SDD-TMPC currently is its heavy online computations.

The high online computational complexity that hampers the implementation of MPC is a well-known problem in the literature. Two solutions have been proposed: Explicit MPC (EMPC) and Approximate MPC (AMPC). In both approaches, all complex computations are performed offline, which allows the controller to work relatively quickly. In EMPC, the state space is divided into regions, each with its unique control law [2]. In linear EMPC, the regions are polyhedrons and the control laws are piecewise affine. However, the number of regions increases rapidly with the number of constraints and the prediction horizon [3]. For example, controlling an inverted pendulum on a cart (with 4 states and 1 input) while imposing box constraints requires 91 and 1638 regions, respectively, for horizons equal to 3 and 10. Thus, even though the optimization problem is not solved online, it can lead to huge memory requirements and a long time needed to find the correct control law in a predefined table.

AMPC uses a machine learning model to approximate MPC, such as neural networks, radial basis functions, or lattice representation [3]. For linear MPC, it is possible to achieve perfect approximation by using Neural Network (NN) with Rectified Linear Unit (ReLU) activation functions. Although linear MPC with a quadratic solver is the fastest feasible implementation of MPC, the speed of computation could still be increased by more than 100 times by using AMPC [4]. To date, AMPC has been used for a variety of applications, including:

- an inverted pendulum on a trolley with a network has 10 layers with 6 neurons per layer) [3]
- a continuous stirred tank reactor with a network with 1 hidden layer with 50 neurons [5]
- a drone with a network with 2 hidden layers with 32 neurons per layer [6]
- a robotic arm with 20 hidden layers with up to 1024 neurons per layer [7]

The complexity of the MPC law to be approximated determines the required size of the network. For example: the robotic arm requires a larger network because it is used to approximate a RMPC with a dynamic tube and a highly nonlinear model with trigonometric functions.

Although NNs allow for accelerated computation, approximating complex control laws can potentially lead to very deep networks. As deep NNs can be energy inefficient, it can be challenging to implement them to control small robots such as micro drones [8]. In [9], it has been shown that by using Spiking Neural Network (SNN) instead, it is possible to reduce power consumption by over 200 times. Although SNNs are rarely used for regression, they have still been used as controllers. Examples of implementations include an optic flow-based controller [10], a deep Reinforcement Learning (RL) trained controller [11], a

Proportional–Integral–Derivative (PID) mimicking controller [12], and an MPC mimicking controller [13].

In this Chapter, SNN is used to approximate SDD-TMPC. The resulting controller, called Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC), enables the implementation of complex control laws in a fast and energy-efficient manner.

The main contributions of this Chapter include:

- Introducing a computationally efficient approximate version of SNN for wheeled robots, via SNN, also noting that this is the first time that a RMPC approach is approximated via SNN.
- Proposing an algorithm for efficiently generating the dynamic tube of SDD-TMPC online, taking into account the robot's shape.
- Training an ASDD-TMPC and implementing it to control a wheeled robot in numerical simulations. The ASDD-TMPC remains safe because the differences between the controller and its approximation are treated as disturbances. Even though large bounds on the disturbances are considered, the controller is not overly conservative because robustness is designed based on state-dependent disturbances, not based on the bounds.

3.2. ASDD-TMPC

In this Section, an extended problem of the case study problem from Section 2.4.2 is introduced. Consequently, additional computational resources are necessary to address the resulting optimization problems.

To address this challenge and enable the controller to function in an online manner, it is going to be approximated with Spiking Neural Network (SNN). SNN is trained using data where the state of the system is an input to the SNN, and the control input to SDD-TMPC is an output to the SNN. Consequently, SNN learns to replicate the behavior of SDD-TMPC.

However, it should be noted that SNN is not an exact approximator. Using the SDD-TMPC controller introduced in Section 2.3.2 would not make Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC) robust. This section proposes a redesign of SDD-TMPC to address the approximation error and enhance safety.

3.2.1. PROBLEM FORMULATION CONTROLLING WHEELED ROBOT

In Section 2.4.2, a prediction model of a unicycle robot is introduced, which is given again by (3.1). The position $(p_{x,t}, p_{y,t})$, in two dimensions and rotation ψ_t of the robot's front head, is controlled by velocity v_t and ω_t. The head of the robot is its front point, located on the main axis of the robot that is perpendicular to the axis of the wheels of the robot. The control objective is to reach a destination without collisions.

$$\dot{\boldsymbol{x}}_t = \begin{bmatrix} \dot{p}_{x,t} \\ \dot{p}_{y,t} \\ \dot{\psi}_t \end{bmatrix} = \begin{bmatrix} v_t \cos(\psi_t) + \rho\,\omega_t \sin(\psi_t) \\ v_t \sin(\psi_t) + \rho\,\omega_t \cos(\psi_t) \\ \omega_t \end{bmatrix} \tag{3.1}$$

Furthermore, both the states and inputs of the system are constrained. In both Section 2.4.2 and the paper in which this problem is proposed [14], it is assumed that only the inputs are constrained. This extension necessitates the consideration of the geometric configuration of the robot.

3.2.2. SDD-TMPC

In Section 2.3.2, the basic version of State-Dependent Dynamic Tube-based MPC (SDD-TMPC) is introduced. This section extends the original SDD-TMPC formulation to take into account the fact, the controller is supposed to be approximated.

SDD-TMPC finds an optimal trajectory of nominal inputs, $\tilde{v}_k$, nominal states $\tilde{z}_k$, and the tube $\mathbb{T}_k$ by solving (2.27) which is in general a nonlinear, non-convex optimization problem. The original SDD-TMPC takes as input the real system state x_k and the nominal state z_k. Then the control input (including an online ancillary input) is determined after discretization with (2.28).

In this Chapter, SDD-TMPC is extended. If the original SDD-TMPC were replaced, the network would need the nominal state as an input, but it would not be possible to compute it, as the output of the network would only contain the input to the real system, because it is not possible to generate a nominal control with a network, as it would add disturbances to the nominal state, but by definition the nominal state cannot be affected by disturbances. Otherwise, the stability proofs would not hold. To solve this problem, an additional constraint is used:

$$\boldsymbol{x}_k \in \{\boldsymbol{z}_k\} \oplus \mathbb{E}^{\max} \tag{3.2}$$

to initialize the first nominal state based only on the current state, and the first nominal state is now a decision variable. This idea is used in, among others, [14, 15]. In such a case, the difference between the nominal state and the actual state must be within the largest possible tube $\mathbb{E}^{\max}$, which can be found via the procedure that is given by [14].

Since there are obstacles in the environment, state constraints on the position are added:

$$\text{shape}(z_{i|k} \oplus \mathbb{E}) \in \mathbb{X} \tag{3.3}$$

However, since the robot has its shape, it must be ensured that not only the head of the robot avoids collisions, but also the entire body. The shape function returns the tube consisting of all possible shapes of the whole body. Many wheeled robots, such as e-pucks [16], turtlebots [17], or irobots [18] can be represented as a circle.

In this case, the tube of all shapes can be represented by a polytope. This polytope can be constructed in the following steps:

- Choose a set of directions (e.g. $\{0, \pi/2, \pi, 3\pi/2\}$). The more directions are chosen, the more optimal the final solution, but the computation cost for solving the optimization problem will increase.
- For each direction θ_k compute the distance with (3.4a), where r is a radius of a robot and $e_k[3]$ is a third element of the vector e_k, i.e. a possible error in the direction.
- Construct a polytope, where each equation defining the polytope can be written as (3.4b). An example of such a polytope can be seen in the figure 3.1.
- To the set obtained in the previous point, add (using Minkowski summation, i.e., the output of this operation is a set containing all possible sums of all possible pairs of elements from both input sets) a box set containing all possible positional errors.

$$\text{dist}_i = \rho(\cos(\min_{e \in \mathbb{E}}(|e[3] - \theta_i|)) + 1) \tag{3.4a}$$

$$-\cos(\theta_i)p_x - \sin(\theta_i)p_y \leq \text{dist}_i \tag{3.4b}$$

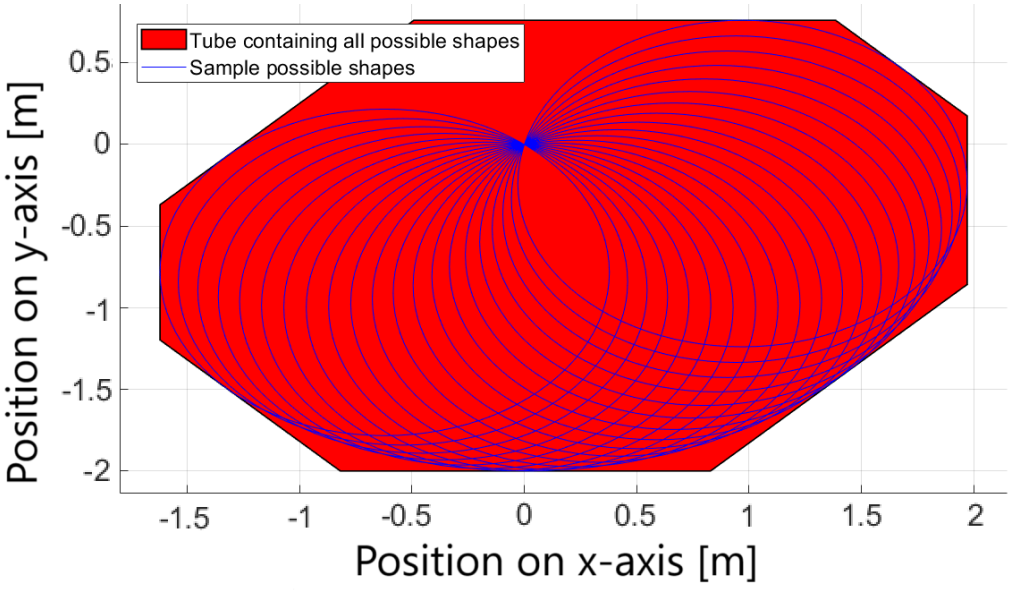


Figure 3.1: The tube contains all possible shapes of a robot under direction being bounded by 0.9 and 2.9 rad.

3.2.3. AMPC WITH SNN

In classical NN architecture, a neuron takes a set of inputs. These inputs can come from the environment or other neurons in the network. The inputs are multiplied by weights found

in the training process. The resulting values are summed and passed through a nonlinear activation function. The final value is an output from the neuron. In a feedforward network, the neurons are grouped into layers. The outputs of the neurons in one layer are sent as inputs to the next layer.

In a regression task, the output of the final layers must be a physical value. The network can be trained using a back-propagation algorithm [19]. It usually uses a mean square error between the outputs of the network and the real data. The new weights may be computed using the batch gradient descent algorithm [20], the Stochastic Gradient Descent (SGD) algorithm [21], the mini-batch gradient descent [22], the Adaptive Moment Estimation (ADAM) algorithm [23], or any other algorithm designed to update NN's weights.

Since all neurons send information all the time, it can lead to inefficient energy consumption compared to SNN [24]. In SNN, each neuron has a memory value m. In the most common application of Leak-and-Fire (LiF) neurons, the dynamics of a neuron can be described by (3.5), where β is a positive hyperparameter below 1, w_i is the weight between the current neuron and its $i-th$ input, σ_i is an $i-th$ input (this value is 0 or 1 if it is also the output of another neuron), and a reset value that decreases the memory value when the neuron fires a spike.

$$m_{k+1} = \beta m_k + \sum_{i=1}^{n} (w_i \sigma_i) - \text{reset}(m_k) \tag{3.5}$$

The network is based on [9], where the first input of the network is not a spike but a continuous value symbolizing the real input, i.e., the state of the robot. Then all neurons in the hidden layers are LiF neurons. However, in the final output layer, the neurons no longer fire spikes, and the memory value of each of them is treated as the output of the network. One could argue that this is no longer an SNN, but this is not important as long as this architecture still allows energy to be saved.

To collect data for training the network, randomly sampled states are simulated to compute and store a control input for that state. This process is repeated until a sufficiently large data set is generated. It is important to note that the solver may find a local minimum that is not necessarily close to the global minimum, so more time (one hour per sample) is spent searching for the solution. The reason for this is that local minima can be treated as outliers during SNN training, which makes the training process more difficult.

There are several problems with the use of SNN. First, the reset mechanism is a step function and cannot be differentiated. Its gradient is 0 or infinite. When SGD is used, the derivative of the step is replaced during backpropagation by the derivative of a function called the surrogate gradient [25]. This makes it more difficult to train SNN, but there is no other choice if a gradient-based training algorithm is chosen.

Second, if MPC has access to the current state, it can find the control input, i.e., it does not need past information. SNN, on the other hand, requires memory to operate. However, SNN

has been shown to work even with a time-invariant dataset such as the MNIST dataset [26]. This can be done by simulating the network for some constant time steps and taking the final values of the memory as the output of the network after those time steps. However, in such a case, it is necessary to use the backpropagation through time algorithm [27] instead of the standard backpropagation, which takes more real time to train.

Third, as learned from early experiments, if the initial weights are generated randomly, it is likely that some neurons will not fire at all, no matter what the input is. This makes their gradient very small, which means that their weights are rarely updated during training. Such neurons can be called dead neurons [24]. In training, mean square error is used, but to solve this problem an additive regularization term is added to penalize the network if some neurons fire less than a certain threshold for the whole batch, i.e. it is fine if a neuron does not fire for a certain state, but if it does not fire at all, it is penalized. The formula for computing the penalty p is given below:

$$p = c \sum_{i=1}^{N_{\text{neuron}}} \left(\text{ReLU}(\alpha - \sum_{j=1}^{N_{\text{batch}}} \sum_{k=1}^{T_{\text{sim}}} \sigma_{i,j,k})\right)^2 \tag{3.6}$$

where N_{neuron} denotes the number of neurons in the network, N_{batch} represents the number of data sample per batch, T_{sim} is the time required to simulate the network to obtain the final output, α is a hyperparameter that demonstrates the minimum number of times a neuron should fire per batch, and c is a hyperparameter that shows the ratio between the loss function and the regularization term. Finally, the value of the variable $\sigma_{i,j,k}$ is either 0 or 1, indicating whether a given neuron in a given batch fired at a given time. It is important to note that this regularization term is incompatible with SGD because the regularization term would force the network to fire each neuron on every sample.

3.2.4. CONSTRAINT SATISFACTION

The main advantage of any RMPC is in providing safety, i.e., in ensuring that none of the constraints is ever violated. To ensure constraint satisfaction after replacing SDD-TMPC with ASDD-TMPC, A disturbance $\boldsymbol{d}_k$ is considered because the MPC control input $\boldsymbol{u}_k^{\text{MPC}}$ is not perfectly mimicked by the SNN input $\boldsymbol{u}_k^{\text{SNN}}$, as indicated in (3.7). $\boldsymbol{d}_k$ is assumed to be bounded i.e. $\boldsymbol{d}_k \in \mathbb{D}$. By adding $\boldsymbol{d}_k$ to the original model, (3.1) results in (3.8). This leads to a direction-dependent set $\mathbb{W}_{i-1|k}^d$ that is obtained via a nonlinear mapping of the original boundary set for the disturbances, as it is given in (3.9). I then include the knowledge of these disturbances within SDD-TMPC by replacing $\mathbb{W}_{i-1|k}(\cdot)$ in (2.27e) with $\mathbb{W}_{i-1|k}(\cdot) \oplus \mathbb{W}_{i-1|k}^d(\cdot)$. Even though there are disturbances on control input, since this has been modeled as a state-dependent additive disturbance, SDD-TMPC guarantees the satisfaction of the state constraints.

As long as the assumption $\boldsymbol{d}_k \in \mathbb{D}$ holds, the constraint will never be violated, and it is possible to compute the probability of the assumption being satisfied using Hoeffding's

inequality [28], but the drawback of this approach is that it requires a lot of samples.

It may be impossible to estimate the set $\mathbb{D}$ before training the NN, but if the results are not satisfactory, generating more data and training a more complex network is possible.

$$\boldsymbol{d}_k = [d_k^v \; d_k^\omega]^T = \boldsymbol{u}_k^{\text{MPC}} - \boldsymbol{u}_k^{\text{SNN}} \tag{3.7}$$

$$\dot{\boldsymbol{x}}_t = \begin{bmatrix} \cos(\psi_t) & \rho\sin(\psi_t) \\ \sin(\psi_t) & \rho\cos(\psi_t) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_t + d_t^v \\ \omega_t + d_t^\omega \end{bmatrix} \tag{3.8}$$

$$\mathbb{W}^d(\psi) = \begin{bmatrix} \cos(\psi_t) & \rho\sin(\psi_t) \\ \sin(\psi_t) & \rho\cos(\psi_t) \\ 0 & 1 \end{bmatrix} \mathbb{D} \tag{3.9}$$

3.3. SIMULATION RESULTS

In this Chapter, an iRobot Create3 is simulated. All used code can be found in the published repository [29], which also contains a Robotics Operating System 2 (ROS2) package. The package is tested in a laboratory at the University of Delft called Cyber Zoo [30]. The shape of this robot is approximated by a circle with a radius r equal to 23.7 cm. The maximum linear velocity $v^{\max}$ is equal to 0.46 $\frac{\text{m}}{\text{s}}$ and the input of the robot is constrained by $v_t + r\omega_t \leq v^{\max}$.

At each iteration, the robot is randomly placed in a circular area with a radius of 1.08 m surrounded by a wall. A robot is instructed to minimize its position and its inputs with cost matrices equal to Q=diag(0.4,0.4), R=diag(0.2,0.2), and F = diag(0.5,0.5). The horizon is set to 20. It is assumed that the disturbance $\boldsymbol{d}$ is at most 20% of the maximum input value. For this disturbance, the largest possible error set $\mathbb{E}_{\max} =: \left\{\boldsymbol{e} \in \mathbb{R}^3 | \max(|\boldsymbol{e}[1]|, |\boldsymbol{e}[2]|) \leq 0.1\text{m}, |\boldsymbol{e}[3]| \leq \pi\right\}$ can be defined. The robot's control trajectory has to satisfy the control input constraint, never let the robot collide with a wall, and let the nominal state reach the terminal set, which is a circle with a radius equal to 0.5 m.

In the next step, the collected data are used to train 2 SNNs: Linear Velocity Network (LVN) and Angular Velocity Network (AVN). It is done because learning the rotational velocity proves to be more challenging. As a result, a single SNN is more focused on minimizing the angular velocity, which leads to a higher number of outliers while trying to predict the linear velocity. 1168 data samples (900 for training and 268 for validation) are created. It is possible to ignore the direction by always rotating the coordinate system by the robot's direction before generating the data. The same approach is used in [31]. Note that this is only possible if the robot is tasked with reaching a destination without being given explicit instructions regarding the direction in which it should reach said destination.

The LVN has 5 hidden layers with 50, 200, 1000, 200, and 50 neurons in the layers, and the network is simulated for 20 iterations per control input. The rotation control input proves

to be more difficult, and AVN has 7 hidden layers with 150, 200, 1000, 1200, 1000, 500, and 150, and is simulated for 40 iterations. In both SNNs, the Arctangent (Atan) [32] function is used as a surrogate gradient and hard reset (after spiking, a neuron's memory is set to 0).

From the experiments, it is learned that longer simulations of SNN work as a regularization factor, i.e., it reduces the validation error, but it makes training much more time-consuming. Thanks to this regularization factor, it is possible to train a deeper neural network with less data. The networks are trained using the ADAM solver with a learning rate 10^{-4} and a batch size of 32.

According to the training for LVN, the average training and validation errors are 0.0326 and 0.0413, respectively. All training errors shown are relative to the maximum control input value. The highest error is equal to 0.156, which means that the initial assumption is very likely to be satisfied. According to the training for AVN, the average training and validation errors are 0.0373 and 0.0431 respectively. The highest error is 0.233, and the assumption $\boldsymbol{d}_k \in \mathbb{D}$ is not satisfied 4 times (3 times in training and 1 time in the validation dataset). It is possible to solve this problem by creating a larger set of possible disturbances, generating new data, and retraining the network. However, even though the initial assumption is not always satisfied, the simulations show that if it happens sporadically, it does not affect the performance too much, but increasing the disturbance set would occur in a more conservative controller.

In the last step, ASDD-TMPC and SDD-TMPC are compared. In figure 3.2 one can see the consistent behavior of the controller when the robot has to move from position $[0.5, 0.5]$ and orientation 0.5π to the origin. It shows the position of the head over time using both controllers. To better demonstrate what the robot is doing (i.e., it first corrects its orientation by moving backward, then it turns to reach the target), the line between the center and the head showing the orientation is added. It can be seen that the behavior of both controllers is very similar, but ASDD-TMPC is slightly more aggressive.

An important challenge is presented by Figure 3.3, where SDD-TMPC can reach the target, but by using ASDD-TMPC, a few centimeters of steady-state error is displayed. This is what should be expected, because according to the largest tube, the largest steady-state error could even be over 0.15 cm, but it is unlikely since the average approximation error is usually much lower than the largest disturbance, according to the assumption. One way to solve this problem would be to use a different controller at the end when a robot is close to the destination. Figure 3.4 shows how the distance changes, and even though ASDD-TMPC is more aggressive, the discrepancy in the control input leads to the constant steady-state error. Finally, Figure 3.5 shows that the relationship between the inputs of both controllers is very similar, with ASDD-TMPC again being more aggressive, but it also clearly shows how ASDD-TMPC is affected by the disturbances on control input.

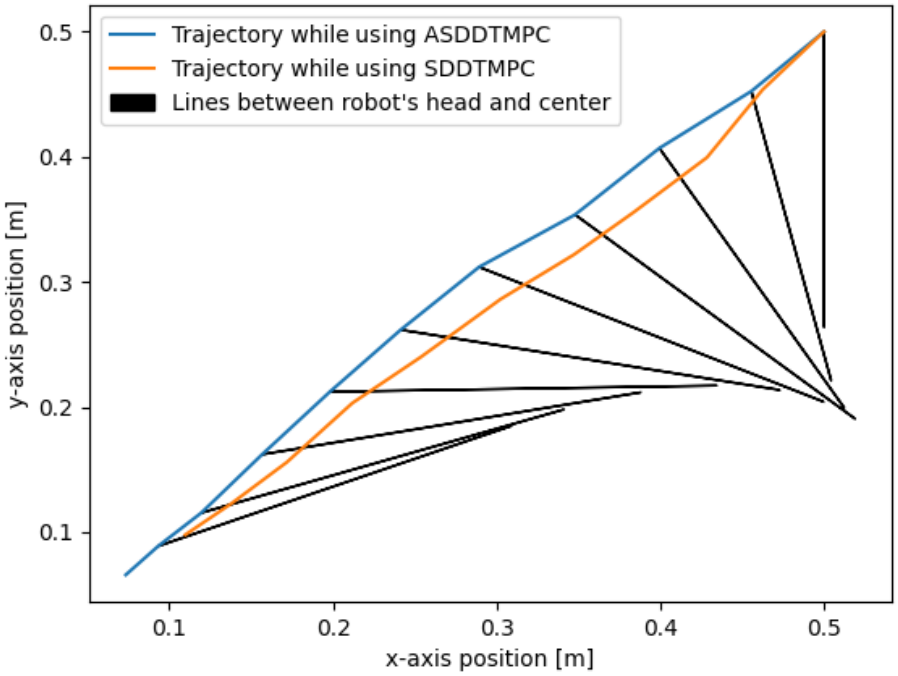


Figure 3.2: The transient behavior of the robot's head while controlled by SDD-TMPC and ASDD-TMPC. The black lines show the distance between the center and the head to better explain the robot's behavior (ASDD-TMPC only).

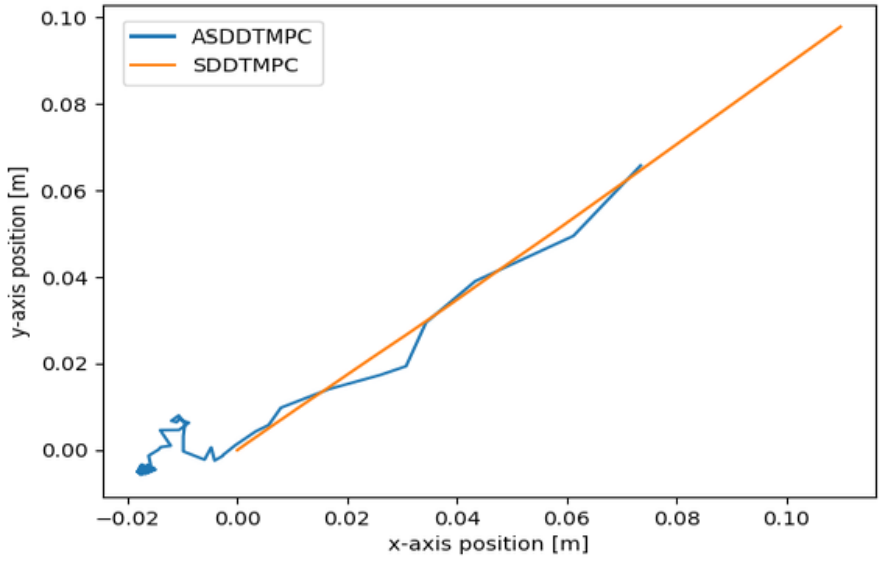


Figure 3.3: The stable behavior of the robot's head while controlled by SDD-TMPC and ASDD-TMPC.

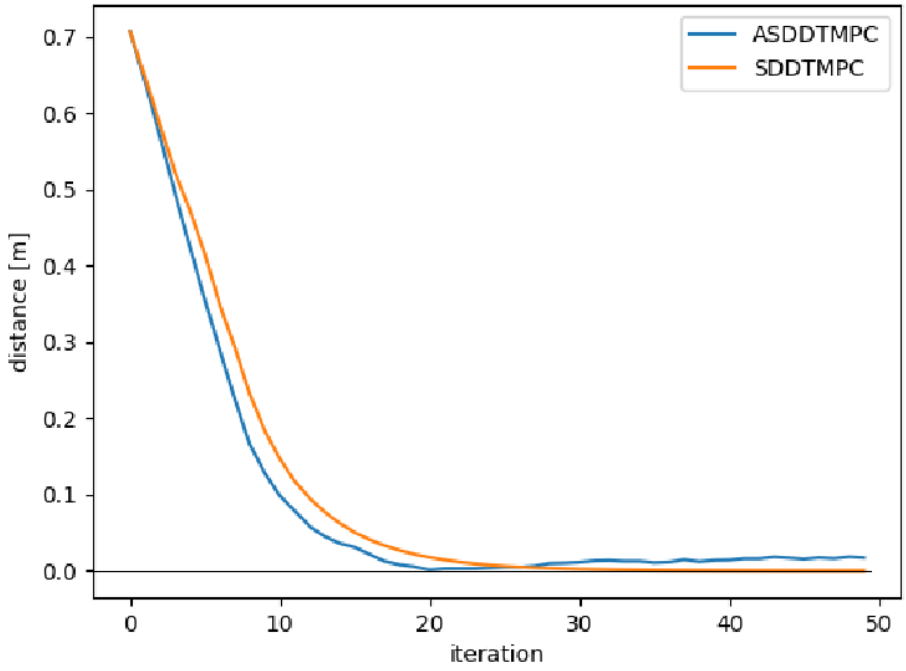


Figure 3.4: Distance between robot head and origin over time

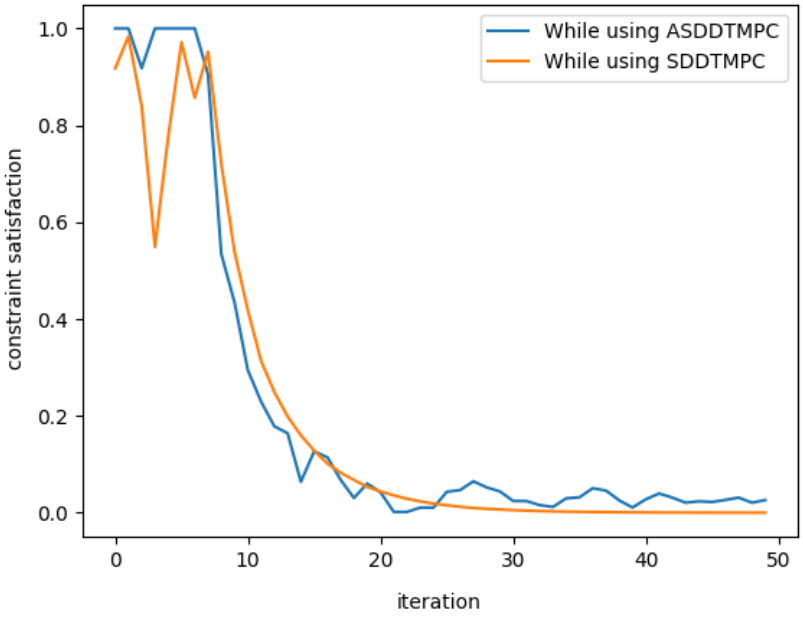


Figure 3.5: Control input over time. It is scaled so that if the value is 1, the robot cannot move faster with the same ratio between linear and angular velocity.

3.4. CONCLUSION

In this Chapter, the original SDD-TMPC is extended, where the main drawback is the time required to compute a control input. This problem is solved by approximating the controller using SNNs for the first time, which reduces the online computation time but introduced an additional control input. Based on the results, the time required to compute the control input is reduced from minutes to milliseconds, a massive improvement. In addition, the controller is extended to take into account the shape of the robot, even when the direction is uncertain.

According to the dataset, two trained networks are able to imitate the original controller, rarely breaking the original assumption that the disturbance in the control input is less than a chosen constant. However, if safety is critical, it is always possible to retrain the network using SDD-TMPC with a larger disturbance set, but the constraint satisfaction will always remain only probabilistic. While training the neural network, there are no assumptions about the model, cost function, etc., so this approach applies to other control problems as long as there is enough data and the network has a suitable architecture (i.e., number of layers and neurons).

The main challenge for now is the generalization, i.e., every time the model or the constraints change (e.g., the robot works in a different environment), the networks have to be re-trained, which makes it impossible to control the robot in an unknown environment. The next critical step would be to add information gathered by sensors as another input to the network. However, it is not clear whether it would be a raw input (e.g., laser scan) or high-level features (e.g., binary map created with laser scan). Generalization with respect to the environment could be achieved using motion primitives. Learning control laws for rotational or longitudinal motions could then be parameterized across scenarios [33] Another important extension would be to find a way to generate samples more efficiently, as currently it takes a lot of time and many more samples to mimic SDD-TMPC for more complex systems.

As the drawbacks associated with the approach—namely, the extensive offline training, the inability to function in a rapidly changing environment, and only probabilistic constraint satisfaction—may prove to be unacceptable for certain implementations, a hierarchical approach is introduced in Chapter 4. It is free of the mentioned problems, but it still requires finding the solution to the optimization problem, so the reduction of computational complexity is not as high.

REFERENCES

[1] F. Surma and A. Jamshidnejad. "Approximate SDD-TMPC with Spiking Neural Networks: An Application to Wheeled Robots". In: *IFAC-PapersOnLine* 58 (2024). DOI: 10.1016/j.ifacol.2024.09.050.

[2] A. Grancharova and T. Johansen. *Explicit Nonlinear Model Predictive Control: Theory and Applications*. 1st. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-28779-4.

[3] B. Karg and S. Lucia. "Efficient Representation and Approximation of Model Predictive Control Laws via Deep Learning". In: *IEEE Transactions on Cybernetics* 50 (2020). DOI: 10.1109/TCYB.2020.2999556.

[4] Y. S. Quan and C. C. Chung. "Approximate Model Predictive Control with Recurrent Neural Network for Autonomous Driving Vehicles". In: *2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2019, pp. 1076–1081. DOI: 10.23919/SICE.2019.8859955.

[5] K. Wang, X. Liang, and J. Xu. "Approximating Model Predictive Controller With Biased ReLU Neural Network". In: *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 3780–3785. DOI: 10.1109/CAC51589.2020.9326497.

[6] A. Tagliabue, D.-K. Kim, M. Everett, and J. How. "Demonstration-Efficient Guided Policy Search via Imitation of Robust Tube MPC". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 462–468. DOI: 10.1109/ICRA46639.2022.9812122.

[7] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe. "Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control". In: *IEEE Robotics and Automation Letters* 5 (2020). DOI: 10.1109/LRA.2020.2975727.

[8] S. Stroobants, C. De Wagter, and G. De Croon. "Neuromorphic Control Using Input-Weighted Threshold Adaptation". In: *Proceedings of the 2023 International Conference on Neuromorphic Systems*. ICONS '23. Santa Fe, NM, USA: Association for Computing Machinery, 2023. DOI: 10.1145/3589737.3605963.

[9] A. Henkes, J. Eshraghian, and H. Wessels. "Spiking neural networks for nonlinear regression". In: *Royal Society Open Science* 11 (2024). DOI: 10.1098/rsos.231606.

[10] J. Dupeyroux, J. J. Hagenaars, F. Paredes-Vallés, and G. C. H. E. de Croon. "Neuromorphic control for optic-flow-based landing of MAVs using the Loihi processor". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE / IEEE Robotics and Automation Society, 2021, pp. 96–102. DOI: 10.1109/ICRA48506.2021.9560937.

[11] G. Tang, N. Kumar, R. Yoo, and K. Michmizos. “Deep Reinforcement Learning with Population-Coded Spiking Neural Network for Continuous Control”. In: *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021, pp. 2016–2029. URL: https://proceedings.mlr.press/v155/tang21a.html.

[12] T. Burgers, S. Stroobants, and G. C. H. E. de Croon. “Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps”. In: *15th Annual International Micro Air Vehicle Conference and Competition*. Institution of Micro Air Vehicles, 2024, pp. 73–79. URL: https://2024.imavs.org/.

[13] R. Halaly and E. Ezra Tsur. “Autonomous driving controllers with neuromorphic spiking neural networks”. In: *Frontiers in Neurorobotics* 17 (2023). DOI: 10.3389/fnbot.2023.1234962.

[14] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. “Robust MPC for tracking constrained unicycle robots with additive disturbances”. In: *Automatica* 90 (2018). DOI: 10.1016/j.automatica.2017.12.048.

[15] D. Mayne, M. Seron, and S. Raković. “Robust model predictive control of constrained linear systems with bounded disturbances”. In: *Automatica* 41 (2005). DOI: 10.1016/j.automatica.2004.08.019.

[16] P. Gonçalves, P. Torres, C. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. “The e-puck, a Robot Designed for Education in Engineering”. In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* 1 (2009). URL: https://e-puck.gctronic.com/index.php?Itemid=36&id=34&option=com_content&view=article.

[17] R. Amsters and P. Slaets. “Turtlebot 3 as a Robotics Education Platform”. In: *Robotics in Education: Current Research and Innovations*. Vol. 1023. Springer, 2019. DOI: 10.1007/978-3-030-26945-6_16.

[18] J. Espada, S. Qiu, R. Gonzalez Crespo, and J. Carús. “Leveraging large language models for autonomous robotic mapping and navigation”. In: *International Journal of Advanced Robotic Systems* 22 (2025). DOI: 10.1177/17298806251325965.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986). DOI: 10.1038/323533a0.

[20] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade*. 2nd. Vol. 7700. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3.

[21] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Physica-Verlag HD, 2010, pp. 177–186. DOI: doi.org/10.1007/978-3-7908-2604-3_16.

[22] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. “Parallelizing Stochastic Gradient Descent for Least Squares Regression: Mini-batching, Averaging, and Model Misspecification”. In: *Journal of Machine Learning Research* 18 (2018). URL: https://jmlr.org/papers/v18/16-595.html?utm_source=chatgpt.com.

[23] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*. arXiv, 2015. DOI: 10.48550/arXiv.1412.6980.

[24] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu. “Training Spiking Neural Networks Using Lessons From Deep Learning”. In: *Proceedings of the IEEE* 111 (2023). DOI: 10.1109/JPROC.2023.3308088.

[25] E. Neftci, H. Mostafa, and F. Zenke. “Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks”. In: *IEEE Signal Processing Magazine* 36 (2019). DOI: 10.1109/MSP.2019.2931595.

[26] S. Subbulakshmi Radhakrishnan, A. Sebastian, A. Oberoi, S. Das, and S. Das. “A biomimetic neural encoder for spiking neural network”. In: *Nature Communications* 12 (2021). DOI: 10.1038/s41467-021-22332-8.

[27] P. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78 (1990). DOI: 10.1109/5.58337.

[28] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. “Learning an Approximate Model Predictive Controller With Guarantees”. In: *IEEE Control Systems Letters* 2 (2018). DOI: 10.1109/LCSYS.2018.2843682.

[29] F. Surma. *Implemenation of Appoximate state-dependent dynamic TMPC*. 2024. DOI: 10.4121/6568e235-289d-4be0-b931-ec219c559f6f.

[30] *CyberZoo*. URL: https://www.tudelft.nl/en/ae/organisation/departments/control-and-operations/facilities/drone-facilities/cyberzoo (visited on 07/19/2025).

[31] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How. “Demonstration-Efficient Guided Policy Search via Imitation of Robust Tubce MPC”. In: *International Conference on Robotics and Automation*. IEEE, 2022, pp. 462–468. DOI: 10.1109/ICRA46639.2022.9812122.

[32] T. T. Sivri, N. P. Akman, and A. Berkol. “Multiclass Classification Using Arctangent Activation Function and Its Variations”. In: *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, 2022, pp. 1–6. DOI: 10.1109/ECAI54874.2022.9847486.

[33] Z. Zhao, A. Girard, and S. Olaru. “Symbolically Synthesized Motion Primitives for Autonomous Navigation”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)* (2024), pp. 5846–5851. URL: https://api.semanticscholar.org/CorpusID:273550736.

4

PARENT-CHILD MPC ARCHITECTURE:

RECURSIVE FEASIBILITY WITHOUT TERMINAL CONSTRAINTS [1]

This paper proposes a novel hierarchical Model Predictive Control (MPC) framework, called the Parent-Child MPC (PC-MPC) architecture, to steer nonlinear systems under uncertainty towards a target set, balancing computational complexity and guaranteeing recursive feasibility and stability without relying on conservative terminal constraints in online decision-making. By coupling a small-horizon Child MPC (C-MPC) layer with one or more large-horizon Parent MPC (P-MPC) layers, the architecture ensures recursive feasibility and stability through adjustable stage-wise constraints derived from tube-based control. As is demonstrated in the case studies, compared to traditional MPC methods, the proposed PC-MPC architecture enhances performance and computational efficiency, reduces conservativeness, and enables scalable planning for certain nonlinear systems.

4.1. INTRODUCTION

As explained in detail in Chapter 1, Model Predictive Control (MPC) employs a model of the system to predict its future behavior over a finite time horizon and to accordingly optimize the control input trajectory. A key strength of MPC lies in its ability to explicitly incorporate constraints during the optimization procedure [2].

As infinite constrained optimization is usually computationally intractable, MPC operates over a finite prediction horizon. Due to this, ensuring stability for MPC is generally challenging, and may be achieved via, e.g., enforcing passivity conditions [3], using the augmented stage cost function corresponding to a single time step as a control Lyapunov

[1] This chapter is based on the submitted paper [1] available at https://www.sciencedirect.com/science/article/pii/S2666720725001389

function [4], or treating the optimal cost of the MPC optimization problem as a control Lyapunov function and demonstrating its decline [2].

These methods for MPC often involve supplementary, terminal constraints. Applied exclusively to the final predicted state, terminal constraints guide the system towards a terminal state region $\mathcal{X}^{\mathrm{f}} \subseteq \mathcal{X}$ near its destination (with $\mathcal{X}$ the admissible state set). The N-step controllable set $\mathcal{X}_N$ is a set including all states that are steerable by an admissible control input sequence of length N — the number of time steps across the prediction horizon — into set $\mathcal{X}^{\mathrm{f}}$ [5]. For larger prediction horizons, this controllable set is usually expanded. In other words, some states, although safely steerable into $\mathcal{X}^{\mathrm{f}}$ with large prediction horizons, lie outside the reach of short-horizon MPC.

Large-horizon MPC is typically computationally prohibitive [6]. It is thus common to simplify (e.g., linearize) the prediction model [7, 8]. This, however, potentially leads to constraint violations due to model inaccuracies. Alternatively, as explained in [9], the sampling time is manipulated to enlarge the controllable state set without increasing the prediction horizon.

Hybrid approaches can integrate MPC with other controllers. An example involves dividing the prediction horizon into a control horizon governed by MPC and a segment where the last MPC input is simply repeated [2]. This approach usually enhances computational efficiency by reducing the number of decision variables. Alternatively, after executing the MPC trajectory for given time steps, a terminal control law (e.g., Linear Quadratic Regulator (LQR) for linear systems) is activated [2, 10, 11]. This method is based on the invariance of the terminal set.

These strategies all rely on reaching a given terminal set via MPC, without inherently enlarging the N-step controllable state set. Hierarchical MPC offers promising alternatives: In [12, 13], MPC performs high-level planning, generating optimal reference trajectories. A lower-level controller executes the plan. A recent bi-level MPC architecture in [14] synergizes two MPC formulations: "Rough long-vision" MPC with a coarse prediction model and large sampling time, and "detailed short-vision" MPC for enhancing and executing the plan across a smaller prediction horizon with smaller sampling times. Long-term MPC provides a trajectory that guides the constraints of short-term MPC.

Inspired by this, the challenge of reaching a designated terminal set through online decision-making is addressed by proposing a multi-level architecture composed of interconnected MPC systems, leveraging Tube-based MPC (TMPC). TMPC ensures constraint satisfaction under bounded uncertainties without significantly increasing online computational complexity [2]. It uses a fixed-shape tube (e.g., a polyhedron or ellipsoid [15]) around the nominal trajectory to guarantee safety [16]. The nominal component of TMPC governs the center of the tube, while an ancillary controller maintains the actual state within the tube. The fixed tube structure, however, may lead to conservative behavior and reduced performance, as shown in Chapter 2 where TMPC causes a substantial unnecessary reduction in the robot velocity, up to 30%. This performance degradation stems from the dual role of the ancillary control law, as it should

both ensure stability and compensate for disturbances and model discrepancies. These dual objectives restrict the flexibility of nominal MPC in selecting optimal control inputs. The key contributions of this Chapter are:

- A novel hierarchical control framework is introduced, the Parent-Child MPC (PC-MPC) architecture, which effectively replaces the nominal component in TMPC for certain nonlinear systems. It guarantees convergence to a designated terminal set without imposing a terminal constraint, thereby avoiding conservativeness and risk of infeasibility typically associated with such constraints.

- A small-horizon Child MPC (C-MPC) controller directly steers the system, emphasizing local optimality. One or more large-horizon Parent MPC (P-MPC) layers address stability and long-term feasibility. Strategic interactions of these layers allow for replacement of terminal constraints with adjustable stage-wise state constraints, including robust positive invariant tubes derived by P-MPC. This significantly enlarges the feasible exploration space of C-MPC, leading to improved performance.

- The cross section of these tubes is composed of a fixed subset $\mathcal{E}^{\mathrm{P}}$ of the state space. It is proven that $\mathcal{E}^{\mathrm{P}}$ is also a robust positive invariant set for the C-MPC nominal trajectory. Once this nominal trajectory enters $\mathcal{E}^{\mathrm{P}}$, the goal of PC-MPC architecture is met, allowing a secondary low-complexity controller (e.g., small-horizon conventional TMPC) to steer the actual state towards a neighborhood of the origin.

Similarly to [14], the trajectories of PC-MPC layers — based on linear TMPC [16] — are used to formulate constraints for C-MPC that ensure long-term feasibility and stability. C-MPC, similarly to an ancillary controller, directly steers the system, which is initialized far from its desired state, with no reference trajectory. Unlike the approach in [17], where MPC serves as an ancillary controller within a nonlinear TMPC framework, C-MPC does not track the nominal trajectories of P-MPC. Instead, it shares the same cost function as P-MPC, while employing a more accurate prediction model and finer sampling times.

Section 4.2 formulates the problem. Section 4.3 details the PC-MPC architecture. Section 4.4 presents and discusses the results of a case study, and Section 4.5 concludes the paper and proposes directions for future research.

4.2. PROBLEM FORMULATION

This section describes the problem of controlling a nonlinear system subject to additive disturbances, with the goal of steering the system from an arbitrary initial state to a desired target state. The system model follows the formulation in [18, 19].

4.2.1. SYSTEM MODELING

The system dynamics is described in discrete time by the following nonlinear model subject to additive disturbances:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + g(\boldsymbol{x}_k) + B\boldsymbol{u}_k + \boldsymbol{w}_k \tag{4.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n \subseteq \mathcal{X}$ is the state vector, $\boldsymbol{u}_k \in \mathbb{R}^m \subseteq \mathcal{U}$ is the control input vector, $\boldsymbol{w}_k \in \mathbb{R}^n \subseteq \mathcal{W}$ is an unknown bounded disturbance vector. The states, control inputs, and disturbances are constrained to convex, compact sets $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{W}$, respectively. Function $g(\cdot)$ is nonlinear Lipschitz with Lipschitz constant $L > 0$, satisfying $g(\mathbf{0}) = \mathbf{0}$. The matrix pair (A, B) is stabilizable. Lipschitz continuity of $g(\cdot)$ implies that:

$$\|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \qquad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X} \tag{4.2}$$

4.2.2. CONTROL PROBLEM

The control objective is to regulate the state towards a suitable robust positive invariant set, including the origin, as defined in [16]. Until this goal is reached, a stage cost $l(\cdot,\cdot)$, given below, is accumulated, with Q and R positive definite matrices:

$$l(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k^\top Q \boldsymbol{x}_k + \boldsymbol{u}_k^\top R \boldsymbol{u}_k \tag{4.3}$$

As shown in [19], systems given by (4.1) under the stated conditions can robustly be controlled using TMPC. The nominal optimization problem of nonlinear TMPC at time step k, with prediction horizon N, for system (4.1) is given by:

$$V(\boldsymbol{x}_k) = \min_{\tilde{\boldsymbol{z}}_{k:k+N}, \tilde{\boldsymbol{v}}_{k:k+N-1}} \sum_{i=k}^{k+N-1} \left(\boldsymbol{z}_{i|k}^\top Q \boldsymbol{z}_{i|k} + \boldsymbol{v}_{i|k}^\top R \boldsymbol{v}_{i|k} \right) + \boldsymbol{z}_{k+N|k}^\top P \boldsymbol{z}_{k+N|k} \tag{4.4a}$$

such that for $i = k, \ldots, k+N-1$:

$$x_k \in \{\boldsymbol{z}_k\} \oplus \mathcal{E} \tag{4.4b}$$

$$\boldsymbol{z}_{i+1|k} = A\boldsymbol{z}_{i|k} + g\left(\boldsymbol{z}_{i|k}\right) + B\boldsymbol{v}_{i|k} \tag{4.4c}$$

$$\boldsymbol{z}_{i+1|k} \in \mathcal{X} \ominus \mathcal{E} \tag{4.4d}$$

$$\boldsymbol{v}_{i|k} \in \mathcal{V} \tag{4.4e}$$

$$\boldsymbol{z}_{k+N|k} \in \mathcal{Z}^{\mathrm{f}} \tag{4.4f}$$

which, rather than directly steering the system state $\boldsymbol{x}_k$, determines nominal control input $\boldsymbol{v}_k$ to steer the nominal state $\boldsymbol{z}_k$. The nominal prediction model is given via (4.4c), which eliminates the disturbance term $\boldsymbol{w}_k$ from (4.1). The nominal state trajectory represents the center of a rigid tube with cross section $\mathcal{E}$, i.e., the set of possible errors between nominal and actual states. The state admissibility constraint (4.4d) has been tightened to ensure the actual state, impacted by uncertainties, remains within the admissible state set $\mathcal{X}$. The notation $a_{k_2|k_1}$, with $k_2 \geq k_1$, shows the value of variable a at time step k_2 predicted at time step k_1,

with $a_{k_1|k_1} := a_{k_1}$. Tilde for decision variables shows a sequence, where $\tilde{a}_{k_3:k_4}$ with $k_4 \geq k_3$ represents $[a_{k_3}, \ldots, a_{k_4}]^\top$. Constraint (4.4b) adjusts the nominal state $\boldsymbol{z}_k$ at time step k to locate the cross section $\{\boldsymbol{z}_k\} \oplus \mathcal{E}$ of the tube within set $\mathcal{X}$, so that the measured state $\boldsymbol{x}_k$ lies within the tube. For all time steps $i = k, \ldots, k+N-1$, the actual control input $\boldsymbol{u}_{i|k}$ should satisfy $\boldsymbol{u}_{i|k} \in \mathcal{U}$. This requires defining tightened constraints on nominal control inputs, as is done in (4.4e). The terminal constraint (4.4f), where $\mathcal{Z}^{\mathrm{f}}$ is a control invariant set, designed as a tightened version of the actual terminal state set $\mathcal{X}^{\mathrm{f}}$, together with the terminal cost $\boldsymbol{z}_{k+N|k}^\top P \boldsymbol{z}_{k+N|k}$ in (4.4a), ensures recursive feasibility — i.e., if a solution initially exists, a solution exists for all future time steps. However, as explained earlier, imposing such a terminal constraint may restrict the set of initial states from which the system is steerable to the origin. Note that $\oplus$ and $\ominus$ show the Minkowski set summation and subtraction, respectively.

The actual system, which is prone to disturbances and model discrepancies, is controlled via an ancillary control law:

$$\boldsymbol{u}_k = \pi(\boldsymbol{x}_k, \boldsymbol{z}_k, \boldsymbol{v}_k), \tag{4.5}$$

designed to ensure, through the tightened constraint (4.4d), that the evolved actual state $\boldsymbol{x}_{k+1}$ remains within the tube, provided that the current state x_k satisfies (4.4b). The tube cross section $\{\boldsymbol{z}_{i|k}\} \oplus \mathcal{E}$, for all time steps $i = k, \ldots, k+N-1$, is a robust positive invariant set when the system follows control input $\pi(\boldsymbol{x}_k, \boldsymbol{z}_k, \boldsymbol{v}_k)$.

The goal is to design an MPC-based control architecture that can steer the system to the origin from any admissible initial state — assuming a feasible control problem — while guaranteeing stability and recursive feasibility. Although solving (4.4) with a sufficiently large prediction horizon theoretically achieves this goal, the computational burden may be prohibitive, as shown in Chapter 2. The control architecture proposed in this paper addresses this issue by alternately solving an additional quadratic program alongside the nonlinear program, reducing the computational complexity while maintaining the desired properties.

4.3. PROPOSED ARCHITECTURE: PARENT-CHILD MPC

This section discusses the proposed Parent-Child MPC (PC-MPC) architecture, which replaces the nominal problem in conventional TMPC given by (4.4) to offer a computationally efficient alternative with stability guarantees. The P-MPC extends standard linear TMPC, incorporating both a nominal MPC problem and a feedback ancillary control law. In contrast, the C-MPC leverages the nominal problem from the nonlinear TMPC formulation in (4.4) to more flexibly steer the system. Unlike traditional TMPC, C-MPC omits terminal constraints, thereby eliminating the restrictive influence of these sets, as discussed in Section 4.1. Instead, C-MPC operates under additional stage-wise constraints derived by PC-MPC, in the form of translated tubes – robust positive invariant sets – composed of a fixed subset $\mathcal{E}^{\mathrm{P}}$ of the state space centered along the nominal state trajectory of P-MPC.

Set $\mathcal{E}^{\mathrm{P}}$ is designed to lie within the controllable state set, from which the nominal trajectory of C-MPC is steerable into a desired terminal set. It is proven that, under the PC-MPC strategy, the untranslated set $\mathcal{E}^{\mathrm{P}}$ itself is a robust positive invariant set for the nominal C-MPC trajectory. Therefore, once the nominal C-MPC trajectory enters this set, the primary objective of the PC-MPC architecture is fulfilled. From this point onward, a secondary control phase – such as a conventional TMPC scheme with a small horizon (and thus affordable online computations), or another control-invariant strategy – may be employed to steer the actual state trajectory towards a desired neighborhood of the origin.

P-MPC employs a simplified nominal version of the system model in (4.1) that eliminates both the Lipschitz nonlinearity and the additive disturbances. The underlying linear TMPC formulation of P-MPC is designed to mitigate the uncertainties introduced by the model discrepancy that stems from omitting the Lipschitz nonlinearities of the original system. By adopting a coarser sampling time for prediction and optimization, P-MPC reduces computational complexity and supports larger prediction horizons. While this long-term planning facilitates stability, directly following the trajectories generated by P-MPC may lead to suboptimal performance due to model inaccuracies.

C-MPC operates at a finer resolution, using a nominal, yet nonlinear, version of the model in (4.1) that solely eliminates the additive disturbances. Exploiting its more accurate model, C-MPC focuses on short-term optimality within a disturbance-driven tube, while inheriting stability and feasibility guarantees from P-MPC. An ancillary control strategy (such as the one in [17]) is ultimately employed to reject these disturbances.

The PC-MPC architecture integrates long-term planning capabilities of P-MPC with responsiveness and short-term optimality of C-MPC, while preserving stability and recursive feasibility. The primary distinction between conventional nonlinear TMPC and the PC-MPC architecture lies in the role of the nominal components: In standard TMPC, an ancillary controller ensures stability by keeping the state within a tube located in the state space through the nominal controller. Thus, reducing the tube size is essential to enhancing flexibility in the nominal trajectory, thereby improving the performance (see tightened state admissibility constraint (4.4d)). In contrast, within the PC-MPC architecture, C-MPC enhances the optimality, whereas P-MPC ensures stability and recursive feasibility by generating a suitable modeling error tube. Consequently, enhancing the flexibility of C-MPC and, thus, the system performance, requires an enlarged modeling error tube.

Next, the formulations of P-MPC and C-MPC are presented. The actual state at time step k is denoted by $\boldsymbol{x}_k$, which is guided by the actual control input $\boldsymbol{u}_k$. The nominal state and control input of P-MPC and C-MPC are represented by $\boldsymbol{z}_k^{\mathrm{P}}, \boldsymbol{v}_k^{\mathrm{P}}$ and $\boldsymbol{z}_k^{\mathrm{C}}, \boldsymbol{v}_k^{\mathrm{C}}$, respectively. The prediction horizon and cross section of the tube — a robust positive invariant set for rejecting uncertainties — for P-MPC and C-MPC are N^{P}, $\mathcal{E}^{\mathrm{P}}$ and N^{C}, $\mathcal{E}^{\mathrm{C}}$, respectively. The actual state and input of P-MPC are given by $\boldsymbol{x}_k^{\mathrm{P}}$ and $\boldsymbol{u}_k^{\mathrm{P}}$.

4.3.1. PARENT MPC

The formulation of Parent MPC (P-MPC) is based on standard linear TMPC [16]. Rejecting the additive disturbances $\boldsymbol{w}_k$ affecting the original system modeled by (4.1) is delegated to C-MPC. Consequently, external disturbances do not appear in the formulation of P-MPC, which is designed for the following dynamic model:

$$\boldsymbol{x}_{k+1}^{\mathrm{P}} = A\boldsymbol{x}_k^{\mathrm{P}} + B\boldsymbol{u}_k^{\mathrm{P}} + \boldsymbol{w}_k^{\mathrm{P}} \tag{4.6}$$

with $\boldsymbol{w}_k^{\mathrm{P}} = g\left(\boldsymbol{x}_k^{\mathrm{P}}\right)$, i.e., both the additive disturbance $\boldsymbol{w}_k$ and the nonlinear term $g\left(\boldsymbol{x}_k^{\mathrm{P}}\right)$ as a controllable part of the dynamics are eliminated. Instead, the nonlinear term is treated as a bounded additive disturbance $\boldsymbol{w}_k^{\mathrm{P}}$. Since $g(\cdot)$ is Lipschitz continuous and $g(\mathbf{0}) = \mathbf{0}$, there is $\left\|g\left(\boldsymbol{x}_k^{\mathrm{P}}\right)\right\| \leq L\boldsymbol{x}_k^{\mathrm{P}}$. Given the boundedness of the state space $\mathcal{X}$, this ensures that $g\left(\boldsymbol{x}_k^{\mathrm{P}}\right)$ is also bounded.

The model in (4.6), therefore, structurally resembles a linear system subject to bounded additive disturbances, consistent with the framework of TMPC in [16]. The nominal component of P-MPC then follows a disturbance-free version of (4.6), given by:

$$\boldsymbol{z}_{k+1}^{\mathrm{P}} = A\boldsymbol{z}_k^{\mathrm{P}} + B\boldsymbol{v}_k^{\mathrm{P}} \tag{4.7}$$

In this context, the error $\boldsymbol{e}_k^{\mathrm{P}}$ between the actual model (4.6) and the nominal model (4.7) of P-MPC is defined as:

$$\boldsymbol{e}_k^{\mathrm{P}} := \boldsymbol{x}_k^{\mathrm{P}} - \boldsymbol{z}_k^{\mathrm{P}} \tag{4.8}$$

The simplified design of P-MPC enables its nominal optimization problem to be cast as a convex or quadratic program, significantly reducing computational complexity.

In linear TMPC, the nominal control input is augmented with an ancillary control law in the form of linear state error feedback for regulating the modeling error $\boldsymbol{e}_k^{\mathrm{P}}$. Accordingly, the control input in P-MPC is defined as:

$$\boldsymbol{u}_k^{\mathrm{P}} = \boldsymbol{v}_k^{\mathrm{P}} + K^{\mathrm{P}}\boldsymbol{e}_k^{\mathrm{P}} \tag{4.9}$$

where K^{P} is a feedback gain matrix selected such that $A + BK^{\mathrm{P}}$ is stable (i.e., all eigenvalues lie strictly within the unit circle). In general, the matrix K^{P} is not unique.

The control law in (4.9) is not directly applied to the actual system. Instead, as is detailed in Section 4.3.2, it serves as a baseline input around which the nominal C-MPC policy is constructed, and warm-starts the C-MPC nominal optimization. Moreover, at the P-MPC level, the choice of the feedback gain matrix K^{P} is primarily important for the construction of the modeling error tube with cross section $\mathcal{E}^{\mathrm{P}}$. This tube is subsequently injected into C-MPC as a sequence of stage-wise constraints on its nominal state, omitting the need for explicit terminal sets. This tube additionally plays a central role in establishing the recursive feasibility for the PC-MPC architecture.

NOMINAL PROBLEM OF P-MPC

The nominal optimization problem of P-MPC spans the Parent prediction horizon N^{P}, and incorporates both the Parent tube with cross section $\mathcal{E}^{\mathrm{P}}$ and the Child tube with cross section $\mathcal{E}^{\mathrm{C}}$. This optimization problem at time step k is given by:

$$V^{\mathrm{P}}\left(\boldsymbol{z}^{\mathrm{C}}_{k|k-1}\right) = \min_{\tilde{\boldsymbol{z}}^{\mathrm{P}}_{k:k+N^{\mathrm{P}}},\, \tilde{\boldsymbol{v}}^{\mathrm{P}}_{k:k+N^{\mathrm{P}}-1}} \left\{ \sum_{i=k}^{k+N^{\mathrm{P}}-1} \left({\boldsymbol{z}^{\mathrm{P}}_{i|k}}^{\top} Q \boldsymbol{z}^{\mathrm{P}}_{i|k} + {\boldsymbol{v}^{\mathrm{P}}_{i|k}}^{\top} R \boldsymbol{v}^{\mathrm{P}}_{i|k} \right) + {\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k}}^{\top} P \boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k} \right\} \tag{4.10a}$$

such that for $i = k, \ldots, k+N^{\mathrm{P}}-1$:

$$\boldsymbol{z}^{\mathrm{C}}_{k|k-1} \in \left\{ \boldsymbol{z}^{\mathrm{P}}_{k} \right\} \oplus \mathcal{E}^{\mathrm{P}} \tag{4.10b}$$

$$\boldsymbol{z}^{\mathrm{P}}_{i+1|k} = A \boldsymbol{z}^{\mathrm{P}}_{i|k} + B \boldsymbol{v}^{\mathrm{P}}_{i|k} \tag{4.10c}$$

$$\boldsymbol{z}^{\mathrm{P}}_{i+1|k} \in \left(\mathcal{X} \ominus \mathcal{E}^{\mathrm{C}} \right) \ominus \mathcal{E}^{\mathrm{P}} \tag{4.10d}$$

$$\boldsymbol{v}^{\mathrm{P}}_{i|k} \in \mathcal{V}^{\mathrm{C}} \ominus K^{\mathrm{P}} \mathcal{E}^{\mathrm{P}} \tag{4.10e}$$

$$\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k} \in \mathcal{Z}^{\mathrm{f,P}} \tag{4.10f}$$

This problem formulation enhances flexibility by encoding the initial Parent nominal state $\boldsymbol{z}^{\mathrm{P}}_{k}$ as a decision variable. As enforced by constraint (4.10b), this state should be determined such that the most recent estimate $\boldsymbol{z}^{C}_{k|k-1}$ of the Child nominal state for time step k lies within the current Parent tube. Since the Parent tube is, by design, a robust positive invariant set, all future Parent nominal states are then guaranteed to remain within this tube. Note that the nominal Child state $\boldsymbol{z}^{\mathrm{C}}_{k}$ is a decision variable in C-MPC (see Section 4.3.2), which is solved only after the P-MPC optimization problem. This explains deploying the most recent available estimation $\boldsymbol{z}^{\mathrm{C}}_{k|k-1}$ as a proxy for the Child nominal state at time step k in constraint (4.10).

The evolution of the nominal state in P-MPC is governed by the simplified dynamics (4.7), restated in constraint (4.10c). To ensure constraint satisfaction despite model simplifications, constraints (4.10d) and (4.10e) impose tightened bounds on state and control input trajectories, respectively. While the trajectories generated by P-MPC do not directly steer the actual system, they indirectly influence the actual trajectories by steering the nominal state trajectory of C-MPC (i.e., the center of the Child tube) to ensure it remains stage-wise within the Parent tube. This condition, as detailed in Section 4.3.2, is essential to eliminating terminal constraints from the C-MPC formulation. Moreover, C-MPC relies on a simplified model of the system that excludes the additive disturbances. This implies that constraint tightening is necessary to always guarantee state admissibility. Accordingly, as encoded in constraint (4.10d), the admissible state set $\mathcal{X}$ is first shrunk by the shape of set $\mathcal{E}^{\mathrm{C}}$ — cross section of the disturbance-driven tube of C-MPC — and is further eroded by set $\mathcal{E}^{\mathrm{P}}$ — cross section of the modeling error tube. This ensures that the set $\left\{ \boldsymbol{z}^{\mathrm{P}}_{i|k} \right\} \oplus \mathcal{E}^{\mathrm{P}} \oplus \mathcal{E}^{\mathrm{C}}$, which contains the actual state trajectory for all $i = k, \ldots, k+N^{\mathrm{P}}-1$, remains a subset of admissible set $\mathcal{X}$. A

similar tightening is applied to the nominal control inputs of P-MPC. In constraint (4.10e), the admissible control input set $\mathcal{U}$ is replaced by a tightened set $\mathcal{V}^{\mathrm{C}}$, designed based on the actual control input policy of C-MPC for the admissibility of its nominal control inputs (see [17] for details), and is further eroded by the set $K^{\mathrm{P}}\mathcal{E}^{\mathrm{P}}$, thereby reserving room for augmenting the Parent ancillary control law later.

To ensure stability, the terminal state $\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k}$ in P-MPC must lie within a terminal set $\mathcal{Z}^{\mathrm{f,P}}$, enforced via constraint (4.10f). The associated terminal cost ${\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k}}^{\top} P \boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k}$ is defined in quadratic form, with matrix P a constant, positive definite matrix obtained by solving the following discrete-time Lyapunov equation:

$$\left(A+BK^{\mathrm{f,P}}\right)^{\top} P \left(A+BK^{\mathrm{f,P}}\right) + Q + {K^{\mathrm{f,P}}}^{\top} R K^{\mathrm{f,P}} = P \tag{4.11}$$

Matrices $A, B, K^{\mathrm{f,P}}$ define the closed-loop dynamics under a terminal control law that follows a state feedback policy with gain $K^{\mathrm{f,P}}$, while Q and R are the stage cost matrices. Note that the feedback gains K^{P}, used in (4.9) for the ancillary control law of P-MPC, and $K^{\mathrm{f,P}}$, used to determine the terminal cost, both need to stabilize the simplified nominal model (4.7). However, they do not necessarily need to be identical. Their independent design can introduce flexibility, as gain K^{P} influences the shape and size of $\mathcal{E}^{\mathrm{P}}$, i.e., the modeling error set, which constrains the nominal trajectory of C-MPC, while gain $K^{\mathrm{f,P}}$ affects the size of the terminal set $\mathcal{Z}^{\mathrm{f,P}}$ for P-MPC, which determines the admissible endpoint of the nominal trajectory of P-MPC. This terminal set may be defined as a level set of the terminal cost, often taking the form of a polyhedron or an ellipsoid, and obtained by imposing ${\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k}}^{\top} P \boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}|k} < c$, where $c > 0$ is a scalar that determines the size of the terminal set.

DESIGNING THE MODELING ERROR TUBE CROSS SECTION $\mathcal{E}^{\mathrm{P}}$

The modeling error tube in P-MPC should be a robust positive invariant set with cross section $\mathcal{E}^{\mathrm{P}}$, such that $\boldsymbol{e}^{\mathrm{P}}_k \in \mathcal{E}^{\mathrm{P}}$ implies $\boldsymbol{e}^{\mathrm{P}}_i \in \mathcal{E}^{\mathrm{P}}$, for $i > k$. Various state-of-the-art algorithms, e.g., in [15, 20], are available for designing cross section of such sets, taking into account that this set is not necessarily unique for a given system subject to given disturbance bounds.

In conventional TMPC, determining optimal shape and size for the tube entails a fundamental trade-off. Smaller tubes, on the one hand, increase flexibility in designing the nominal trajectory by enlarging the tightened state admissible set (see constraints (4.4d) and (4.10d)). On the other hand, a smaller tube often necessitates a larger feedback gain matrix K^{P} to ensure robust stability. This, in turn, tightens the admissible control input space (see (4.10e)). Thus, either design choice may lead to an overall degradation of closed-loop performance.

Since the tube of TMPC encapsulates the nominal state trajectory of C-MPC, shrinking this tube — while expanding the admissible state space for the nominal state trajectory of P-MPC — has the opposite effect on the nominal state trajectory of C-MPC. The trajectory of C-MPC, however, should enhance optimality. Hence, increased flexibility in state space exploration is highly desirable. Accordingly, the PC-MPC architecture pursues a fundamentally different

design objective than that of conventional TMPC: It enlarges the cross section $\mathcal{E}^{\mathrm{P}}$ of the modeling error tube, provided that the computational demand remains tractable for solving the online optimization of C-MPC. Since, by the end of the primary control phase, the nominal C-MPC trajectory lies within set $\mathcal{E}^{\mathrm{P}}$, this set becomes the feasible exploration region for the conventional TMPC in the secondary phase of control. Therefore, the design trade-offs for set $\mathcal{E}^{\mathrm{P}}$ should also account for the computational demands of the conventional TMPC. In practice, for many systems described by (4.1), with reasonably simple terminal sets, if C-MPC (which operates under stricter constraints) is computationally affordable over horizon N^{C}, then so is the conventional TMPC.

The enlarged tube of P-MPC serves as a stage-wise constraint for C-MPC, thereby expanding its feasible region and enabling it to pursue performance-enhancing control strategies. Note that increasing the size of the $\mathcal{E}^{\mathrm{P}}$ would never result in the C-MPC finding a less optimal trajectory. However, increasing the search space for the C-MPC may be computationally prohibitive. To preserve the stability and recursive feasibility guarantees of the overall framework, the set $\mathcal{E}^{\mathrm{P}}$ should satisfy:

$$\mathcal{E}^{\mathrm{P}} \subseteq \mathcal{Z}^{\mathrm{C}}_{N^{\mathrm{C}}} \tag{4.12}$$

where $\mathcal{Z}^{\mathrm{C}}_{N^{\mathrm{C}}}$ is the N^{C}-step controllable set for the nominal component of C-MPC that steers its nominal state – within N^{C} time steps – into a terminal region $\mathcal{Z}^{\mathrm{f,C}}$, which is designed offline by tightening the actual terminal state set $\mathcal{X}^{\mathrm{f}}$ to account for the impact of the additive disturbances on the actual system.

This approach draws inspiration from applications of TMPC in passenger car systems [18] and planar quadrotor drones [21].

Remark. Whenever computing the maximal set $\mathcal{Z}^{\mathrm{C}}_{N^{\mathrm{C}}}$ is intractable (e.g., due to complex nonlinear dynamics), a control-invariant subset, e.g., $\mathcal{Z}^{\mathrm{f,C}}$, may be used instead. Larger sets yield a larger Parent tube, where this increases the flexibility of C-MPC in exploring its search space.

BALANCING THE COMPUTATIONAL OVERHEAD OF P-MPC

Since the model of P-MPC is linear and its nominal optimization problem is convex (not necessarily quadratic due to the terminal constraint), P-MPC can, in view of computational costs, accommodate a larger prediction horizon than C-MPC. Larger controllable sets can be reached by sufficiently increasing the prediction horizon [5]. To reduce the computational burden of P-MPC while still enabling a large Parent prediction horizon N^{P}, two complementary strategies are proposed as outlined next.

Reduced update rate: P-MPC does not control the system, so it may be solved less frequently than C-MPC to reduce computational overhead. This requires that the updated

Parent tube always covers the full Child prediction horizon, i.e., the number of time steps between two consecutive control updates by P-MPC be strictly smaller than $N^{\mathrm{P}} - N^{\mathrm{C}} + 2$, assuming identical sampling times for both P-MPC and C-MPC.

Coarser planning sampling: Using larger sampling times, P-MPC can extend its horizon without adding more optimization variables. For stability analysis, the underlying dynamics should use the base sampling time. This can be enforced by holding control inputs constant over fixed intervals, with the associated variables and constraints removed at implementation.

4.3.2. CHILD MPC

Child MPC (C-MPC) solves a nonlinear TMPC problem to directly steer the system, optimizing the same cost function structure as P-MPC, instead of tracking its state trajectory. P-MPC shares its nominal control input trajectory and tube with C-MPC, which must ensure that its nominal state trajectory remains within the tube defined by P-MPC. By exploiting its flexibility, C-MPC discovers trajectories that enhance optimality beyond the conservative solution provided by P-MPC. The nominal optimization problem solved by C-MPC is formulated by:

$$V^{\mathrm{C}}\left(\boldsymbol{x}_k, \tilde{\boldsymbol{z}}^{\mathrm{P}}_{k:k+N^{\mathrm{P}}}, \tilde{\boldsymbol{v}}^{\mathrm{P}}_{k:k+N^{\mathrm{P}}-1}\right) = \min_{\tilde{\boldsymbol{z}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}}, \Delta\tilde{\boldsymbol{v}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}-1}} \left\{ \sum_{i=k}^{k+N^{\mathrm{C}}-1} \left({\boldsymbol{z}^{\mathrm{C}}_{i|k}}^{\top} Q\, \boldsymbol{z}^{\mathrm{C}}_{i|k} + {\boldsymbol{v}^{\mathrm{C}}_{i|k}}^{\top} R\, \boldsymbol{v}^{\mathrm{C}}_{i|k} \right) + {\boldsymbol{z}^{\mathrm{C}}_{k+N^{\mathrm{C}}|k}}^{\top} P\, \boldsymbol{z}^{\mathrm{C}}_{k+N^{\mathrm{C}}|k} \right\} \tag{4.13a}$$

such that for $i = k, \ldots, k+N^{\mathrm{C}}-1$:

$$\boldsymbol{x}_k \in \left\{\boldsymbol{z}^{\mathrm{C}}_k\right\} \oplus \mathcal{E}^{\mathrm{C}} \tag{4.13b}$$

$$\boldsymbol{z}^{\mathrm{C}}_{i+1|k} = A\boldsymbol{z}^{\mathrm{C}}_{i|k} + g\left(\boldsymbol{z}^{\mathrm{C}}_{i|k}\right) + B\boldsymbol{v}^{\mathrm{C}}_{i|k} \tag{4.13c}$$

$$\boldsymbol{z}^{\mathrm{C}}_{j|k} \in \left\{\boldsymbol{z}^{\mathrm{P}}_{j|k}\right\} \oplus \mathcal{E}^{\mathrm{P}} \qquad \text{for } j = k, \ldots, k+N^{\mathrm{C}} \tag{4.13d}$$

$$\boldsymbol{v}^{\mathrm{C}}_{i|k} = \boldsymbol{v}^{\mathrm{P}}_{i|k} + K^{\mathrm{P}}\left(\boldsymbol{x}^{\mathrm{P}}_{i|k} - \boldsymbol{z}^{\mathrm{P}}_{i|k}\right) + \Delta\boldsymbol{v}^{\mathrm{C}}_{i|k} \tag{4.13e}$$

$$\Delta\boldsymbol{v}^{\mathrm{C}}_{i|k} \in \left(\mathcal{V}^{\mathrm{C}} \ominus K^{\mathrm{P}}\mathcal{E}^{\mathrm{P}}\right) \ominus \left\{\boldsymbol{v}^{\mathrm{P}}_{i|k}\right\} \tag{4.13f}$$

This optimization is warm-started by the (virtual) solution of P-MPC, i.e., by $\tilde{\boldsymbol{z}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}} = \tilde{\boldsymbol{x}}^{\mathrm{P}}_{k:k+N^{\mathrm{C}}}$ and $\Delta\tilde{\boldsymbol{v}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}-1} = [0, \ldots, 0]^{\top}$ — both spanning the first N^{C} time steps.

Constraint (4.13b) ensures that current nominal state $\boldsymbol{z}^{\mathrm{C}}_k$ of C-MPC, included as a decision variable, is determined such that current measured state $\boldsymbol{x}_k$ of the system is inside the tube $\left\{\boldsymbol{z}^{\mathrm{C}}_k\right\} \oplus \mathcal{E}^{\mathrm{C}}$ of C-MPC. Constraint (4.13c) implies that the nominal component of C-MPC operates based on the original nonlinear dynamic model (4.1), omitting the external disturbance $\boldsymbol{w}_k$. Thus, the tube $\mathcal{E}^{\mathrm{C}}$ of C-MPC is designed to account for rejecting these

disturbances for the actual system. In essence, the PC-MPC architecture allows C-MPC to avoid designing and deploying explicit terminal sets, a typically challenging task for nonlinear MPC frameworks. Instead, a sequence of tightened stage-wise state constraints, as given in (4.13d), which ensures that the nominal state trajectory of C-MPC remains within the tube of P-MPC. These constraints, when satisfied alongside constraint (4.10d) within the P-MPC nominal optimization loop, ensure safe tightening of the C-MPC nominal state set, such that the actual states remain within the original admissible set $\mathcal{X}$. Constraint (4.13e) defines the nominal C-MPC law, where the P-MPC input (4.9) acts as a baseline, and C-MPC input increment $\Delta \boldsymbol{v}^{\mathrm{C}}_{i|k}$ is treated as a decision variable. Constraint (4.13f) ensures that the resulting C-MPC nominal inputs remain within the tightened admissible input set $\mathcal{V}^{\mathrm{C}}$.

If $\boldsymbol{x}_k \notin \left(\mathcal{Z}^{\mathrm{C}}_{N^{\mathrm{C}}} \oplus \mathcal{E}^{\mathrm{C}}\right)$ holds, C-MPC invokes P-MPC to re-solve its optimization problem, initialized at the most recent admissible state, to redefine the Parent tube.

4.3.3. STABILITY AND RECURSIVE FEASIBILITY

An MPC optimization problem is recursively feasible if, whenever a solution exists for an initial state, a feasible solution also exists for all subsequent states along the resulting closed-loop trajectory [2]. A given robust positive invariant set $\mathcal{O}$ containing the origin is robustly exponentially stable for system (4.1), if, for all admissible disturbances $\boldsymbol{w}_k \in \mathcal{W}$, state trajectories starting within a certain bounded region converge exponentially towards set $\mathcal{O}$ [16]. The tube of P-MPC $\mathcal{E}^{\mathrm{P}}$ serves as such a robust positive invariant set $\mathcal{O}$ within the PC-MPC architecture for the nominal trajectory of C-MPC. Thus, once the C-MPC nominal trajectory enters set $\tilde{\mathcal{Z}}^{\mathrm{C}}_{N^{\mathrm{C}}}$, a dual control strategy — such as the original TMPC formulation in (4.4)) — may be deployed. Accordingly, the PC-MPC architecture is established, governed by (4.10)–(4.13), to be guaranteed to exponentially steer the nominal state trajectory towards set $\tilde{\mathcal{Z}}^{\mathrm{C}}_{N^{\mathrm{C}}}$. This, in turn, guarantees that the actual state trajectory reaches the actual terminal set $\mathcal{X}^{\mathrm{f}}$. To do so, it is shown that the Parent tube, which is by design a robust positive invariant set for P-MPC, also remains robustly invariant under the full PC-MPC architecture. The following two theorems are proven.

Theorem 7. **Robust invariance of the modeling error set $\mathcal{E}^{\mathrm{P}}$ for C-MPC nominal state trajectory:** Under the PC-MPC architecture, the modeling error set $\mathcal{E}^{\mathrm{P}}$ — defining the cross section of the Parent tube — constitutes a robust positive invariant set for the nominal state trajectory of C-MPC.

Proof. The stage-wise state constraint (4.13d) ensures that the nominal state trajectory of C-MPC remains entirely within the Parent tube, making this tube a robust positive invariant set for this trajectory. Designing the modeling error set according to (4.12), guarantees that the nominal P-MPC trajectory is steerable to the origin. Once this occurs, the Parent tube effectively becomes the untranslated modeling error set $\mathcal{E}^{\mathrm{P}}$. Consequently, $\mathcal{E}^{\mathrm{P}}$ itself serves as a robust positive invariant set for the nominal C-MPC trajectory, completing the proof. □

For Theorem 7 to be applicable, the nominal C-MPC problem should admit at least one feasible trajectory. The following theorem establishes the guaranteed existence of such a trajectory.

Theorem 8. **Feasibility of the P-MPC-based warm start for nominal C-MPC:** The solution provides as a warm start for the C-MPC nominal problem (4.13) by the P-MPC layer, under the ancillary control law (4.9), always yields a feasible solution for nominal C-MPC, ensuring feasibility for the nominal C-MPC problem, whenever the P-MPC problem is itself feasible.

Proof. To prove feasibility, one should show that the warm-started trajectories $\tilde{\boldsymbol{z}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}} = \tilde{\boldsymbol{x}}^{\mathrm{P}}_{k:k+N^{\mathrm{C}}}$ and $\tilde{\boldsymbol{v}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}-1} = \tilde{\boldsymbol{u}}^{\mathrm{P}}_{k:k+N^{\mathrm{C}}-1}$ satisfy all constraints of the C-MPC problem, i.e., (4.13b)–(4.13d) and (4.13f), with the control law given by (4.13e). Assuming feasibility for the previous time step $k-1$, from (4.13b), $\boldsymbol{x}_{k-1} \in \left\{\boldsymbol{z}^{\mathrm{C}}_{k-1}\right\} \oplus \mathcal{E}^{\mathrm{C}}$ holds. The tube of C-MPC is robust positive invariant for the actual state trajectory. Hence, $\boldsymbol{x}_k \in \left\{\boldsymbol{z}^{\mathrm{C}}_{k|k-1}\right\} \oplus \mathcal{E}^{\mathrm{C}}$ also holds. Moreover, based on (4.10) for the P-MPC problem $\boldsymbol{z}^{\mathrm{C}}_{k|k-1} = \boldsymbol{x}^{\mathrm{P}}_{k}$ holds. Therefore, $\boldsymbol{x}_k \in \left\{\boldsymbol{x}^{\mathrm{P}}_{k}\right\} \oplus \mathcal{E}^{\mathrm{C}}$ holds. Moreover, with the warm-start input sequence, i.e., $\tilde{\boldsymbol{v}}^{\mathrm{C}}_{k:k+N^{\mathrm{C}}-1} = \tilde{\boldsymbol{u}}^{\mathrm{P}}_{k:k+N^{\mathrm{C}}-1}$, constraint (4.13f) boils down to the P-MPC input constraint (4.10e), which is satisfied by the assumption of feasibility of the P-MPC problem. The dynamics of the warm-started C-MPC trajectory align with the P-MPC dynamics (4.6), which is equivalent to constraint (4.13c) in C-MPC when warm-started by $\tilde{\boldsymbol{x}}^{\mathrm{P}}_{k:k+N^{\mathrm{C}}}$. Finally, the tube $\left\{\boldsymbol{z}^{\mathrm{P}}_{i|k}\right\} \oplus \mathcal{E}^{\mathrm{P}}$ of P-MPC is robust positive invariant, i.e., it contains all actual P-MPC states — implying satisfaction of constraint (4.13d). □

Theorems 7 and 8 jointly establish that the PC-MPC guarantees the existence of at least one feasible solution that steers system (4.1) into a designated terminal set, thereby ensuring stabilizability.

4.3.4. SPECIAL CASES

Thus far, this Chapter has been focusing on the general implementation of the PC-MPC architecture under the assumption that the original system can be stabilized via conventional TMPC using a linear model with bounded modeling error. This section considers simpler, yet common, special cases.

DETERMINISTIC MPC

A widely studied variant is the deterministic MPC problem, where the disturbance in (4.1) is assumed to be zero. In this case, C-MPC becomes a standard deterministic MPC and solves

the following deterministic optimization problem:

$$V^{\mathrm{C}}(\boldsymbol{x}_k) = \min_{\tilde{\boldsymbol{x}}_{k+1:k+N^{\mathrm{C}}}, \Delta\tilde{\boldsymbol{u}}_{k:k+N^{\mathrm{C}}-1}} \left\{ \sum_{i=k}^{k+N^{\mathrm{C}}-1} \left(\boldsymbol{x}_i^\top Q \boldsymbol{x}_i + \boldsymbol{u}_i^\top R \boldsymbol{u}_i \right) + \boldsymbol{x}_{k+N^{\mathrm{C}}|k}^\top P \boldsymbol{x}_{k+N^{\mathrm{C}}|k}^\top \right\} \tag{4.14a}$$

such that for $i = k, \ldots, k+N^{\mathrm{C}}-1$:

$$\boldsymbol{x}_{i+1|k} = f\left(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}\right) \tag{4.14b}$$

$$\boldsymbol{x}_{i+1|k} \in \left\{ \boldsymbol{z}^{\mathrm{P}}_{i+1|k} \right\} \oplus \mathcal{E}^{\mathrm{P}} \tag{4.14c}$$

$$\boldsymbol{u}_{i|k} = \boldsymbol{u}^{\mathrm{P}}_{i|k} + \Delta\boldsymbol{u}_{i|k} \tag{4.14d}$$

$$\boldsymbol{u}_{i|k} \in \mathcal{U} \tag{4.14e}$$

The formulation of P-MPC remains unchanged, except for constraints (4.10d) and (4.10e), which will not involve any disturbance-driven error set (i.e., $\boldsymbol{z}^{\mathrm{P}}_{i+1} \in \mathcal{X} \ominus \mathcal{E}^{\mathrm{P}}$ and $\boldsymbol{v}^{\mathrm{P}}_i \in \mathcal{U} \ominus K\mathcal{E}^{\mathrm{P}}$ for $i = k, \ldots, k+N^{\mathrm{P}}-1$). Finally, in the second phase, when the actual state reaches $\tilde{\mathcal{Z}}^{\mathrm{C}}_{N^{\mathrm{C}}}$, a deterministic MPC controller replaces the PC-MPC architecture.

LINEAR TMPC

Even without disturbances and modeling errors, the MPC problem can become computationally challenging when the number of decision variables or constraints is large, causing solvers to fail within limited time budgets. In such cases, the PC-MPC architecture offers value. Since both P-MPC and C-MPC components share the same system model, P-MPC can operate with a larger horizon and/or smaller sampling time, effectively planning further ahead, especially when the P-MPC optimization does not need to be solved every iteration. Despite the absence of modeling error, a Parent tube with cross section $\mathcal{E}^{\mathrm{P}}$ is designed as a positive invariant set (rather than a robust one), as well as an ancillary control law that establishes the positive invariance of $\mathcal{E}^{\mathrm{P}}$. Using solvers for convex or quadratic programs ensures computational tractability of C-MPC.

4.3.5. GENERALIZATION TO MULTI-LEVEL PC-MPC ARCHITECTURE

While a single P-MPC layer can extend the planning horizon and enhance feasibility, it may still fall short in steering the system to the terminal set — particularly in scenarios that demand large horizons and many decision variables due to complex dynamics or tight constraints. To address this, a multi-level PC-MPC architecture can be adopted, where multiple P-MPC systems are stacked in a hierarchical structure. In such a recursive structure, each P-MPC serves as the Child to a higher-level Parent and itself operates as a TMPC augmented with an ancillary control law, as described in Section 4.3.4. Since each level adheres to the original design assumptions, this architecture remains theoretically valid at all depths.

As with the bi-level scheme, once the nominal state trajectory of a lower-level P-MPC enters the untranslated tube of its Parent, the top-most Parent layer is discarded, and control is delegated to the next layer. For intermediate layers, the required tube is generally larger than that used for the original C-MPC

However, adding MPC layers introduces a critical trade-off: While larger Parent tubes facilitate feasibility and robustness, adding more layers increases the number of tubes where their Minkowski sum must remain within the state admissible set $\mathcal{X}$. Therefore, the design must balance warm-start feasibility, computation, and constraint satisfaction across all levels.

4.4. CASE STUDIES

This section presents two numerical case studies designed to demonstrate the performance of the PC-MPC architecture. The first case study highlights its advantage over standard TMPC in linear systems, particularly in terms of feasibility and robustness. The second case study shows how this architecture can restore feasibility in a nonlinear control scenario, where standard TMPC fails due to reachability limitations.

- **Case study 1:** A PC-MPC architecture is implemented for a linear system showcasing its ability to maintain robustness against disturbances while improving feasibility and reducing conservativeness compared to standard TMPC (The code is published in [22]).
- **Case study 2:** Based on the nonlinear TMPC formulation in [17], this case modifies the problem such that the original TMPC becomes infeasible due to a restrictive terminal constraint. The PC-MPC architecture is then introduced to extend the reachability set and to recover feasibility (The code is provided in [23]).

All simulations are conducted on a single computer equipped with an Intel Core i9-13900HX CPU, 16 GB of RAM, and an NVIDIA GeForce RTX 4070 GPU.

4.4.1. LINEAR PROBLEM

A discrete-time double integrator system is considered with additive disturbance, given by:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k + \boldsymbol{w}_k \tag{4.15}$$

This system models a simple kinematic scenario, with the state vector including position and velocity, and the control input corresponding to acceleration. The control objective is to steer the system from the initial state $\boldsymbol{x}_0 = [2700, 0]^\top$ toward the origin, minimizing the cost function:

$$l(\tilde{\boldsymbol{x}}_{0:T^s}, \tilde{u}_{0:T^s}) = \sum_{i=0}^{T^s} \left(\boldsymbol{x}_i^\top \boldsymbol{x}_i + u_i^2 \right) \tag{4.16}$$

where T^{s} is the simulation time. The system is subject to constraints $-50 \leq \boldsymbol{x}[1] \leq 10000$, $|\boldsymbol{x}[2]| \leq 10000$, $|u| \leq 5$, where $[i]$ is used to show element i of a vector. The disturbance vector $\boldsymbol{w}_k$ is unknown and bounded $|\boldsymbol{w}_k| \leq 1$ for all integer values of k.

The following controllers are implemented for comparison:

TMPC: The simplest controller satisfying the design requirements is a standard linear TMPC, as described in [16], which outlines conditions for the existence of both the terminal set and the tube. The tube is computed via the method in [20] with the ancillary control law (4.9), using the gain $K^{\mathrm{P}} = [0.2054, 0.7835]$, obtained by minimizing the following cost function:

$$l(\tilde{\boldsymbol{z}}_{0:T^{\mathrm{s}}}, \tilde{v}_{0:T^{\mathrm{s}}}) = \sum_{i=0}^{T^{\mathrm{s}}} \left(\boldsymbol{z}_i^{\top} \boldsymbol{z}_i + 10 v_i^2 \right) \tag{4.17}$$

This cost function penalizes the input more heavily (i.e., by a factor 10) than the original cost in (4.16). This prevents overly aggressive control that will potentially violate input constraints due to excessive demand from the ancillary controller. The corresponding tube cross section is shown in Figure 4.1. After constraint tightening, the admissible bounds become $-40.77 \leq \boldsymbol{z}_k[1] \leq 99990.77$, $|\boldsymbol{z}_k| \leq 9996.41$, and $|v_k| \leq 2.86$ for all non-negative integer values k. The MPT3 toolbox [24] is used to compute the terminal set, which should be symmetric about the origin. Since the tightened constraint on $\boldsymbol{z}_k[1]$ is asymmetric, it is limited further to its smaller bound, i.e., $|\boldsymbol{z}_k[1]| \leq 40.77$, when designing the terminal set. The terminal cost matrix in (4.4a) is set to $P = \begin{bmatrix} 2.94 & 2.36 \\ 2.36 & 4.36 \end{bmatrix}$. TMPC uses a horizon of 60.

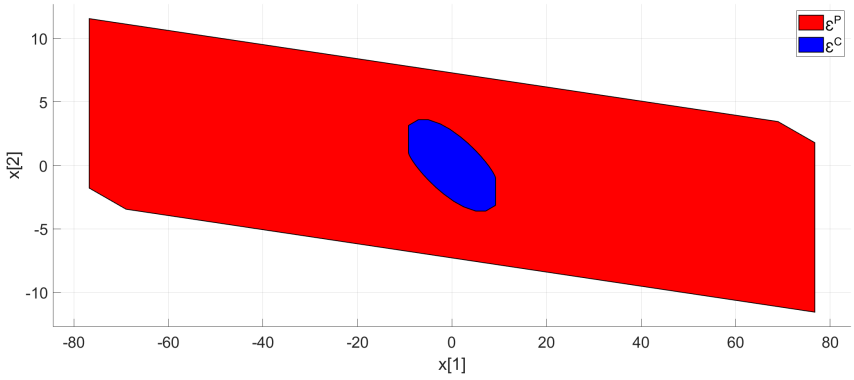


Figure 4.1: Cross section of the disturbance-aware tube used by C-MPC (the same as TMPC) and P-MPC.

Bi-level PC-MPC: Prior to reaching $\mathcal{E}^{\mathrm{P}}$, nominal C-MPC uses stage-wise constraints derived by P-MPC. P-MPC is designed to guide the system towards the target state $[40, 0]^{\top}$, using the

following cost function:

$$l(\tilde{z}^{\mathrm{P}}_{0:T^{\mathrm{s}}}, \tilde{v}^{\mathrm{P}}_{0:T^{\mathrm{s}}}) = \sum_{i=0}^{T^{\mathrm{s}}} \left((\boldsymbol{z}_i^{\mathrm{P}\top} - [40,0])(\boldsymbol{z}_i^{\mathrm{P}} - [40,0]^\top) + (v_i^{\mathrm{P}})^2 \right) \tag{4.18}$$

Targeting $[40,0]^\top$ instead of the origin allows for a larger terminal set $\mathcal{Z}^{\mathrm{f,P}}$ and tube cross section $\mathcal{E}^{\mathrm{P}}$, delegating the goal of reaching the origin to C-MPC (or conventional TMPC in the second control phase). Upon entering $\mathcal{E}^{\mathrm{P}}$, conventional TMPC is applied with a reduced prediction horizon 10. Since the system is linear and disturbance-free at the Parent level, the MPT3 toolbox can be used to compute both the tube cross section and the terminal set. To create the terminal set, the ancillary control law of P-MPC is used with gain matrix $K^{\mathrm{f,P}} = [0.0057, 0.11]$, obtained by solving a feedback design problem with a significantly larger input penalty ($\times 10000$) — allowing to increase the tube size (see Figure 4.1).

After tightening, the bounds become $36 \leq \boldsymbol{z}_k^{\mathrm{P}}[1] \leq 99914$, $|\boldsymbol{z}_k^{\mathrm{P}}[2]| \leq 9984.86$, and $|v_k^{\mathrm{P}}| \leq 2$ for all non-negative integers k. To reduce the number of decision variables, the same input is repeated every 4 time steps. The Parent prediction horizon N^{P} is set to 120, which results in 30 decision variables. To further reduce computation, P-MPC is resolved every 6 time steps.

Results for case study 1 Two 60-step trajectories are simulated, one per controller. The PC-MPC requires 41 time steps before P-MPC becomes inactive. All optimization problems are solved using MATLAB's *quadprog* function [25] with the *active set* algorithm [26], which consistently results in the largest solution speed.

Figure 4.2 compares three controllable sets, showing that the PC-MPC architecture offers the largest set. With zero initial velocity, this architecture successfully solves problems considering more than three times the initial distance from the origin that conventional TMPC can handle. The figure shows that $\mathcal{E}^{\mathrm{P}}$ remains within the N^{C}-controllable set of C-MPC, ensuring stability and recursive feasibility. Conventional TMPC allows enhanced velocity freedom, while the tighter acceleration constraints of the PC-MPC architecture result in earlier deceleration.

Figure 4.3 shows the trajectories of states, control inputs, and cumulative costs: TMPC achieves a lower cost, with a maximum advantage of 117.43 and a final cost 3% lower than the PC-MPC architecture, which, however, is significantly faster. TMPC takes an average of 8.21 ms per iteration, compared to 1.10 ms per iteration for the PC-MPC architecture (or 2.73 ms during P-MPC updates).

4.4.2. NONLINEAR PROBLEM

Case study 2: Nonlinear system In [17], a nonlinear TMPC with a horizon equal to 6 has been proposed, employing a nonlinear ancillary control law. It has been applied to the following system:

$$\begin{bmatrix} \boldsymbol{x}_{k+1}[1] \\ \boldsymbol{x}_{k+1}[2] \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k[2] \\ \sin(\boldsymbol{x}_k[1]) + u_k \end{bmatrix} + \boldsymbol{w}_k \tag{4.19}$$

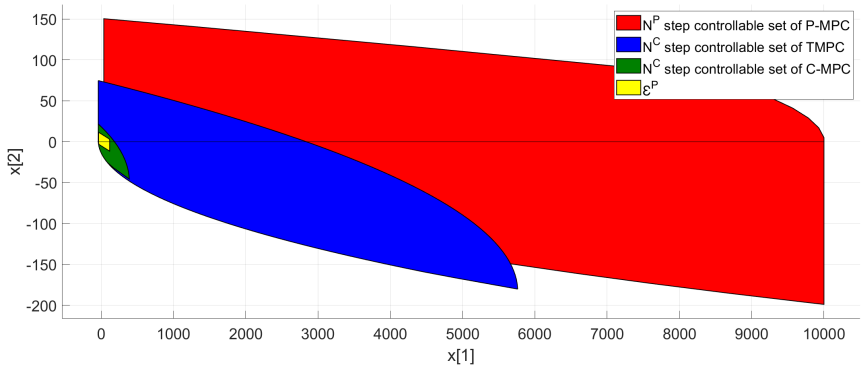


Figure 4.2: Controllable sets and tube cross section for P-MPC

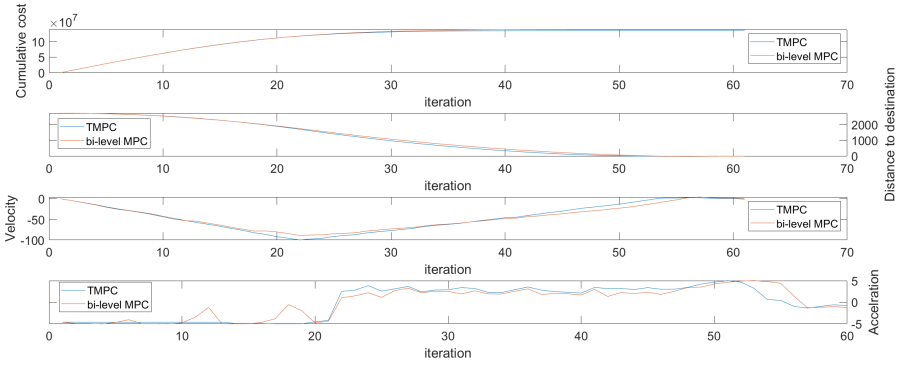


Figure 4.3: Cost, state (including the position and velocity), and control input (i.e., acceleration) over time for TMPC and the PC-MPC architecture.

The state $\boldsymbol{x}_k$ is unconstrained, while the input is limited to $|u_k| \leq 0.5$, and the disturbance satisfies $|\boldsymbol{w}_k| \leq 0.1$. The goal is to steer the system from $\boldsymbol{x}_0 = [1,1]^\top$ towards the origin, with a terminal constraint enforcing convergence to zero.

The same system is considered, but with tighten the input constraint to $|u_k| \leq 0.3$. Under this constraint, no MPC controller can satisfy the terminal constraints within the specified horizon, resulting in infeasibility. The nominal input constraint is set to be $|v_k| \leq 0.25$ for TMPC. While extending the horizon potentially resolves this, it significantly increases computational effort and, due to non-convexity, can degrade solver performance.

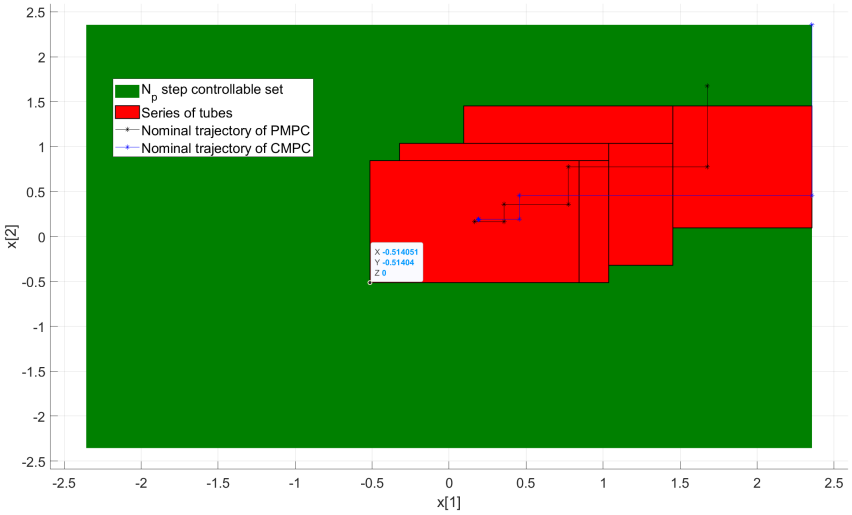


Figure 4.4: The tubes, nominal states of TMPC, and nominal states of C-MPC.

A bi-level PC-MPC architecture using the original TMPC as C-MPC and the following linearized model for P-MPC:

$$\boldsymbol{z}^{\mathrm{C}}_{k+1} = \begin{bmatrix} 0 & 1 \\ 0.46 & 0 \end{bmatrix} \boldsymbol{z}^{\mathrm{C}}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v^{\mathrm{C}}_k + \begin{bmatrix} 0 \\ g\left(\boldsymbol{z}^{\mathrm{C}}_k\right) \end{bmatrix} \tag{4.20}$$

where $g(\boldsymbol{x}_k) = \sin(\boldsymbol{x}_k[1]) - 0.46\boldsymbol{x}_k[1]$. To ensure validity, the state is restricted to $\left|\boldsymbol{z}^{\mathrm{C}}_k\right| \leq 4\pi/3$. These constraints do not reduce the feasibility compared to the original TMPC, which is already infeasible outside this domain. The function $g(\cdot)$ is Lipschitz and bounded, allowing us to treat it as a bounded modeling error for P-MPC, i.e., $\boldsymbol{w}^{\mathrm{P}}_k[1] = 0$ and $\left|\boldsymbol{w}^{\mathrm{P}}_k[2]\right| \leq 0.4$.

A linear ancillary control law is designed using (4.9) with $K^{\mathrm{P}} = [0.051, 0]$, optimized for $Q = I_{2\times 2}$ and $R = 20$. This yields a tube $\mathcal{E}$, such that $|\boldsymbol{e}^{\mathrm{P}}_k| \leq 0.68$. Consequently, the nominal input must satisfy $\left|v^{\mathrm{P}}_k\right| \leq 0.22$. The terminal set is defined as $\left|\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}}\right| \leq 0.68$, with a horizon of $N^{\mathrm{P}} = 10$, and a terminal cost given by ${\boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}}}^\top \mathrm{diag}(1.15, 2.15) \boldsymbol{z}^{\mathrm{P}}_{k+N^{\mathrm{P}}}$. For C-MPC, the

controller from [17] is reused, replacing the terminal constraint with time-varying constraints from P-MPC, as described in (4.13). Optimization problems are solved using *quadprog* or *fmincon* [27] with the Sequential Quadratic Programming (SQP) algorithm for nonlinear problems.

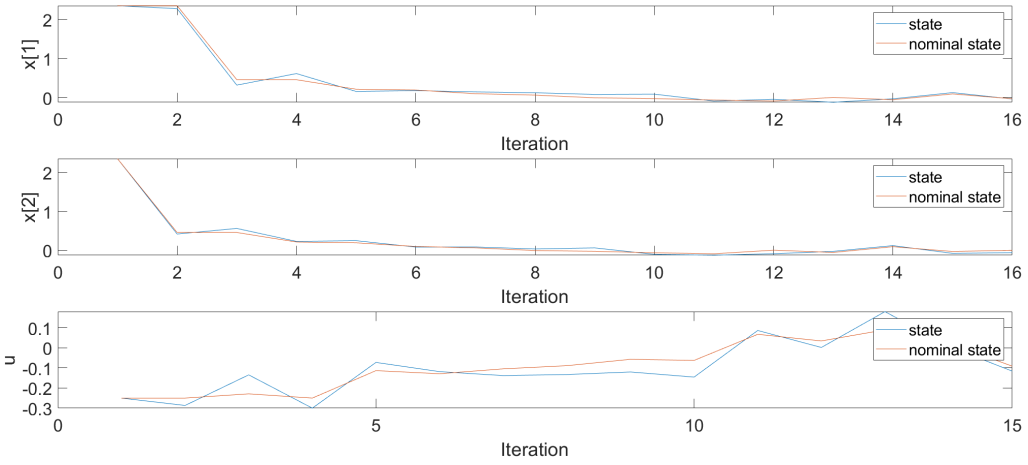


Figure 4.5: Trajectory of nominal and actual states and inputs of C-MPC

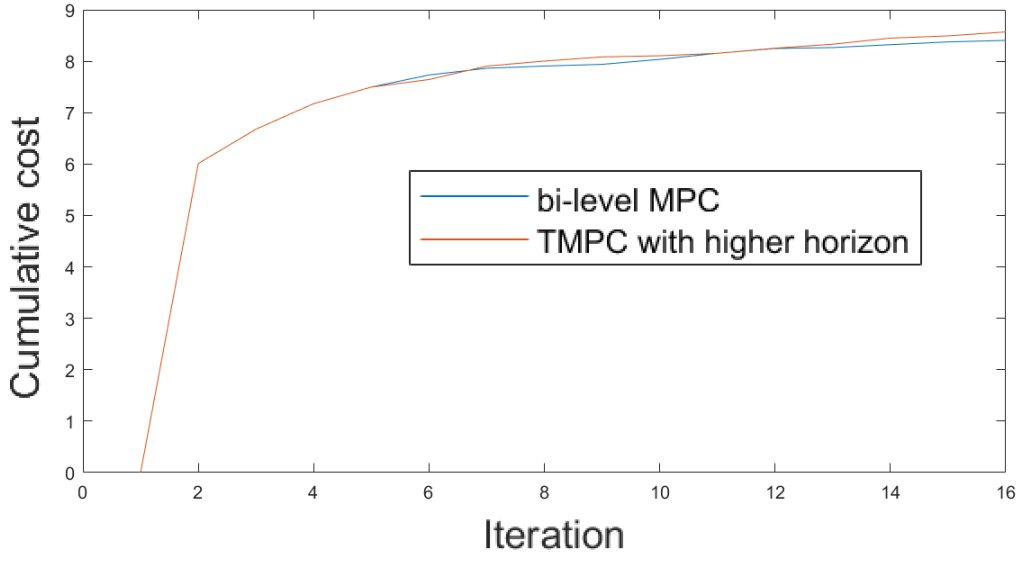


Figure 4.6: Cumulative cost of the PC-MPC architecture and original TMPC with a larger horizon.

Results for case study 2 Both the PC-MPC architecture and TMPC with an extended horizon are implemented to steer the system from $\boldsymbol{x}_0 = [\pi 4/3, \pi 4/3]^\top$ towards the origin. Figure 4.4 shows the tubes generated by P-MPC and the nominal trajectory determined by C-MPC, demonstrating that C-MPC successfully finds a feasible trajectory. Simulation results for the PC-MPC architecture are presented in Figure 4.5. Most notably, Figure 4.6 compares the cumulative costs of both controllers, showing that they achieve comparable performance under the same cost function. Although both controllers yield similar optimal performance, their computational costs differ significantly. TMPC with an extended horizon requires an average of 7.2 ms per solve, while C-MPC takes 3.8 ms and P-MPC just 0.5 ms on average.

4.5. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

This paper introduced a novel architecture, called the PC-MPC. The P-MPC layer generates long-term plans using a simplified model and/or smaller sampling times for computational efficiency and to ensure stability, while the C-MPC layer refines the plan in real time to enhance performance. Formal guarantees for stability and recursive feasibility of the PC-MPC architecture are provided.

The primary advantage of this architecture lies in its ability to reduce computational cost while planning further ahead. Although C-MPC solves nonlinear, non-convex problems, the warm start from MPC improves the solver efficiency.

The benefits of the PC-MPC architecture are demonstrated through two case studies. The first shows that this architecture reduces the computation time and expands the controllable set in linear systems. The second illustrates that — thanks to short-term corrections of C-MPC — the architecture remains effective for systems with Lipschitz nonlinearity, even though P-MPC uses an inaccurate model.

Note that the concept of PC-MPC is easily extendable and may be utilized with other controllers, even in the absence of satisfaction of the assumptions. In Chapter 5, a new, redesigned PC-MPC is introduced to reduce computations for the multi-robot exploration problem. This problem is characterized by two properties: first, the possible actions of the robots belong to a finite set; and second, the system's nonlinearity cannot be represented by a linear model, even if significant simplifications are made. Furthermore, Chapter 5 demonstrates the potential for PC-MPC application in the coordination of multi-robot systems. This finding indicates that the PC-MPC model possesses the capacity to be implemented in a variety of contexts and settings.

Future research should focus on applying this architecture to more complex controllers, such as State-Dependent Dynamic Tube-based MPC (SDD-TMPC) presented in Chapter 2, and to real-world systems, e.g., quadrotors [7]. Evaluating and comparing tube generation algorithms to enhance feasibility and performance in more demanding applications are also topics for future research.

REFERENCES

[1] F. Surma and A. Jamshidnejad. “Recursive feasibility without terminal constraints via parent–child MPC architecture”. In: *Results in Control and Optimization* 22 (2026). DOI: https://doi.org/10.1016/j.rico.2025.100653.

[2] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, 2017. ISBN: 9780975937754.

[3] P. Falugi. “Model predictive control: a passive scheme”. In: *IFAC Proceedings Volumes* 47 (2014). DOI: 10.3182/20140824-6-ZA-1003.02165.

[4] W.-H. Chen and Y. Yan. “New stability theory of model predictive control: modified stage cost approach”. In: *International Journal of Systems Science* 56 (2025). DOI: 10.1080/00207721.2024.2409846.

[5] E. Kerrigan and J. Maciejowski. “Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control”. In: *Proceedings of the 39th IEEE Conference on Decision and Control*. Vol. 5. 2000. DOI: 10.1109/CDC.2001.914717.

[6] S. Liu and J. Liu. “Economic model predictive control with extended horizon”. In: *Automatica* 73 (2016). DOI: 10.1016/j.automatica.2016.06.027.

[7] M. Chipofya, D. Lee, and K. Chong. “Trajectory Tracking and Stabilization of a Quadrotor Using Model Predictive Control of Laguerre Functions”. In: *Abstract and Applied Analysis* 2015 (2015). DOI: 10.1155/2015/916864.

[8] W. F. Lages and J. A. Vasconcelos Alves. “REAL-TIME CONTROL OF A MOBILE ROBOT USING LINEARIZED MODEL PREDICTIVE CONTROL”. In: *IFAC Proceedings Volumes* 39 (2006). DOI: 10.3182/20060912-3-DE-2911.00166.

[9] D. Zhou and K. V. Ling. “The effect of sample/hold time on initial feasible set in model predictive control design”. In: *2013 Australian Control Conference*. IEEE, 2013, pp. 212–217. DOI: 10.1109/AUCC.2013.6697275.

[10] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. “Robust MPC for tracking constrained unicycle robots with additive disturbances”. In: *Automatica* 90 (2018). DOI: 10.1016/j.automatica.2017.12.048.

[11] D. He. “Dual-mode nonlinear MPC via terminal control laws with free-parameters”. In: *IEEE/CAA Journal of Automatica Sinica* 4 (2017). DOI: 10.1109/JAS.2016.7510013.

[12] M. Brdys, M. Grochowski, T. Gminski, K. Konarczak, and M. Drewa. “Hierarchical predictive control of integrated wastewater treatment systems”. In: *Control Engineering Practice* 16 (2008). DOI: 10.1016/j.conengprac.2007.01.008.

[13] G. S. van de Weg, H. L. Vu, A. Hegyi, and S. P. Hoogendoorn. "A Hierarchical Control Framework for Coordination of Intersection Signal Timings in All Traffic Regimes". In: *IEEE Transactions on Intelligent Transportation Systems* 20 (2019). DOI: 10.1109/TITS.2018.2837162.

[14] A. Jamshidnejad, D. Sun, A. Ferrara, and B. De Schutter. "A novel bi-level temporally-distributed MPC approach: An application to green urban mobility". In: *Transportation Research Part C: Emerging Technologies* 156 (2023). DOI: 10.1016/j.trc.2023.104334.

[15] F. Tahir. "Efficient computation of Robust Positively Invariant sets with linear state-feedback gain as a variable of optimization". In: *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*. ieee, 2010, pp. 199–204. DOI: 10.1109/ICEEE.2010.5608613.

[16] S. Raković and D. Mayne. "A SIMPLE TUBE CONTROLLER FOR EFFICIENT ROBUST MODEL PREDICTIVE CONTROL OF CONSTRAINED LINEAR DISCRETE TIME SYSTEMS SUBJECT TO BOUNDED DISTURBANCES". In: *IFAC Proceedings Volumes* 38 (2005). DOI: 10.3182/20050703-6-CZ-1902.00440.

[17] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi. "Tube-based robust nonlinear model predictive control". In: *International Journal of Robust and Nonlinear Control* 21 (2011). DOI: 10.1002/rnc.1758.

[18] H. E. T. Yiqi Gao Andrew Gray and F. Borrelli. "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles". In: *Vehicle System Dynamics* 52 (2014). DOI: 10.1080/00423114.2014.902537.

[19] S. Yu, H. Chen, and F. Allgöwer. "Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2650–2655. DOI: 10.1109/CDC.2011.6160389.

[20] S. Rakovic, E. Kerrigan, K. Kouramas, and D. Mayne. "Invariant approximations of the minimal robust positively Invariant set". In: *IEEE Transactions on Automatic Control* 50 (2005). DOI: 10.1109/TAC.2005.843854.

[21] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone. "Robust online motion planning via contraction theory and convex optimization". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5883–5890. DOI: 10.1109/ICRA.2017.7989693.

[22] F. Surma. *Code to test the implementation of linear Bi-level Parent-Child MPC*. 2025. DOI: 10.4121/c648d7b4-05ff-4f0e-bc11-230b4f9dfecc.v1.

[23] F. Surma. *Code to test the implementation of nonlinear Bi-level Parent-Child MPC*. 2025. DOI: 10.4121/c9ef0e3e-0ace-4cb8-bd58-451c7ff2f4bd.v1.

[24] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. "Multi-Parametric Toolbox 3.0". In: *Proc. of the European Control Conference*. IEEE, 2013, pp. 502–510. DOI: 10.23919/ECC.2013.6669862.

[25] *quadprog*. URL: https://www.mathworks.com/help/optim/ug/quadprog.html (visited on 03/15/2025).

[26] A. Forsgren, P. E. Gill, and E. Wong. “Primal and dual active-set methods for convex quadratic programming”. In: *Mathematical Programming* 159 (2016). DOI: 10.1007/s10107-015-0966-2.

[27] *fmincon*. URL: https://www.mathworks.com/help/optim/ug/fmincon.html (visited on 03/22/2025).

5

FUZZY-LOGIC-BASED MODEL PREDICTIVE CONTROL:

A PARADIGM INTEGRATING OPTIMAL AND COMMON-SENSE DECISION-MAKING[1]

This Chapter introduces a novel concept, Fuzzy-Logic-based MPC (FLMPC), along with a multi-robot control approach for exploring unknown environments and locating targets. Traditional Model Predictive Control (MPC) methods rely on Bayesian theory to represent environmental knowledge and optimize a stochastic cost function, often leading to high computational costs and a lack of effectiveness in locating all the targets. The presented approach instead leverages FLMPC and extends it to a bi-level Parent-Child MPC (PC-MPC) architecture for enhanced coordination and extended decision-making horizon. Extracting high-level information from probability distributions and local observations, FLMPC simplifies the optimization problem and significantly extends its operational horizon compared to other MPC methods. Extensive simulations are conducted in unknown 2-dimensional environments with randomly placed obstacles and humans. The performance and computation time of FLMPC are compared against MPC with a stochastic cost function. The impact of integrating the high-level parent FLMPC layer is evaluated. The results indicate that the new approaches significantly improve both performance and computation time, enhancing the coordination of robots and reducing the impact of uncertainty in large-scale Search-and-Rescue (SaR) environments

5.1. INTRODUCTION

Thus far, this thesis has focused on extending Robust MPC (RMPC). The primary benefit of RMPC is its ability to ensure safety as long as the initial assumptions remain valid, a crucial

[1]This chapter is based on the submitted paper [1] available at https://www.sciencedirect.com/science/article/abs/pii/S1568494626002656

consideration for numerous problems. However, for a lot of problems, it is not possible to implement RMPC [2]. This phenomenon may be attributed to the presence of excessively extensive or even infinite uncertainty bounds. Alternatively, the system's complexity may prevent the identification of a solution that is universally safe. To illustrate this point, consider a mission planning for the Search-and-Rescue (SaR) problem, where the uncertainty cannot be described with a robust approach.

Since 2021, there have been over 200,000 deaths due to various natural and technological disasters [3]. This number would even be larger without the proper intervention of the rescue teams, showing the importance of systematic and rapid SaR in the aftermath of disasters. In SaR, a team of trained rescuers is dispatched to investigate unfamiliar environments and to locate and assist the trapped victims. To achieve an optimal and expedient completion of the SaR mission, the mission must be operated time-efficiently, as prolonged search times are associated with a diminished probability of survival for the trapped victims [4, 5]. Even in situations that do not initially appear life-threatening, the expenditure of valuable time on repetitive searches within a confined space or the failure to conduct a comprehensive search of the surrounding area can prove detrimental [4].

In the modern era, SaR missions that used to be conducted solely by human rescuers are increasingly delegated to teams composed of autonomous robots and humans, where the robots should take over tasks that are life-threatening for human rescuers, including the exploration of unknown environments [6]. Robots have been deployed in large-scale disasters, e.g., in the SaR operations for the tsunami that struck the Fukushima nuclear power plant in 2011 [7]. Various types of robots are currently used in search and rescue operations, including aerial robots [8] and (under-)ground robots [9], e.g., snake robots [10] and legged robots [11].

The long-term objective of SaR is to use fully autonomous teams of robots in high-risk missions, without any need for human intervention. This is motivated by the inherent dangers associated with navigating damaged buildings. In such cases, firefighters may have a limited window of opportunity to evacuate. They should locate the exit expeditiously, while retreating through the same route they entered the building may not be possible due to ongoing structural damages [4]. To make multi-robot teams effective in SaR missions, their decisions/actions should be coordinated [12]. Crucially, these robots should be capable of making both short-term and long-term plans that ensure fast and safe exploration and evacuation of the building, despite uncertainties [4].

5.1.1. PROPOSED SOLUTION

In this Chapter, a novel version of Model Predictive Control (MPC), called Fuzzy-Logic-based MPC (FLMPC) is introduced. It is then extended with Parent-Child MPC (PC-MPC) that particularly suits spatially and/or temporally large-scale control problems that involve multiple levels of uncertainty, e.g., exploration of unknown environments through SaR

robotics. The effectiveness of FLMPC is shown, both in its original formulation and when reformulated within the bi-level architecture, for multi-robot exploration mission planning via computer-based simulations.

The main motivation for introducing FLMPC is the following: As is later presented in Section 5.2, a variety of approaches that base their decision-making in uncertain situations on the use of probability values face a common issue, i.e., they require frequent updates to the probability distributions inside the control loop. This not only diminishes the efficiency of the solver, but also demands assumptions regarding all future states and measurements of the system [13, 14]. In complex decision-making tasks, e.g., in searching for targets in unknown areas, human experts prove very adept, while they do not focus on calculating any precise changes in the probabilities over time. Instead, humans rely on their "common sense" and adhere to fundamental principles such as "avoid danger" or "explore less-known places first". To emulate such an approach via a mathematical control system, Fuzzy Logic (FL), a mathematical approach that handles imprecision by mimicking human logic, is integrated into the MPC formulation. FLMPC allows for making decisions based on multiple objectives and constraints, such that these decisions are effective in the long-term and challenging, uncertain environments.

In [15], it is proposed that the use of fuzzy sets be employed to identify the optimal decision in a "fuzzy environment," which can be defined as an environment characterized by multiple conflicting rewards and soft constraints. In the context of robotic research, the exploration of unknown environments, the destruction of obstacles, and the rescue of humans are considered to be valuable objectives. Given that the objectives in question are capable of directing a robot in a variety of divergent directions, they are inherently conflicting. Similarly, robots mustn't risk their safety to a great extent. As the priority of rescuing a human is of the utmost importance, a safe path to a human can be nonexistent. The implementation of soft constraints is essential to ensure that robots prioritize human safety over their own.

In actual SaR operations, individuals are coordinated by an incident commander, who analyzes the available information and directs the efforts of numerous responders simultaneously [6]. This allows multiple rescuers to be coordinated and enables them to locally make reactive and rapid decisions based on newly acquired information during the search process. Inspired by this, bi-level PC-MPC architecture, introduced in Chapter 4 is combined with FLMPC, where in the higher level an FLMPC system tunes the parameters of one or multiple low-level FLMPC systems to steer them in relevant sub-sets of the admissible solutions, while these low-level FLMPC systems react properly to unexpected/unpredicted measurements, akin to human rescuers.

5.1.2. MAIN CONTRIBUTION AND STRUCTURE OF THE CHAPTER

The main contributions of this Chapter are:

- **Developing FLMPC for effective handling of uncertainties:** A novel FLMPC framework is introduced for exploring unknown environments and for locating targets using dynamic fuzzy maps. FLMPC addresses the computational and performance limitations of traditional MPC approaches that rely on stochastic cost functions and Bayesian theory for interpreting unknown environments.
- **Redesigning bi-level PC-MPC architecture for multi-robot control:** A bi-level PC-MPC architecture is proposed to enhance coordination and to extend the decision-making horizon of FLMPC. This improves the ability of robots to handle large-scale environments and multiple levels of uncertainty.
- **Evaluation and comparison:** Extensive simulations are performed, demonstrating the superiority of FLMPC over conventional MPC with a stochastic cost function. They reveal significant improvements in both performance and computation time, as well as enhanced coordination and robustness in large-scale SaR missions.

The Chapter is structured as follows. Section 5.2 reviews alternative approaches from related literature, Section 5.3 gives the problem statement, the theory of FLMPC, and its adoption and leverage via a bi-level PC-MPC architecture for SaR robotics. Section 5.4 presents the results of 2 case studies, comparing FLMPC with MPC using a stochastic cost function, and evaluating single-level versus bi-level parent-child FLMPC. Finally, Section 5.5 concludes the paper and outlines future research directions.

5.2. RELATED WORK

The concept of FLMPC introduced in this Chapter and its implementation for SaR are based on fuzzy optimization, MPC, and grid-based exploration of unknown environments. Thus, below a background discussion on these topics is provided.

5.2.1. FUZZY DECISION-MAKING

Fuzzy Logic (FL) is first introduced in [16] for describing deterministic uncertainties using fuzzy sets. The fuzziness arises from imprecise boundaries in quantifying a concept, rather than from inherent randomness or unpredictability. Fuzzy sets may, in general, be seen as mathematical representations of vague linguistic terms used by humans that are not (precisely) quantified. For example, terms such as “high”, “hot”, and “tall” possess no clear delineation with, respectively, “low”, “cold”, and “short”. Consequently, in fuzzy set theory, a membership function is associated with each fuzzy set that provides the extent to which an element or a value (e.g., temperature 26°C) belongs to that fuzzy set (e.g., the set “hot”). While membership functions are frequently defined on single scalar inputs, they can also accept vector inputs, resulting in multidimensional membership functions. The utilization of these in the context of describing complex concepts, which are frequently associated with SaR, is of particular utility.

As explained in Chapter 2, the Fuzzy Inference System (FIS) constitutes a pivotal component of FL, wherein a series of fuzzy rules are implemented to govern these membership values. Rules are typically expressed as "IF conditions THEN output," where conditions are defined by fuzzy sets that are combined using logical operators. The FIS employs an aggregation of the membership degrees of the inputs to evaluate the degree to which the rules apply, subsequently combining the rule outputs to generate fuzzy conclusions. The conversion of these imprecise results into precise output values is achieved through a process known as defuzzification, which facilitates the implementation of effective decision-making or control mechanisms in real-world settings.

In [15], FL has been leveraged for decision-making, particularly to control a given system or agent within a fuzzy environment with possibly multiple fuzzy goals (i.e., goals that, instead of a clear or binary distinction between being achieved and not being achieved, allow for a range of satisfactory outcomes associated with different degrees of fulfillment) and fuzzy (also called soft) constraints (i.e., constraints that allow flexibility or degrees of satisfaction, rather than being strictly binary). Each goal and constraint is represented by a fuzzy set, which receives a state or control input of the controlled system/agent as input. The objective of the fuzzy decision-making system is to provide a multi-stage process in the fuzzy environment that results in a sequence of control inputs, given the current state of the controlled system/agent. This sequence should result in an optimal outcome for the controlled system/agent with regard to the (fuzzy) goals. To estimate the final outcome, an aggregation operator may be used that exhibits both monotonicity and commutativity properties. Monotonicity implies that when one input of the aggregation operator is increased, then the output either increases or remains unchanged. Commutativity implies that the order of the inputs to the aggregation operator does not affect the outcome [17].

Later on, the concept of decision-making in fuzzy environments served as the foundation for fuzzy optimization [18], which emulates the decision-making processes of humans when confronted with situations involving competing objectives. In [19], fuzzy MPC is introduced, where conventional MPC is solved using fuzzy optimization. Fuzzy MPC is shown to result in a steady-state for the controlled system more rapidly than standard MPC, although the primary disadvantage of fuzzy MPC is its high computation time. The most notable distinction of fuzzy MPC in comparison with previous methods based on fuzzy optimization is that fuzzy MPC performs aggregation operations separately on the fuzzy goals and the fuzzy constraints, whereas previous fuzzy optimization methods treat the fuzzy goals and constraints similarly and aggregate them altogether. Thus, fuzzy MPC properly addresses multi-stage control problems that involve delayed goals (particularly rewards). In such cases, adopting state realizations that do not result in immediate rewards may still be beneficial when it allows one to later reach states that will result in very high cumulative rewards. Since fuzzy constraints should prohibit the realization of the higher-risk states, such an impact on the overall reward will be obstructed when the goals and constraints are merged and treated the same. The

stability of fuzzy MPC is proved by imposing hard terminal constraints and by assuming the existence of a locally stabilizing terminal control law that replaces the fuzzy MPC strategy as soon as the system reaches the hard terminal set [20]. It has later been shown that replacing the fuzzy MPC strategy with a terminal control law is deemed unnecessary by including additional terminal fuzzy goals for the states [21].

In Chapter 2, FL and MPC have been integrated to incorporate the dynamics of the external disturbances into the decision-making of MPC through fuzzy models. Moreover, FL and MPC are also used to represent the nonlinear dynamics of the controlled system via fuzzy models embedded in MPC; to include fuzzy decision-making and MPC in modular or hierarchical control architectures that collaboratively steer the controlled systems with complex dynamics towards desired objectives [12, 22]

5.2.2. MPC IN EXPLORATION OF UNKNOWN ENVIRONMENTS

Path planning for multiple robots that should explore unknown environments is a highly relevant problem for various applications, including SaR. Chapter 2 introduced State-Dependent Dynamic Tube-based MPC (SDD-TMPC), which ensures that the system will adhere to the given commands and reach the established objectives in both optimal and safe ways. In Chapter 3, the extension of SDD-TMPC into Approximate State-Dependent Dynamic Tube-based MPC (ASDD-TMPC) is undertaken to address the primary drawback of the time needed to solve its optimization problem. While ASDD-TMPC may possess the capacity to execute commands, but it still requires a controller at a higher level to formulate a strategy.

It is common to formulate the exploration problem of the robots in the discrete space domain by representing the environment via a grid, where each cell in the grid may be empty or occupied by various elements. After initializing their positions, at every time step, the robots make either an omnidirectional (in all possible directions, e.g., forward, backward, and to the sides) or a restricted (e.g., only in specific directions, e.g., forward, forward-right, forward-left) movement that takes into account their limited maneuverability [13, 14, 23–25]. After each movement, the robots receive new measurements from the environment and update their map of the environment, commonly using the Bayesian rule [23, 26].

The mission planning for the robots is often represented as a multi-stage, nonlinear, non-convex optimization problem that generally includes both the constraints and the objective function of SaR in the decision-making [13, 14, 23, 24, 27, 28]. However, solving such optimization problems faces two main challenges: First, since the robots continuously receive new information about their environment during the mission, the optimization problem should be resolved very frequently [28]. Second, due to the non-convex nature of the problem and the limited time to solve it in real-world SaR missions, the returned solutions, if received in time, are likely to be sub-optimal, thus degrading the overall performance. Therefore, the formulation of the optimization problem is of utmost importance, and the decision variables

should efficiently encode the most relevant information.

Each MPC input, in general, is a single decision variable that is determined for all time steps along the prediction horizon. In exploring unknown environments that are represented by a grid through robots, the decision variable is commonly the movement of the robots per discrete time step, i.e., the direction or position of the next cell the robot should move to. This approach in literature is often referred to as motion encoding [13, 14, 23, 27], which faces multiple challenges: Determining long paths for large-scale environments via motion encoding results in large numbers of decision variables, thus in excessive computational efforts. Moreover, since the selection of each candidate path depends on the position of all its cells, changing the value of one decision variable in motion encoding requires changing the entire path [14]. This implies that the value of the objective function is very sensitive to single changes in the decision variables. By assigning different decision variables to each motion, a solver can return complex paths. However, despite the apparent advantage, this is not a significant factor. In various cases, the optimal paths are simply the shortest paths that connect the current position of the robot to its target via straight lines. In such cases, motion encoding results in unnecessarily large computational costs. In [14], priority encoding has been proposed, where a number is initially decided, as a hyperparameter, for the goals, which are the decision variables. The path is determined by connecting these goals through straight lines, leading to fewer decision variables and, thus, reducing the sensitivity of the value of the objective function to any changes in the decision variables.

In selecting optimal paths for robots, multiple candidate paths are usually graded and compared according to an objective function. Maximizing the probability of detecting the targets across the prediction horizon is a common objective in literature [13, 23], but it does not guarantee the minimization of the detection time, which is also crucial in SaR [29]. Alternatively, a discounted detection utility function has been proposed to formulate the objective function in [30]. Next to maximizing the cumulative probability of target detection, a discount factor encourages the robots to prioritize visiting locations that have a higher probability of detecting the targets, aiming at speeding up the target detection procedure.

To incorporate the SaR uncertainties into the decision-making, the path planning problem is usually formulated within a stochastic framework. However, solving the optimization problem within a stochastic framework using MPC results in complex representations for the objective function, for instance, the cumulative expected value of an objective within the prediction horizon. Predicting the probability distribution for all the cells in real life, however, results in expensive computations. Alternatively, simplifying assumptions may be made about the future measurements in the cells: For instance, in [14], the cells that are detected within the prediction window are assumed to be vacant.

In general, probability-based objective functions are suited mainly when knowledge about the initial probability distribution of the target locations in the environment exists. Otherwise, additional non-probabilistic terms should be included in the objective function, e.g., the

(un)certainty level about the environmental knowledge, to encourage the robots to visit unexplored parts of the environment [12, 14]. A detection-based reward may also be included in the objective function to steer the robots to visit cells that include the targets [14]. A pheromone map has been used in [24], to enhance the coordination of the robots by discouraging them from moving closely to cells that are currently visited. This prevents the robots from limiting their exploration to the same neighborhood. A motion cost may also be incorporated within the objective function, to discourage the robots from unnecessary movements [14].

A crucial step in solving the path planning optimization problem is to choose a proper optimization solver. The literature suggests, among others, the Genetic Algorithm (GA) [14, 23], Particle Swarm (PS) algorithm [13], Sand Cat Swarm algorithm [27], Bayesian Optimization algorithm [30], and Ant Colony Optimization [24, 29]. Additionally, it should be decided whether or not the optimization problem is formulated in a decentralized/distributed [24, 28], a centralized [14], or a combined framework. For example, in [23], the targets (whose positions are modeled according to a given distribution) are divided into groups, and each robot is assigned to one group of these targets and performs independently from other robots (since robots do not contribute to each other's decision-making, this is a decentralized architecture). In [12], the robots use both a decentralized framework, based on FIS, and a centralized controller, based on MPC.

5.3. PROPOSED METHODS

In this Section, the details of the proposed methods are explained for autonomous multi-robot path planning in unknown and uncertain environments. This includes the problem statement, the new concept of FL leveraging Fuzzy-Logic-based MPC (FLMPC) for exploration of unknown environments, and extending it with redesigned PC-MPC, in which bi-level FLMPC is formulated for efficient solution of complex large-scale path planning problems.

5.3.1. PROBLEM STATEMENT

n^{rob} number of autonomous robots (e.g., flying/ground search robots) are considered that should map an initially unknown environment represented by a two-dimensional grid of n^{h} horizontal and n^{v} vertical cells, where all cells in the grid have the same size and shape (namely that of a square). The dynamics of the control system that is used to steer the robots in the environment is formulated in the discrete time. Assuming a static environment, each cell c in the environment is represented via its horizontal and vertical coordinates $[\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}]^\top$, where $\xi_c^{\mathrm{h}} \in \{1, \dots n^{\mathrm{h}}\}$ and $\xi_c^{\mathrm{v}} \in \{1, \dots, n^{\mathrm{v}}\}$, and is associated with a state s_c^{r} that contains all the relevant information that describes the status of the cell. In particular, the state of a cell in a SaR mission may include information about its occupancy status (e.g., empty, embedding humans, occupied by obstacles, etc.). The admissible set of the different realizations for s_c^{r}

is shown by $\mathcal{S}$. In general, this set is given as an ordered set $\mathcal{S} = \{\rho_1, \ldots, \rho_{|\mathcal{S}|}\}$, where $|\cdot|$ is used for the cardinality of the set and ρ_i's for $i = 1, \ldots, |\mathcal{S}|$ are all possible realizations of s_c^{r} for all cells c in the environment. At every time step, robots can move to an adjacent cell (including diagonal movements). Note that the states associated with the cells may correspond to different real-life concepts. For the robots to distinguish these concepts, the set $\mathcal{S}$ is defined to be ordered.

While the environment is assumed to remain unchanged in time, the knowledge about the environment evolves in time as the robots explore the environment. Thus, per time step k, the path planning system has access to an estimate $s_{c,k}$ of the real state s_c^{r} of each cell in the environment, where this estimation is updated in time. No data or information is initially available regarding the content of the cells or the number of humans in the environment. As the robots traverse the environment, they scan it via their sensors, which in general may be prone to imperfections. The objective of the multi-robot system is to detect/rescue as many victims as possible in the shortest possible time, while reducing the number of robots that become out of function (e.g., that are crashed into or that are caught in fire) during the SaR.

Table 5.1 shows the mathematical notations that are frequently used in this Chapter, together with their definitions. A regular font is used for scalars and a bold font is used for vectors.

5.3.2. FLMPC

This Section introduces FLMPC, where the discussions are based on the following assumptions:

A1 FLMPC determines optimal trajectories for a system, which then follows these trajectories to dynamically interact with an initially unknown environment.

A2 There is a finite number of admissible states for the environment, and these are known to the system.

A3 The states of the cells are generally graded according to multiple, potentially competing objectives.

A4 The system is generally exposed to constraints, which depend on the environmental states, and these should be satisfied when interacting with the environment.

FLMPC is introduced for a system that should dynamically interact with an environment prone to Assumption **A2**. Due to the environmental uncertainties, the objectives of the system (prone to Assumption **A3**) and its constraints (prone to Assumption **A4**) are described as fuzzy goals and fuzzy constraints, which may conflict with one another. Each mathematically given by fuzzy sets and, thus, associated with membership functions. Each membership function, thus, receives as input the current state $\boldsymbol{x}_k$ of the system and the current knowledge

Table 5.1: Frequently used mathematical notations and their definition

$b_{c,i,k}$	Probability of a cell c having ρ_i as a realization of a state at time-step k
c	2-element vector containing position of a cell
$\boldsymbol{e}_k$	Current knowledge about the environment
$f(\cdot)$	Model of system's dynamics
$\tilde{f}_l(\cdot)$	Model describing how to update fuzzy maps
$\mathcal{M}^{\mathrm{c}}_k$	Certainty map at time-step k
$\tilde{\mathcal{M}}^{\mathrm{env}}{}_k$	Set of all fuzzy maps at time-step k
$\mathcal{M}^{\mathrm{p}}_k$	Probability map at time-step k
n^{cluster}_k	Number of clusters
n^{con}	Number of constraints
n^{f}	Number of fuzzy states required to describe a cell
n^{goal}	Number of goals
n^{h}	Horizontal size of a grid
n^{p}	Prediction horizon (for MPC in general)
n^{path}	The longest possible path computed during optimization
n^{rob}	Number of robots
n^{travel}	The longest possible part of a path defined by a single decision variable
n^{v}	Vertical size of a grid
$o_c(k)$	Observation of cell c at time-step k
$\mathcal{O}(k)$	Set of all measurements at time-step k
$p(\cdot\|\cdot)$	Conditional probability
$\mathbb{P}$	An ordered set of cells to be visited by a robot computed by Parent Fuzzy-Logic-based MPC (P-FLMPC)
r	Reference
$\mathcal{R}$	Set of all robots
$\tilde{\boldsymbol{s}}_{c,k}$	A vector of fuzzy states describing a cell c at time-step k
$\tilde{s}_{c,l,k}$	A l^{th} fuzzy state of a cell c at time-step k
$s_{c,k}$	An estimate of a state of a cell c
$s^{\mathrm{cluster}}_{j,k}$	A score given to the j^{th} cluster at the time-step k
s^{exp}	If a score of a cluster is below this value, it is considered explored.
s^{r}_c	A real state of a cell c
s^{unexp}	If a score of a cluster is above this value, it is considered unexplored.
$\mathcal{S}$	The admissible set of all possible states of a cell in the environment.
$\boldsymbol{u}_k$	Input at time step k
w^{agg}	Parameter used in the final aggregation
w^{con}	Parameter used to aggregate constraints
w^{goal}	Parameter used to aggregate goals
$w^{\mathrm{goal,P}}$	Parameter used to aggregate goals but used by P-FLMPC
$w^{\mathrm{HL,C}}(c)$	A weight given to a cell by P-FLMPC
w^{cluster}	A weight used in computing scores for clusters
$w^{\mathrm{tune}}_{c,\kappa_{c,k}}$	A parameter that shows the weight given to a cell c at time-step k used by single level FLMPC
$w^{\mathrm{tune,C}}_{j,\kappa_{j,k}}$	A parameter that shows the weight given to a cell c at time-step k used by Child Fuzzy-Logic-based MPC (C-FLMPC)
$w^{\mathrm{tune,P}}_{j,\kappa_{j,k}}$	A parameter that shows the weight given to a cell c at time-step k used by P-FLMPC
$w^{\mathrm{tune}}_{c,\kappa_{c,k},r}$	A parameter that shows the weight given to a cell c at time-step k by the r^{th} robot
$\bar{w}^{\mathrm{tune}}_{c,\kappa_{c,k},r}$	A parameter that shows the weight given to a cell c at time-step k by the r^{th} robot while taking into consideration other robots' weights
$\boldsymbol{x}_k$	State of the system at time k
$\hat{\boldsymbol{x}}_k$	Trajectory of states starting at time k
$\hat{\boldsymbol{x}}^{\mathrm{neigh}}_k$	All states in the neighborhood of all states in trajectory $\hat{\boldsymbol{x}}_k$
$\mathcal{X}_k$	Set of all observable states along a trajectory $\hat{\boldsymbol{x}}_k$ and weights allocated to them
$\mathcal{X}^r_k$	Set of all robots' $\mathcal{X}_k$
$z_c(k)$	Certainty of cell c at time-step k
$\alpha(\cdot)$	A parameter that shows how distance affects goals
$\beta(\cdot)$	A parameter that shows how time affects goals
γ	A constant discount factor between 0 and 1 used to show how time affects cost
$\delta_{c,\kappa_{c,k}}$	Predicted distance between a robot and a cell c at time step κ
δ^{max}	Maximum distance sensed around a robot
κ	This parameter represents the predicted time-step
$\lambda^{\mathrm{h}}_{i,k}$	Horizontal position of a center of the i^{th} cluster at time step k
$\lambda^{\mathrm{v}}_{i,k}$	Vertical position of a center of the i^{th} cluster at time step k
$\mu^{\mathrm{agg}}(\cdot)$	Function that returns how a trajectory of states satisfies all goals and constraints
$\mu^{\mathrm{cluster}}_{i,k}(c)$	Membership of a cell belonging to the i^{th} cluster at time step k
$\mu^{\mathrm{con}}_i(\cdot)$	Function that returns how a state satisfy i^{th} constraint
$\mu^{\mathrm{con}}(\cdot)$	Function that returns how a trajectory of states satisfies all constraints
$\mu^{\mathrm{goal}}_i(\cdot)$	Function that returns how a state satisfy i^{th} goal
$\mu^{\mathrm{goal}}(\cdot)$	Function that returns how a trajectory of states satisfies all goals
ξ^{h}_c	Horizontal coordinate of a cell
ξ^{v}_c	Vertical coordinate of a cell
ρ_i	i^{th} possible state
σ	A constant parameter used to compute $w^{\mathrm{HL,C}}(c)$

$\boldsymbol{e}_k$ about the states of the environment (a vector that includes $s_{c,k}$ at time step k for all cells of the environment), and returns a value in $[0,1]$ that reflects the importance of the state with regards to that fuzzy goal. Note that the degree of importance may be different (sometimes contradicting) for the same state in regards to the different objectives. This, for the fuzzy constraints, implies that the realization of those states whose membership degree to the corresponding fuzzy sets is less than 1 penalizes the system.

FLMPC determines a candidate state trajectory $\hat{\boldsymbol{x}}_k = \{\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_{k+n^{\mathrm{p}}}\}$, based on the candidate input trajectory $\hat{\boldsymbol{u}}_k = \{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+n^{\mathrm{p}}-1}\}$, for the system per time step k, with the hyper-parameter n^{p} representing the prediction horizon. This state trajectory should maximize the consensus for all different objectives, violating or being penalized by constraints. Each candidate trajectory involves a total number of $n^{\mathrm{p}}(n^{\mathrm{goal}} + n^{\mathrm{con}})$ importance/satisfaction degrees, with n^{goal} and n^{con} the total number of fuzzy goals and fuzzy constraints, respectively. Therefore, for a solver to identify an optimal solution, a single aggregated degree of importance should be obtained to be used to evaluate the entire trajectory. In fuzzy optimization, where the goals and the constraints are represented via fuzzy sets, aggregation operations are employed at the end, after estimating individual degrees of importance, to derive the overall degree of importance for a given solution. This allows for taking into account the satisfaction levels associated with both the goals and the constraints. For FLMPC, three primary operations are performed: (1) aggregation of the importance levels of a candidate solution with regard to all the fuzzy goals; (2) aggregation of the satisfaction levels for all the fuzzy constraints; (3) aggregation of the results from the previous two stages to determine the best solution at the final stage.

AGGREGATING THE IMPORTANCE DEGREES FOR ALL FUZZY GOALS

Suppose that the candidate solution of FLMPC at time step k is trajectory $\hat{\boldsymbol{x}}_k$. Note that there is no assumption that the prediction model provides information about the future perception of the environment, and thus, for the sake of simplicity, the knowledge $\boldsymbol{e}_k$ available at time step k is used within the entire prediction horizon $\{k+1, \ldots, k+n^{\mathrm{p}}\}$. To aggregate the degrees of importance with regard to all the goals and to determine an overall degree of importance $\mu^{\mathrm{goal}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k)$ for this trajectory, the generalized mean is used, given below:

$$\mu^{\mathrm{goal}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k) = \left(\frac{1}{n^{\mathrm{p}}} \sum_{j=1}^{n^{\mathrm{p}}} \left(\mu_1^{\mathrm{goal}}(\boldsymbol{x}_{k+j}, \boldsymbol{e}_k) \star \ldots \star \mu_{n^{\mathrm{goal}}}^{\mathrm{goal}}(\boldsymbol{x}_{k+j}, \boldsymbol{e}_k) \right)^{w^{\mathrm{goal}}} \right)^{\frac{1}{w^{\mathrm{goal}}}} \tag{5.1}$$

with $\mu_i^{\mathrm{goal}}(\boldsymbol{x}_{k+j}, \boldsymbol{e}_k)$ for $i \in \{1, \ldots, n^{\mathrm{goal}}\}$ the degree of importance associated to state $\boldsymbol{x}_{k+j}$ with regards to the i^{th} goal, based on the current knowledge $\boldsymbol{e}_k$ about the states of the environment, and $\star$ the symbol used to represent one of possible S-norms [16]. Depending on the application, obtaining a larger number of high-value rewards may excel over achieving a constant stream of low-value rewards, even though both cases may result in the same mean

value. For instance, detecting a human (associated with a single high-valued reward) may generally be more important than exploring a larger part of the environment (associated with a stream of low-valued rewards). Tuning the weight w^{goal} in (5.1) allows us to distinguish the relative importance of such candidate trajectories. A larger value for w^{goal} may be considered to give more power to larger importance degrees (i.e., are closer to 1), resulting in a generalized mean that is closer to the maximum importance degrees.

AGGREGATING THE SATISFACTION DEGREES FOR ALL FUZZY CONSTRAINTS

In safety-critical applications, e.g., in SaR, those constraints whose violation is at higher risks, i.e., corresponds to the least membership degrees for the given solution trajectory, play a crucial role in the decision-making. Nevertheless, using the minimum function to aggregate the satisfaction levels of all the constraints poses two problems: First, the minimum function treats two trajectories equally if their worst constraint violations have the same membership degree, even though the trajectory with more violations should be less preferred. Second, it is usually very challenging for an optimization solver to identify an optimal state trajectory when a single state (the one associated with the highest risk) within the entire state trajectory evaluates the constraint satisfaction. Therefore, it is proposed to use the parameterized Yager T-norm function given below to estimate the overall degree $\mu^{\text{con}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k)$ of satisfaction of the fuzzy constraints for the candidate state trajectory $\hat{\boldsymbol{x}}_k$. There is:

$$\mu^{\text{con}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k) = \max\left\{0, 1 - \left(\sum_{j=1}^{n^{\text{p}}}\sum_{i=1}^{n^{\text{con}}}\left(1 - \left(\mu_i^{\text{con}}\left(\boldsymbol{x}_{k+j}, \boldsymbol{e}_k\right)\right)^{w^{\text{con}}}\right)\right)^{\frac{1}{w^{\text{con}}}}\right\} \tag{5.2}$$

with $\mu_i^{\text{con}}\left(\boldsymbol{x}_{k+j}, \boldsymbol{e}_k\right)$ the degree of satisfaction of the i^{th} constraint by state $\boldsymbol{x}_{k+j}$ within the candidate trajectory $\hat{\boldsymbol{x}}_k$, given the current knowledge $\boldsymbol{e}_k$ about the states of the environment. Unlike the minimum operator, this aggregation operator considers the satisfaction levels corresponding to states within a candidate trajectory. Moreover, for $w^{\text{con}} > 1$ it returns an aggregated value that is smaller than the aggregated value determined by the minimum operator for the same set of input values. This is because (5.2) introduces an additional penalty for all other constraints whose satisfaction degree is below 1. The weight w^{con} is calibrated to tune this penalty properly.

OVERALL AGGREGATION FOR DETERMINING THE SOLUTION OF FLMPC

Ultimately, in order to associate an overall membership degree $\mu^{\text{agg}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k)$ to the candidate state trajectory $\hat{\boldsymbol{x}}_k$, the following function is proposed for FLMPC:

$$\mu^{\text{agg}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k) = \mu^{\text{goal}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k) * \left(\mu^{\text{con}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k)\right)^{w^{\text{agg}}} \tag{5.3}$$

where the symbol $*$ represents one of possible standard T-norms in fuzzy logic [16]. The positive weight w^{agg} specifies the relative importance between the fuzzy goals and the fuzzy

constraints. For $w^{\text{agg}} < 1$, the goals are deemed to be of greater importance than the constraints, and vice versa.

OPTIMIZATION PROBLEM OF FLMPC

The following constrained optimization problem is solved by FLMPC, to determine an optimal state trajectory $\hat{\boldsymbol{x}}_k$ that corresponds to the maximum overall aggregated grade:

$$\max_{\hat{\boldsymbol{x}}_k} \mu^{\text{agg}}(\hat{\boldsymbol{x}}_k, \boldsymbol{e}_k) \tag{5.4a}$$

$$\text{such that for } \kappa \in \{k+1, \ldots, k+n^{\text{p}}\}:$$

$$\boldsymbol{x}_{\kappa+1} = f(\boldsymbol{x}_\kappa, \boldsymbol{u}_\kappa) \tag{5.4b}$$

$$\hat{\boldsymbol{x}}_k = \{\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_{k+n^{\text{p}}}\} \tag{5.4c}$$

where the constraint given by (5.4b) describes the dynamics of the system, i.e., the evolution of its states based on the given inputs.

In general, the constrained optimization problem (5.4) is nonlinear and non-convex.

Remark. The aggregation operations given in (5.1) and (5.2) are examples that have been chosen for their generality and useful properties (e.g., the ability to be fine-tuned by adjusting their hyperparameters for SaR). It is, of course, possible to implement FLMPC with different aggregation operators, depending on the application and the problem.

5.3.3. FLMPC FOR EXPLORING UNKNOWN ENVIRONMENTS

This section explains how FLMPC is adopted to solve the problem described in Section 5.3.1, where Assumptions **A2** and **A3** hold, i.e., the environment has a limited number of cells in a grid map with a finite number of states per cell, and the possible control inputs are constrained what means that there is a finite number of possible actions and states. Moreover, multiple objectives are possible. Additionally, the following assumptions are considered:

A5 Each time a robot visits a cell, it collects observations from that cell and all cells in a given neighborhood of the cell, i.e., a circular area of radius $\delta^{\max}$.

A6 The certainty about the observations for the cells that are in its neighborhood declines with the distance of the cell from the robot.

A7 There is a finite number of candidate trajectories per time step per robot that are assessed by FLMPC to determine an optimal one.

Next, it is discussed how to grade various candidate trajectories for robots, given maps that capture and are updated based on environmental information. This step is needed to solve the FLMPC optimization problem (5.4) for mission planning of SaR robots.

MAPPING UNKNOWN ENVIRONMENTS

While the states of the cells in an unknown environment are not (fully) known, at each time step, the most recent data should be used to update the perception of the robots about these states for planning or updating the mission of the robots. One prevalent method for organizing the knowledge of the environment is through the creation of maps that are regularly updated when new data or information is captured from the environment.

Probability and certainty maps, explained next, are the most common in the literature. Research shows that integration of probabilistic and fuzzy knowledge about uncertain environments significantly enhances the environmental awareness of autonomous robots [31]. Accordingly, to the above maps, the concept of fuzzy maps that is particularly useful for autonomous SaR via robots in unknown and uncertain environments is added. Note that in the mathematical notations, fuzzy concepts are distinguished from non-fuzzy concepts via a tilde symbol on top of the fuzzy notations.

Probability maps A probability map $\mathcal{M}_k^{\mathrm{p}}$ at time step k contains the probability distribution for all possible realizations of the states per cell for the given time step. The size of the probability map is, thus, $n^{\mathrm{h}} \times n^{\mathrm{v}} \times |\mathcal{S}|$, i.e., it may be considered as a matrix of size $n^{\mathrm{h}} \times n^{\mathrm{v}}$ (i.e., the total number of cells in the environment), for which each element is a vector of size $|\mathcal{S}|$: The vector in position $\left(\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}\right)$ of the probability map matrix includes the probability distribution for state variable $s_{c,k}$ of cell c in the environment, with the i^{th} element of this vector showing the probability, estimated at time step k, that the realized value of state $s_{c,k}$ is ρ_i (i.e., the i^{th} element of set $\mathcal{S}$, as defined in Section 5.3.1). This element of the vector is called the "belief" that the realization of the state of cell c at time step k is ρ_i and is denoted by $b_{c,i,k}$.

The beliefs about the states of the cells are, in general, subject to changes over time. Before the start of the mission, an initial probability distribution, including initial values $b_{c,i,0}$ for $i \in \{1,\ldots,|\mathcal{S}|\}$, is assigned to all cells c with $[\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}]^\top \in \left\{[1,1]^\top, \ldots, [n^{\mathrm{h}}, n^{\mathrm{v}}]^\top\right\}$. In the absence of initial environmental knowledge, the initial probability distributions may follow a uniform distribution for the cells. Alternatively, the values for $b_{c,i,0}$ may be initialized based on the knowledge about the environment before the disaster occurred.

In the course of the exploration, the robots collect data via their sensors, e.g., in the form of measurements or images. Each measurement, which is also referred to as an observation, is specified by $o_{c,k}$ for cell c at time step k. This data is then used to update the beliefs regarding the states of the cells in the environment. The following equation, inspired by the work in [26], is used to update the elements of the probability map for all cells c with $[\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}]^\top \in \left\{[1,1]^\top, \ldots, [n^{\mathrm{h}}, n^{\mathrm{v}}]^\top\right\}$:

$$b_{c,i,k+1} = \frac{p\left(o_{c,k}|\rho_i\right)}{\sum_{\ell=1}^{|\mathcal{S}|} p\left(o_{c,k}|\rho_\ell\right) b_{c,\ell,k}} b_{c,i,k} \qquad \text{for } i = 1,\ldots,|S| \tag{5.5}$$

where $p(o_{c,k}|\rho_i)$ is the likelihood that the observation in cell c at time step k is $o_{c,k}$, whereas the real state s_c^{r} of the cell is ρ_i. This likelihood may depend on the sensing capabilities or accuracy.

Certainty maps In exploration via robotic systems, probability maps are common, where current and/or future beliefs are incorporated into the objective function that should be optimized by the robots while exploring the environment. Nevertheless, alternative methods exist that employ other types of environmental maps. For instance, a prevalent map is the certainty map $\mathcal{M}_k^{\mathrm{c}}$, a matrix of size $n^{\mathrm{h}} \times n^{\mathrm{v}}$ with time-varying elements in $[0,1]$, represented by $z_{c,k}$ for cell c (i.e., for the element in position $\left(\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}\right)$ of the certainty map) at time step k. Each value indicates the degree of certainty regarding the observation acquired for the corresponding cell.

The certainty map is initialized with zeros, where these values increase compared to their values from the previous time step with the acquisition of new observations. The certainty degrees remain unchanged or decrease (for dynamic environments) in time when no new observations are acquired. As is indicated in Assumptions **A5** and **A6**, when visiting a cell, the robot acquires observations from the neighboring cells too. The farther the cell c from the robot, the smaller the contribution of this observation to the enhancement of $z_{c,k}$.

On the one hand, considering only one type of environmental map may hinder the performance of the robotic system. On the other hand, considering multiple maps may be computationally unaffordable for real-time large-scale SaR robotic applications, due to the high costs of updating the maps inside the optimization loop of the robots. Thus, it is crucial to find a reliable, computationally efficient approach to model the environment, capturing as much relevant information as possible. This is the main motivation for proposing fuzzy environmental maps next.

Fuzzy maps In this Chapter, it is proposed to model the perception of the robots about the environmental states via various low-level fuzzy maps, e.g., fuzzy certainty maps. These low-level fuzzy maps altogether build a matrix of size $n^{\mathrm{h}} \times n^{\mathrm{v}}$ that, instead of crisp variables, includes vectors of fuzzy variables as its elements. These fuzzy variables are each associated with one feature of the environment, e.g., the degree of certainty that a cell belongs to a specific category (represented as a label or fuzzy set) of certainty.

The main advantage of fuzzy environmental maps, as is showcased in this Chapter, is that these maps are perfectly compatible with FLMPC and, when embedded in FLMPC, can be updated outside the optimization loop. This is because their fuzzy, high-level nature captures a broader range of information than probability maps, allowing for less frequent updates. Thus, the resulting computational burden on online resources is significantly reduced. When an environment is modeled via n^{f} low-level fuzzy maps, each cell of the environment is in fact described by a fuzzy vector of dimension n^{f}. Unlike probability maps, with low-level fuzzy

maps, there is more flexibility in modeling various sources of uncertainty, following one's common sense [32].

New observations are first used to update the probability map. Then, the elements of the low-level fuzzy maps are updated based on the new observations, the updated probability map, and the most recent fuzzy maps. Each element of fuzzy vector $\tilde{\boldsymbol{s}}_{c,k} = [\tilde{s}_{c,1,k}, \ldots, \tilde{s}_{c,n^{\mathrm{f}},k}]^\top$, a vector that includes all the fuzzy characteristics/states associated to cell c and that is placed at position $\left(\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}\right)$ of the fuzzy map $\tilde{\mathcal{M}}_k^{\mathrm{env}}$ of the environment at time step k, is updated by:

$$\tilde{s}_{c,\ell,k+1} = \tilde{f}_\ell\left(\tilde{\mathcal{M}}_k^{\mathrm{env}}, \mathcal{M}_{k+1}^{\mathrm{p}}, \mathcal{O}_k\right) \qquad \ell = 1, \ldots, n^{\mathrm{f}} \tag{5.6}$$

with $\tilde{\mathcal{M}}_k^{\mathrm{env}}$ the most recent low-level fuzzy map of size $n^{\mathrm{h}} \times n^{\mathrm{v}} \times n^{\mathrm{f}}$ that includes all the fuzzy vectors $\tilde{\boldsymbol{s}}_{c,k}$ corresponding to all cells c with $[\xi_c^{\mathrm{h}}, \xi_c^{\mathrm{v}}]^\top \in \left\{[1,1]^\top, \ldots, [n^{\mathrm{h}}, n^{\mathrm{v}}]^\top\right\}$, $\mathcal{M}_{k+1}^{\mathrm{p}}$ the probability map that has just been updated based on the newly acquired observations, $\mathcal{O}_k$ the set of the most recent observations acquired from the environment by all the robots, and $\tilde{f}_\ell(\cdot)$ a fuzzy mapping that is used for updating the ℓ^{th} fuzzy characteristic that has been associated to each cell in the environment. Although (5.6) is represented in a general way and, thus, contains a considerable number of inputs to the fuzzy model $\tilde{f}_\ell(\cdot)$, in the implementations for SaR each fuzzy model requires only a few number of the inputs that impact each element $\tilde{s}_{c,\ell,k+1}$.

The elements of a fuzzy vector corresponding to a low-level fuzzy map may be described by the following characteristics:

- **Binary vs non-binary:** Binary variables are conventionally restricted to values 0 and 1, whereas non-binary variables may adopt intermediate values. A binary map may also be described through fuzzy logic, which extends the Boolean logic [16]. For instance, the occupancy state of a cell is "empty" or "occupied", or its risk state is "safe" or "dangerous", with these labels represented by fuzzy sets.

- **Static vs dynamic:** Static variables depend only on the current measurements, beliefs, and (fuzzy) inputs, whereas dynamic variables (e.g., cell certainty) have a memory and, hence, depend also on their value in the previous iteration.

Remark. When used for autonomous mission planning of SaR robots, FLMPC leverages the update equations given by (5.6) to be embedded in (5.4a) given in the general formulation of FLMPC in (5.4).

Remark. Fuzzy maps in FLMPC are classified as rewards or constraints and are employed to grade candidate trajectories of the robots, as is shown via (5.4). The third category of fuzzy maps in FLMPC are auxiliary maps that do not directly affect the cost, but are required to be used within $\tilde{f}_\ell(\cdot)$ to be able to update both reward and constraint fuzzy maps.

GRADING CANDIDATE TRAJECTORIES OF A ROBOT

Each candidate trajectory $\hat{\boldsymbol{x}}_k = \{\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_{k+n^{\mathrm{p}}}\}$ for a robot is defined as a sequence that contains the coordinates of the cells that are planned at time step k to be visited at time step κ, with $\kappa \in \{k+1, \ldots, k+n^{\mathrm{p}}\}$. Based on Assumption **A5**, when following trajectory $\hat{\boldsymbol{x}}_k$, the robot also observes all cells in a neighborhood of radius $\delta^{\max}$ of the cells within the trajectory. The number of possible trajectories with size n^{p} and starting from the current state $\boldsymbol{x}_k$ of the robot is finite. A set $\mathcal{X}_k(\boldsymbol{x}_k)$ is defined, with each element $\hat{\mathcal{X}}_k(\hat{\boldsymbol{x}}_k)$ of it also a set that contains one of the possible trajectories for time step k, as well as all those cells that are observed by the robot when traveling through the given trajectory.

To aggregate the satisfaction degrees for all the fuzzy constraints that are imposed on the robot in the given environment, as explained in Section 5.3.2, (5.2) is used. Nevertheless, for aggregating the importance degrees with regard to the fuzzy goals for each candidate trajectory, the following points should be considered:

P1 In SaR it is crucial both to maximize the probability of detecting and locating a human through the robot trajectory and to minimize the time of the detection.

P2 In various cases (especially when the environment is not highly dynamic), visiting the same cell on multiple occasions does not increase the overall goal satisfaction.

P3 In SaR, as explained above, not only the cells that embed the robot trajectory play a role in the overall value of the objective function, but also do the cells in the neighborhood of these cells (i.e., all cells in each element $\hat{\mathcal{X}}_k(\hat{\boldsymbol{x}}_k)$ of $\mathcal{X}_k(\boldsymbol{x}_k)$).

P4 Since the FLMPC problem (5.4) is, in general, nonlinear and non-convex, it is difficult, if not impossible, to guarantee the global optimality of the solutions using nonlinear optimization solvers. Thus, I should prevent the solver from ending up in a sub-optimal region of its search region by generating attraction zones that include high-reward cells and their neighborhoods.

To incorporate these points in the aggregation of the rewards for a candidate trajectory, I augment the system with additional tuning weights, which belong to $[0,1]$ and per time step tune the degree of importance of visiting the cells in $\hat{\mathcal{X}}_k(\hat{\boldsymbol{x}}_k) \in \mathcal{X}_k(\boldsymbol{x}_k)$ with regard to the fuzzy goals. Hence, the tuning weight $w^{\mathrm{tune}}_{c,\kappa_{c,k}}$ for cell $c \in \hat{\mathcal{X}}_k(\hat{\boldsymbol{x}}_k)$ that is planned at time step k to be observed at time step $\kappa_{c,k}$, with $\kappa_{c,k} \in \{k+1, \ldots, k+n^{\mathrm{p}}\}$, is a function of both the distance $\delta_{c,\kappa_{c,k}}$ of the cell from the position of the robot predicted for time step $\kappa_{c,k}$ and $\kappa_{c,k}$ itself, i.e.,

the number of time steps it takes the robot to visit cell c. For all $c \in \hat{\mathcal{X}}_k(\hat{\mathbf{x}}_k)$ there is:

$$w^{\text{tune}}_{c,\kappa_{c,k}} = \tag{5.7a}$$
$$\begin{cases} \max\left\{ \dfrac{1}{-\ln\left(\alpha\left(\delta_{c,\kappa_{c,k}}\right)\beta\left(\kappa_{c,k}\right)\right)+1}, w^{\text{tune}}_{c,\kappa_{c,k-1}} \right\}, & \text{if } \alpha\left(\delta_{c,\kappa_{c,k}}\right) > 0 \\ w^{\text{tune}}_{c,\kappa_{c,k-1}}, & \text{if } \alpha\left(\delta_{c,\kappa_{c,k}}\right) = 0 \end{cases}$$

$$\alpha\left(\delta_{c,\kappa_{c,k}}\right) = \max\left\{0, 1 - \frac{\delta_{c,\kappa_{c,k}}}{\delta^{\max}}\right\} \tag{5.7b}$$

$$\beta\left(\kappa_{c,k}\right) = \gamma^{\kappa_{c,k}} \tag{5.7c}$$

where $\gamma \in [0,1]$ is a constant design parameter. Note that (5.7b) and (5.7c) incorporate points **P4** and **P1**, respectively. The main motivation behind the formulation of (5.7a) is that the tuning weights should be obtained via a monotonic function that assigns values in $[0,1]$ to its outputs, associating an equal importance to both factors that are formulated via (5.7b) and (5.7c). Since the cell may have been visited more than once, in (5.7a) the updated value of a tuning weight is considered only if it is larger than its previous value, which enables incorporation of point **P2** in the computations.

Finally, the overall degree $\mu^{\text{goal}}(\hat{\mathbf{x}}_k)$ of satisfaction for the candidate trajectory $\hat{\mathbf{x}}_k$ for SaR robots $\forall \hat{\mathcal{X}}_k(\hat{\mathbf{x}}_k) \in \mathcal{X}_k(\mathbf{x}_k)$ is obtained by leveraging (5.1) via:

$$\mu^{\text{goal}}(\hat{\mathbf{x}}_k, \boldsymbol{e}_k) = \left(\frac{1}{\max\limits_{\mathcal{Z} \in \mathcal{X}_k(\mathbf{x}_k)} |\mathcal{Z}|} \sum_{c \in \hat{\mathcal{X}}_k(\hat{\mathbf{x}}_k)} \left(\mu^{\text{goal}}_1(c, \boldsymbol{e}_k) \star \ldots \star \mu^{\text{goal}}_{n^{\text{goal}}}(c, \boldsymbol{e}_k) \right)^{w^{\text{goal}} + \frac{1}{w^{\text{tune}}_{c,\kappa_{c,k}}}} \right)^{\frac{1}{w^{\text{goal}}}} \tag{5.8}$$

Note that the maximization operation in the denominator of (5.8) can be performed offline before the optimization problem of FLMPC is solved. This avoids the optimizer from maximizing the cost function by reducing the number of observed cells, which is an undesirable behavior. Finally, by considering the set $\hat{\mathcal{X}}_k(\hat{\mathbf{x}}_k)$, instead of the trajectory $\hat{\mathbf{x}}_k$, in estimation of the overall value of the objective function associated to a trajectory via (5.8), point **P3** is effectively incorporated in the estimations.

Once the aggregated importance degree with regard to the goals and the aggregated satisfaction degree with regard to the constraints have been obtained, (5.4) is implemented to determine an optimal trajectory for the robot.

COORDINATION OF MULTIPLE ROBOTS

To explore an unknown environment, utilizing a multi-robot system is common. To avoid multiple robots visiting the same cells, which hinders the efficiency of the exploration, these robots must be coordinated. In [33], three optimization-based coordinating methods have been compared for multi-robot systems: central control, distributed control with serial

communication among the robots, and distributed control with parallel communication among the robots. As expected, the central optimization-based approach achieves the best results with regard to optimality, due to its global view of the system. The two distributed optimization-based approaches perform very closely, whereas the distributed method with serial communication among the robots shows a faster convergence rate.

In this Chapter, a central system is considered that receives information from all robots regarding their current states, observations, and planned trajectories. The robots receive updated environmental information from this central system. The set $\mathcal{R}$ that includes the indices of all robots is considered. Note that in this case, the variables that have been defined for a robot in the previous sections are shown via an additional subscript index that refers to the robot index. Each time a robot has computed their trajectory and tuning weights $w^{\text{tune}}_{c,\kappa_{c,k},r}$ for this trajectory they have planned at time step $\kappa_{c,k}$ via (5.7a) and $\forall r \in \mathcal{R}$, these weights are sent to the central system. While cooperating, the robots are given weights computed by other robots and use updated cooperative weight $\bar{w}^{\text{tune}}_{c,\kappa_{c,k},r}$ computed with the following equation

$$\bar{w}^{\text{tune}}_{c,\kappa_{c,k},r} = \max\left\{0, w^{\text{tune}}_{c,\kappa_{c,k},r} - \max_{\varepsilon\in(\mathcal{R}\setminus\{r\})} w^{\text{tune}}_{c,\kappa_{c,k},\varepsilon}\right\} \tag{5.9}$$

wherein the tuning weight assigned to a cell by each robot is reduced by the largest tuning weight assigned to that cell by other robots.

This approach bears a resemblance to the previously mentioned distributed controller with serial communication, but a clear distinction emerges. In [33], all controllers solve multiple optimization problems while communicating their decisions through interconnecting variables. The control inputs are only dispatched to robots once convergence has been achieved between all controllers. However, the exploration problem using a multi-robot system lacks such flexibility, due to the long optimization time despite the need for fast decisions. Instead, once the trajectories are determined, the control inputs steer the robots and are sent to the central controller. Additionally, each robot needs to know the updated values of the tuning weights for other robots in its neighborhood, to estimate (5.9). Thus, these values are also shared with the robots.

Nevertheless, this approach has a potential drawback: A robot that earlier decides to visit a cell with a high reward may take the chance away from another robot whose visit to the same cell could potentially result in a situation that would globally be optimal. In other words, this robot may be penalized for attempting to reach the high-reward cell. This is the rationale behind the decision to have the central controller solve a general optimization problem to return a trajectory for each robot. However, this process cannot be executed too frequently, as it necessitates a significant amount of computational resources, particularly given the multiplication of decision variables by the number of robots, which presents a formidable challenge. In Section 5.3.4, it is proposed to use the central system, which not only facilitates the exchange of information but also solves a global optimization problem that assists each

robot in finding an optimal trajectory, allowing for less greedy solutions.

MOTION ENCODING VIA FLMPC

In the implementation of motion encoding, which is explained in Section 5.2.2, two horizons are considered: The maximum number n^{p} of the FLMPC decision variables (that is equal to the prediction horizon of FLMPC) and the maximum number n^{path} of cells that a predicted path consists of. Each decision variable is a 2-element vector that contains a distance and a direction. These two values may be used to generate a sequence of the cells that should be traveled by the robot during the corresponding sampling time. The number of cells traveled in the prediction window is smaller than or equal to a pre-tuned parameter n^{travel}, with $n^{\mathrm{path}} \leq n^{\mathrm{p}} n^{\mathrm{travel}}$. To avoid over-constraining the FLMPC optimization problem, the heading is allowed to continuously vary within the interval $[0, 2\pi]$. After determining the positions of the robot in cells via the optimization problem, the robot is forced to locate itself in the center of the cells.

Remark. If a decision variable would result in the creation of a path long enough to guide the robot outside the environment, the distance element of the decision variable is reduced to the highest value that would allow the robot to remain within the environment.

The rationale for this design is two-fold: On the one hand, allowing for constructing the path from, i.e., n^{p} decision variables provides flexibility in generating optimal paths, especially around complex-shaped obstacles. On the other hand, using the highest length of a path (i.e., $n^{\mathrm{p}} n^{\mathrm{travel}}$) results in a very high computational load. Accordingly, introducing n^{path} enables adaptive reduction of the number of decision variables used when such a complex path for the exploration is not deemed necessary (e.g., no complex-shaped obstacles are around the robot).

During the FLMPC optimization loop stops and trajectories are planned for a robot, they are locally assigned an overall fuzzy grade (see Section 5.3.3) based on the current knowledge of the robot about the environment (i.e., its local maps). Then, as it is explained in Section 5.3.3, these grades are updated according to the plans of other robots that are communicated with the robots via the central system.

The primary rationale for adopting this methodology rather than conventional priority encoding is that when a robot is positioned close to the boundaries of the environment, priority encoding is susceptible to selecting a distance in a direction that is contrary to that of the nearest wall. This is due to the fact that priority encoding samples the position over the entire environment. Consequently, areas close to the boundaries of the environment or obstacles may not be sampled at all, especially if the environment is of a considerable size. This, in SaR, is particularly undesirable since humans may be situated close to the boundaries, but should not remain unattended by the robots.

5.3.4. BI-LEVEL FLMPC BASED ON PC-MPC ARCHITECTURE

As robots operate under limited temporal and spatial horizons n^{p} and n^{path}, respectively, in the event of a large and/or highly dynamic environment, both horizons may need to be extended. Otherwise, upon a relatively short-sighted path planning, the robots may overlook targets (e.g., humans) that are located farther away from them. However, since the robots should make decisions rapidly, given the environmental variations, larger horizons may hinder their applicability.

In Chapter 4, it is proposed to use a bi-level Parent-Child MPC (PC-MPC) architecture combining long-sighted Parent MPC (P-MPC) and short-sighted Child MPC (C-MPC) controllers into two levels where P-MPC makes a plan within a larger prediction window with a larger sampling time (both for decision-making and for modeling the dynamics of the controlled system), which makes the computations more efficient, and C-MPC operates upon a smaller sampling time to compensate for lost optimality because of the less precise predictions of P-MPC. In Parent-Child Fuzzy-Logic-based MPC (PC-FLMPC), to afford the computations, the C-FLMPC deploys a smaller, rolling horizon. These two FLMPC levels interact with each other for decision-making via additional constraints that are formulated and sent to the C-FLMPC via the P-FLMPC, ensuring constraint satisfaction and improved optimality in the long term.

In the real-world, human rescuers are guided by an "incident commander" who oversees the long-term planning and coordination of the rescue team. The PC-FLMPC architecture proposed here allows for to deployment of a long-sighted P-FLMPC system to resemble the incident commander for coordination of the robots that operate according to short-sighted C-FLMPC systems. Next, the details about the PC-FLMPC architecture are provided.

MAPPING FOR LONG-TERM DECISION-MAKING OF THE P-FLMPC

As explained above in the introduction of the Section 5.3.4, the main motivation of the PC-FLMPC architecture is to balance the computational efficiency and the future vision of FLMPC. Since the P-FLMPC system is long-sighted, to make the computations efficient, the map of the FLMPC system is made coarsened by clustering multiple cells into one.

All cells in the environment are dynamically clustered, based on fuzzy membership degrees, by the P-FLMPC system. Each C-FLMPC system, responsible for steering one robot, receives a command from the P-FLMPC system about the next cluster it should explore. Per clustering, each cell in the environment is assigned to one or more fuzzy clusters with varying degrees of membership.

These fuzzy clusters are distinguished based on a high-level fuzzy score that is assigned to them by the P-FLMPC. This score is indicative of the potential accumulated reward that is associated with the fuzzy cluster and is estimated by taking a weighted average of the aggregated grades of all the cells within that cluster, considering their degree of membership in

the cluster. Therefore, for the score $s_{j,k}^{\text{cluster}}$ of the the j^{th} fuzzy cluster at time step k, there is:

$$s_{j,k}^{\text{cluster}} = \left(\frac{\sum_{c=1}^{n^{\text{h}}n^{\text{v}}} \sum_{i=1}^{n_k^{\text{cluster}}} \left(\mu_{i,k}^{\text{cluster}}(c,\boldsymbol{e}_k)\mu_k^{\text{agg}}(c,\boldsymbol{e}_k)\right)^{w^{\text{cluster}}}}{\sum_{c=1}^{n^{\text{h}}n^{\text{v}}} \sum_{i=1}^{n_k^{\text{cluster}}} \mu_{i,k}^{\text{cluster}}(c,\boldsymbol{e}_k)} \right)^{\frac{1}{w^{\text{cluster}}}}, \qquad j \in \left\{1,\ldots,n_k^{\text{cluster}}\right\} \tag{5.10a}$$

where the following equation holds:

$$\mu_k^{\text{agg}}(c,\boldsymbol{e}_k) = \min\left\{\max\left(\mu_1^{\text{goal}}(c,\boldsymbol{e}_k),\ldots,\mu_{n^{\text{goal}}}^{\text{goal}}(c,\boldsymbol{e}_k)\right),\mu_1^{\text{con}}(c,\boldsymbol{e}_k),\ldots,\mu_{n^{\text{con}}}^{\text{con}}(c,\boldsymbol{e}_k)\right\} \tag{5.10b}$$

In (5.10a), $\mu_{i,k}^{\text{cluster}}(c,\boldsymbol{e}_k)$ is the membership degree of cell c to the i^{th} cluster at time step k, n_k^{cluster} is the total number of fuzzy clusters determined by the P-FLMPC system at time step k, and $n^{\text{h}}n^{\text{v}}$ is the total number of the cells in the environment (note that some of these cells may be associated to the j^{th} fuzzy cluster with a null membership degree). The rationale behind (5.10b) is the same as for (5.1)-(5.3), but instead of a trajectory, here a single cell is considered. The index k is used for the membership degrees to indicate that these degrees may vary in time as the global map of the environment is updated by the P-FLMPC system based on the local maps updated by C-FLMPC systems. Also, as before, the weight w^{cluster} is used with the same purpose (increasing the importance of a few high scores over multiple low scores).

Each fuzzy cluster is associated with a state at time step k by the P-FLMPC system, where this state adopts one of the following 4 realizations: "unexplored", "to be explored", "being explored", "explored". In case at time step k the fuzzy cluster has never been scanned by any robots before, its state is "unexplored". However, if this cluster is within the planning of one or more robots at time step k, its state is updated to "to be explored". If at time step k, one or multiple robots are already exploring the fuzzy cluster, its state at this time step is "being explored". Finally, if the fuzzy cluster has already been explored before time step k, the state of this cluster at time step k is "explored".

Whenever the score of a fuzzy cluster determined via (5.10a) is less than a specified threshold, designated as s^{exp}, the state of the fuzzy cluster is classified as "explored". Upon completion of its exploration, the robot is no longer associated with a group. In such a scenario, the P-FLMPC solves the original optimization problem once more, incorporating supplementary constraints to ensure that the initial group assigned to each robot (excluding those without a group) remains in the group they explore. Once the optimization problem has been resolved, the subsequent group is assigned to the robot that has completed the exploration of its group. The score of the fuzzy clusters may increase in the course of the mission, particularly in dynamic environments, where the degree of uncertainty associated with a fuzzy cluster tends to grow when it is not subjected to prolonged exploration. Consequently, when

the score of a fuzzy cluster exceeds a designated threshold value, s^{unexp}, the state of that fuzzy cluster is classified as "unexplored" once more.

Remark. The value of s^{exp} should be tuned to be adequately smaller than that of s^{unexp}, to prevent the state of fuzzy clusters from undergoing rapid and arbitrary changes under the influence of random measurements.

HIGH-LEVEL LONG-TERM PLANNING BY THE P-FLMPC

The P-FLMPC system generates a high-level plan for each robot by assigning a path to the robot that represents a sequence of fuzzy clusters with state "unexplored". The final cost is calculated in a manner identical to that previously described for FLMPC, except for three modifications. To guarantee that the high-level planning covers the entirety of the environment, the P-FLMPC system operates under an additional constraint, i.e., each single fuzzy cluster must be included in at least one of the proposed paths for the robots.

Second, the decision variables of the P-FLMPC system include the destinations rather than directions with distances. This is because, otherwise, the previously mentioned constraint would be unnecessarily challenging and inefficient to implement. It should be noted that distances with directions are utilized by FLMPC to guarantee that multiple unbiased steps would be incorporated. However, this is not a concern for P-FLMPC, as it is required to encompass all possible destinations due to the constraint.

To grade the trajectories composed of the fuzzy clusters, the P-FLMPC system utilizes (5.1) (instead of (5.3) as there are no constraints for the P-FLMPC system because they are supposed to be dealt with directly by the C-FLMPC system, as constraints can be changed rapidly based on new measurements), based on (5.2) and (5.8). However, instead of the membership degrees for the cells included in the trajectories, the scores of the fuzzy clusters involved, as computed by (5.10), are considered. Accordingly, a hyperparameter $w^{\text{goal,P}}$ is introduced. This term bears the same meaning as the previously defined w^{goal}, yet it is employed in the context of grading clusters as opposed to trajectories (5.10). It may, therefore, assume a distinct value.

Grading high-level trajectories by P-FLMPC: The sequence of "unexplored" fuzzy clusters assigned to a robot by the P-FLMPC system may include clusters that are not the same as or are even far from the current fuzzy cluster that embeds the robot. This allows the P-FLMPC system to redistribute the robots whenever a considerable number of them have been spawned in or around the same fuzzy cluster, which impedes the global optimality of the multi-robot system. Consequently, the time required to reach the same fuzzy cluster may vary for the robots assigned to the same trajectory. Moreover, exploring the "unexplored" fuzzy clusters that are assigned to the robots by the P-FLMPC system may require the robots to traverse through intermediate fuzzy clusters with a state rather than being unexplored. Such clusters, unlike fuzzy clusters of state unexplored, do not contribute as much to the overall reward that is collected by the robot for accomplishing the exploration tasks that are

assigned to it by the FLMPC system (see Section 5.3.3 for details, taking into account that the discussions given in that section for the cells should be generalized for the fuzzy clusters when discussing the high-level decision-making). Therefore, these effects should be incorporated in grading the candidate trajectories in the high-level decision-making. Accordingly, (5.7) for fuzzy cluster indexed $i \in \{1,\dots,|\mathbb{P}_k|\}$ that is planned to be explored at time step $\kappa_{j,k}$ and that belongs to trajectory $\mathbb{P}_k = \{[\lambda^{\mathrm{h}}_{1,k},\lambda^{\mathrm{v}}_{1,k}]^\top,\dots,[\lambda^{\mathrm{h}}_{|\mathbb{P}_k|,k},\lambda^{\mathrm{v}}_{|\mathbb{P}_k|,k}]^\top\}$ determined at time step k by the P-FLMPC system for a robot, is reformulated via (5.11), i.e.:

$$w^{\mathrm{tune,P}}_{j,\kappa_{j,k}} = \max\left\{\frac{1}{-\ln\left(\beta\left(\kappa_{j,k}\right)\right)+1}, w^{\mathrm{tune,P}}_{c,\kappa_{j,k}-1}\right\}, \tag{5.11a}$$

$$\beta\left(\kappa_{j,k}\right) = \gamma^{\sum_{i=1}^{j}\left\|[\lambda^{\mathrm{h}}_{i,k},\lambda^{\mathrm{v}}_{i,k}]^\top-[\lambda^{\mathrm{h}}_{i-1,k},\lambda^{\mathrm{v}}_{i-1,k}]^\top\right\|} \tag{5.11b}$$

where $w^{\mathrm{tune,P}}_{j,\kappa_{j,k}}$ is the tuning weight for cluster j that is planned via the P-FLMPC system at time step k to be explored at time step $\kappa_{j,k}$ by the robot. Note that $\kappa_{j,k} \in \{k+1,\dots,k+n^{\mathrm{p}}\}$ and $|\mathbb{P}_k|$ shows the total number of the fuzzy clusters in path $\mathbb{P}_k$. Moreover, $[\lambda^{\mathrm{h}}_{i,k},\lambda^{\mathrm{v}}_{i,k}]^\top$ shows the coordinates of the center of the i^{th} fuzzy cluster in path $\mathbb{P}_k$ determined at time step k, whereas $[\lambda^{\mathrm{h}}_{0,k},\lambda^{\mathrm{v}}_{0,k}]^\top$ is the coordinates of the robot that is assigned path $\mathbb{P}_k$ by the P-FLMPC system at time step k.

Re-structuring fuzzy clusters with internally inaccessible cells: While exploring a fuzzy cluster, reaching some cells in that cluster may turn out to be infeasible from the interior of the cluster. This may be, for instance, due to structures, such as walls, that have not been discerned earlier: See Figure 5.1, where the entire rectangular shape shows one fuzzy cluster, in which the cells in the three separate yellow sub-clusters (A, B, C) are not accessible to each other, due to the existence of the wall (shown in dark blue). Subsequently, such a fuzzy cluster is re-clustered into smaller fuzzy sub-clusters. The resulting fuzzy sub-cluster that includes the largest portion of the cells in the original fuzzy cluster (for instance, sub-cluster C in Figure 5.1) is kept in the original trajectory and replaces the original fuzzy cluster in that trajectory. The remaining fuzzy sub-clusters are either merged with other existing fuzzy clusters by the P-FLMPC system, or are excluded from the map in case there is no way to reach them based on the current configuration of the robots (this, for instance, holds for sub-clusters B in Figure 5.1).

To reallocate the resulting fuzzy clusters that have been discarded from the original trajectory (for instance, sub-cluster A in Figure 5.1), the P-FLMPC system first assesses whether or not at least one cell of the cluster has previously been assigned to any other fuzzy clusters with a non-zero membership degree(for instance, in Figure 5.1, some cells jointly belongs to fuzzy (sub-)clusters 2 and A, as well to 3 and A. Then, the mean value for the membership degrees of the cells within the resulting fuzzy cluster to all other such fuzzy clusters is estimated, and at the end, the resulting fuzzy cluster is merged with the fuzzy cluster that corresponds to the highest mean value of membership degrees. Note that after being merged, the transferred cells

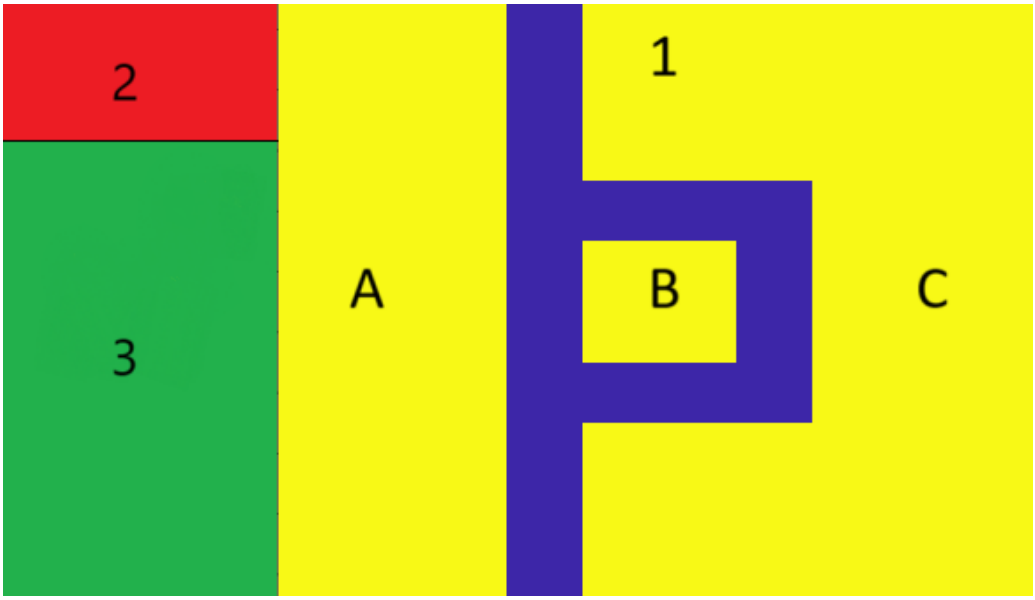


Figure 5.1: Three fuzzy clusters, numbered 1 to 3, are illustrated in different colors: In this case, there is a wall (shown in dark blue) that is identified, leading to the division of the first fuzzy cluster into three smaller fuzzy sub-clusters (shown in yellow). Each sub-cluster is represented by a letter A, B, or C. Given that the sub-cluster C is the largest (i.e., it includes more environmental cells), it is considered as the fuzzy cluster 1 that should replace the original fuzzy cluster 1. Since the fuzzy sub-cluster B that is encountered by the wall is not connected to any other fuzzy (sub-)clusters, it cannot be included in any trajectories by the P-FLMPC system. The fuzzy sub-cluster A is to be merged with cluster 3 by the P-FLMPC system.

adopt the membership degrees they have with respect to the new fuzzy cluster they are merged into.

LOW-LEVEL SHORT-TERM PLANNING BY THE C-FLMPC

Once the high-level plan has been made by the P-FLMPC system, each robot is assigned to a trajectory of fuzzy clusters for exploration. Moreover, the robot receives a local fuzzy map from the P-FLMPC system, updated based on data collected from all robots and using (5.6), for the region that embeds the given high-level trajectory of the robot. The C-FLMPC systems implement (5.7), but to ensure that each robot can devote its full attention to its designated fuzzy cluster, (5.7a) is transformed into (5.12a), where an additional weight, denoted by w_{hl}, is introduced. The value of this weight is dependent upon whether the fuzzy cluster is currently in a state “to be explored” or “being explored”.

$$w_{c,\kappa_{c,k}}^{\text{tune,C}} = \tag{5.12a}$$

$$\begin{cases} \max\left\{ \dfrac{1}{-\ln\left(\alpha\left(\delta_{c,\kappa_{c,k}}\right)\beta\left(\kappa_{c,k}\right)w^{\text{HL,C}}(c)\right)+1}, w_{c,\kappa_{c,k}-1}^{\text{tune,C}} \right\}, & \text{if } \alpha\left(\delta_{c,\kappa_{c,k}}\right)w^{\text{HL,C}}(c) > 0 \\ w_{c,\kappa_{c,k}-1}^{\text{tune,C}}, & \text{if } \alpha\left(\delta_{c,\kappa_{c,k}}\right)w^{\text{HL,C}}(c) = 0 \end{cases}$$

where in case the fuzzy cluster has the state "to be explored", there is:

$$w^{\text{HL,C}}(c) = \begin{cases} \exp\left(\dfrac{\left\|[\lambda_{j,k}^{\text{h}},\lambda_{j,k}^{\text{v}}]^\top - [\lambda_{0,k}^{\text{h}},\lambda_{0,k}^{\text{v}}]^\top\right\|}{2\sigma^2}\right), & \text{if } \left\|[\lambda_{j,k}^{\text{h}},\lambda_{j,k}^{\text{v}}]^\top - [\lambda_{0,k}^{\text{h}},\lambda_{0,k}^{\text{v}}]^\top\right\| < \eta \\ 0, & \text{if } \left\|[\lambda_{j,k}^{\text{h}},\lambda_{j,k}^{\text{v}}]^\top - [\lambda_{0,k}^{\text{h}},\lambda_{0,k}^{\text{v}}]^\top\right\| \geq \eta \end{cases} \tag{5.12b}$$

Otherwise, in case the fuzzy cluster has the state being explored, there is:

$$w^{\text{HL,C}}(c) = \mu_{i,k}^{\text{cluster}}(c) \tag{5.12c}$$

where $w_{c,\kappa_{c,k}}^{\text{tune,C}}$ is the tuning weight for cell c that is planned via the C-FLMPC system at time step k to be explored at time step $\kappa_{c,k}$, $w^{\text{HL,C}}(c)$ is the correction weight that is determined for cell c in the course of the C-FLMPC planning and via the high-level P-FLMPC system, j is the index of the cluster that is planned that for the robot to explore, and η is a user-defined threshold.

If the robot is not situated within the designated fuzzy cluster, the appropriate course of action is to guide the robot to the cluster. The reward is inversely proportional to the distance of the robot from the center of the fuzzy cluster. In this equation, $\left\|[\lambda_{j,k}^{\text{h}},\lambda_{j,k}^{\text{v}}]^\top - [\lambda_{0,k}^{\text{h}},\lambda_{0,k}^{\text{v}}]^\top\right\|$ represents the distance between the robot and the center of the fuzzy cluster, while the parameter, designated as σ is a variable chosen by the designer and expressed in meters. However, to guarantee that the robot proceeds in the direction of the selected fuzzy cluster, a weight of 0 is applied instead if the robot's movements would increase the direction to the center of the fuzzy cluster. If a robot is already within the fuzzy cluster, the membership of the cells is transformed into weights $w^{\text{HL,C}}(c)$.

BI-LEVEL ARCHITECTURE

To illustrate the operational principles of the system from the perspective of a single robot, a diagram (see Figure 5.2) is included. Starting from the right-upper corner, it can be noticed that the robot collects measurements that are affected by noise over time. Subsequently, the central controller generates a probability map $\mathcal{M}_k^{\text{p}}$ based on the received data. Subsequently, fuzzy maps $\tilde{\mathcal{M}}_k^{\text{env}}$ are generated based on the measurements, the current fuzzy map, and the probability map. The fuzzy maps are transmitted to the fuzzy cluster allocator, which generates the high-level fuzzy scores. These include the computation of both the membership of all cells

in each fuzzy cluster (which may change over time) and the score of each fuzzy cluster. The P-FLMPC then sends the next fuzzy cluster to be explored. The cluster is calculated at the beginning of the mission and again each time the currently explored fuzzy cluster is finished.

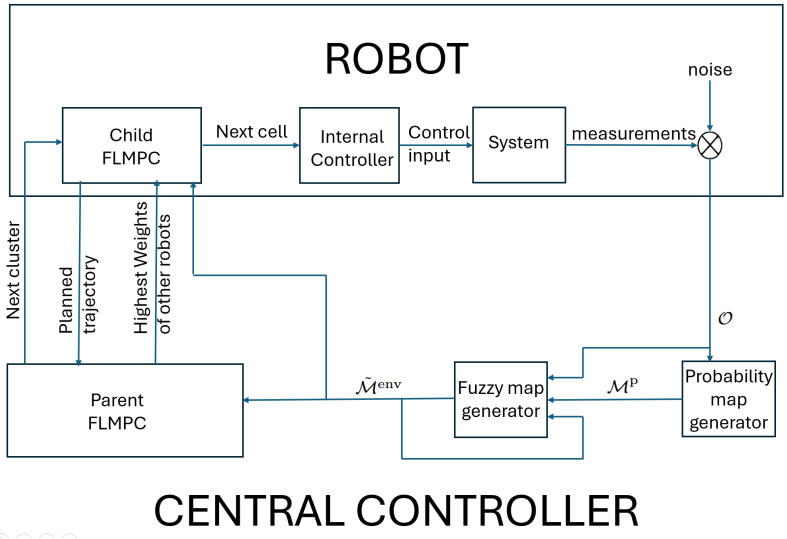


Figure 5.2: The control diagram is presented from the perspective of a single robot.

The C-FLMPC is connected to a robot and requires knowledge of the current cluster to which it is allocated, the trajectories of other robots, and the fuzzy maps. Based on the aforementioned knowledge, a planned trajectory of cells to be visited is computed. Subsequently, the planned trajectory is transmitted to the central controller, thereby enabling other robots to become acquainted with the local robot's plan and circumventing a scenario in which two robots attempt to rescue the same individual. The C-FLMPC then transmits the reference r to the internal controller, which is presumed to be equipped to the robot. This internal controller could be, for example, ASDD-TMPC (see Chapter 3). Ultimately, the lowest-level controller oversees direct robot control through inputs u.

Algorithm 1 provides a comprehensive understanding of the operational principles of the central controller. The specific instructions for implementing each line of the pseudo-code given in Algorithm 1 are detailed in the current Section.

Algorithm 1 This algorithm shows how the P-FLMPC system works for SaR robotics from the point-of-view of the central controller.

Require: An unknown environment, robots are spawned in known cells
  Generate initial probability map $\mathcal{M}_k^{\mathrm{p}}$
  Generate initial fuzzy maps $\tilde{\mathcal{M}}_k^{\mathrm{env}}$
  Generate initial fuzzy clusters, estimating the scores via (5.10a)
  while Mission is not finished **do**
    Generate new high-level plan via FLMPC
    Allocate new trajectories of fuzzy clusters to robots that are not assigned to a trajectory
    while scores for all fuzzy clusters are above a given threshold **do**
      for each robot in $\mathcal{R}$ **do**
        Send fuzzy maps to the robot
        Compute all weights used in (5.9)
        Send the weights to the robot
        Get planned trajectory from the robot
        Get measurements from the robot
      end for
      Update probability map
      Update fuzzy maps
      if a new "wall" is found **then**
        if it is required to divide a fuzzy cluster into smaller fuzzy clusters **then**
          Allocate or delete resulting fuzzy clusters
        end if
      end if
      Update scores of fuzzy clusters
    end while
  end while

5.4. CASE STUDIES

This Section presents two case studies: The first one compares an MPC-based controller with a stochastic cost function against FLMPC, evaluating their performance based on the time required to detect all humans in the environment and the computation time. Both controllers utilize the same optimization solvers, encoding methods, probability map updates, and hyperparameters (if shared). The second case study expands the analysis by comparing the performance of single-level FLMPC and bi-level PC-FLMPC architectures in a larger environment.

After evaluating various general-purpose nonlinear optimization solvers from the global optimization toolbox of MATLAB, the PS algorithm [34, 35] proved to be the most effective for this problem, balancing both the realized values for the cost function and the computation time.

The code used to generate the data and results (including figures) is available in [36]. To run all simulations efficiently within a reasonable time frame, the Delft Blue supercomputer at Delft University of Technology [37] is used because it provides 64 CPU cores per node with a

high memory throughput of 250 GB RAM.

5.4.1. CASE STUDY 1: MPC WITH STOCHASTIC COST FUNCTIONS COMPARED TO FLMPC

To evaluate the performance of FLMPC, this section presents an illustrative case study focused on a relatively straightforward problem, i.e., locating humans in an unknown environment using a multi-robot system in the shortest possible time. Each environment is randomly generated, and, except for its size, is unknown. The experiment is repeated across 100 randomly generated environments, comparing two controllers: MPC with a stochastic cost function based on prior work [14] and a single-level FLMPC. While randomness in the initial placement of humans is inherent in the search process, this variability is mitigated through multiple simulation runs. Figure 5.3 illustrates an example of a randomly generated environment.

Both controllers use the same parameters: n^{p}=5, n^{travel}=14, and n^{path}=20. Moreover, to maintain a balanced trade-off between computational efficiency and coordination of the robots, the central controller is executed every 5 time steps.

Remark. Note, term MPC with stochastic cost function is used instead of Stochastic MPC (SMPC). This is because, in general, it is assumed that for MPC to be classified as SMPC, there should be implemented chance constraints. [14] proposes MPC without probabilistic constraints.

PROBLEM SETUP

The environment is composed of 40×40 cells, i.e. $n^{\mathrm{h}} = n^{\mathrm{v}} = 40$. Each cell in the environment is described by a state from the ordered set $\mathcal{S}$ ={empty, human, obstacle} (i.e., ρ_1 = empty, ρ_2 = human, ρ_3 = obstacle). No prior information about the real states of the environment is available. The mission commences with the random placement of n^{rob} robots, with $2 \leq n^{\mathrm{rob}} \leq 10$.

Per step, each robot may traverse one cell in any direction, including diagonally. The movement of the robots is unconstrained. Upon reaching a cell containing a human, the human is deemed rescued, and the cell is rendered empty afterwards. If a robot attempts to access a cell with an obstacle, the movement is canceled. Subsequent to the completion of each movement, the robots obtain data regarding cells in a neighborhood of maximum $\delta^{\max} = 6$ cells. Measurement accuracy decreases as the distance between the robot and a cell increases. To apply (5.5) to cell c where the robot is located at time step k, the following relationship is used model the perception accuracy:

$$p\left(o_{c,k}|\rho_i\right) = \begin{cases} 0.02 & o_{c,k} \neq \rho_i \\ 0.96 & o_{c,k} = \rho_i \end{cases} \qquad \forall \left(o_{c,k}, \rho_i\right) \in \mathcal{S} \times \mathcal{S} \tag{5.13}$$

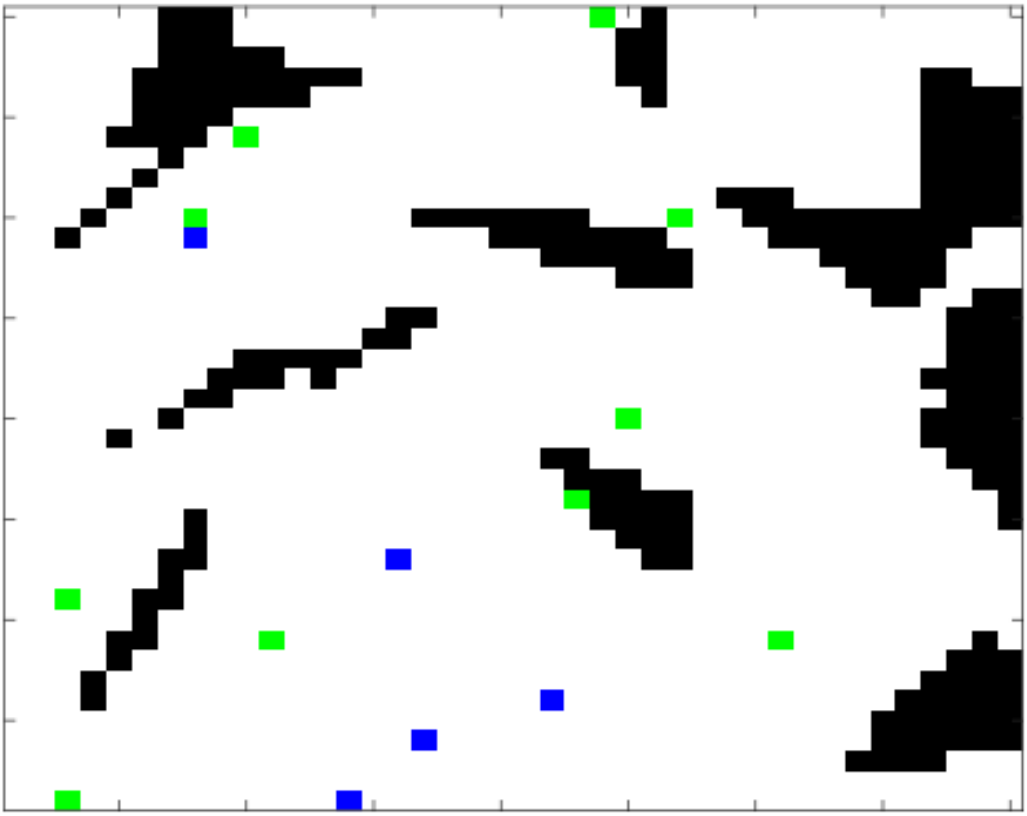


Figure 5.3: A randomly generated environment for the first case study: In this environment, white is used to represent empty spaces, black is used to represent the obstacles, green is used to represent humans, and blue is used to represent the robots.

The perception accuracy for neighboring cells within the perception field of the robot depends on their distance from the cell where the robot is located. The larger the distance, the higher the uncertainty of the observations. To introduce the impact of the distance, detectability $d_{c',c,k}$ of cell c', which is $\delta_{c',\kappa_{c',k}}$ cells away from cell c where the robot is located and is planned to be observed by the robot at time step $\kappa_{c',k}$, at time step k is formulated by:

$$d_{c',c,k} = \max\left\{1 - \left(\frac{\delta_{c',\kappa_{c',k}}}{\delta^{\max}}\right)^2, 0\right\} \tag{5.14}$$

Cell c' with a non-zero detectability is perceived by the robot, where the perception accuracy $\forall\left(o_{c',k}, \rho_i\right) \in \mathcal{S} \times \mathcal{S}$ is determined via:

$$p\left(o_{c',k}|\rho_i\right) = p\left(o_{c,k}|\rho_i\right) d_{c',c,k} + 0.25\left(1 - d_{c',c,k}\right) \tag{5.15}$$

where (5.15) holds only for cells with a positive detectability value. Obviously, when c' and c

refer to the same cell, the detectability of the cell according to (5.14) is 1 and thus, (5.15) boils down to (5.13).

For simplicity, a static environment is assumed, where humans and obstacles do not move. Simulations continue until all humans are rescued or a predefined time limit is reached, calculated by $500/n^{\text{rob}}$ time steps based on the number of deployed robots. Extensive simulations confirm that this termination criterion prevents outliers, such as a robot failing to receive reliable measurements multiple times in a row, from skewing the results. As robots lack initial knowledge about the environment, their initial belief vector for each cell is set to $[0.34, 0.33, 0.33]^\top$.

MPC WITH A STOCHASTIC COST FUNCTION

The goal is to replicate the algorithm from [14], but modifications are required due to differences with the case study. The original cost function in [14] is a weighted summation of four terms (search effectiveness, uncertainty, detection response, and motion cost). For this case study, the first two terms are retained, but the detection response is removed due to a higher likelihood of false human detections. Without this modification, robots may move toward falsely reported human locations. To address this issue, a new term is introduced to reward robots for reaching cells where the likelihood of the presence of a human is close to 100%. Moreover, in the absence of maneuvering penalties, the motion cost term is unnecessary. Unlike the obstacle-free environment in [14], an additional constraint is required in this case study, i.e., a penalty function that increases the cost when a robot enters a cell with a high probability of containing an obstacle. More details can be found in Appendix A5

FLMPC

Designing FLMPC involves three key steps, i.e., defining the fuzzy variables, modeling them using membership functions, and setting hyperparameters to regulate the behavior of the controller. For this case study, 5 fuzzy variables, namely "passability", human detection reward", "exploration reward", "uncertainty", and "measurement consistency" have been considered (see Table 5.2).

Table 5.2: List of all fuzzy variables used by FLMPC

	Reward	Constraint	Auxiliary	Dynamic
Passability		✓		
Human detection reward	✓			
Exploration reward	✓			
Uncertainty			✓	✓
Measurement consistency			✓	

The rewards and constraints of the environment are identified based on the goals and

setup of the case study. Locating humans is the primary objective in search and rescue. Thus, "human detection reward" is one of the fuzzy variables that is chosen as a reward. Moreover, incorporating additional goals improves the behavior of robots, as evidenced by other approaches that are based on multiple goals (see, e.g., [14, 29]). We, thus, additionally introduce the fuzzy variable "exploration reward" for exploring the unknown environment.

Remark. The maximum degree for the "human detection reward", as shown in Figure 5.4 (top right), is intentionally set below 1. This, based on (5.8), ensures that tuning weights can still influence the decisions of robots, an effect that disappears when the membership degrees in (5.8) reach 1. The impact of the tuning weights, as it is explained in Section 5.3.3, is crucial in penalizing robots for delaying a rescue.

Remark. Rescuing humans should be prioritized over exploring the environment. This priority explains why the maximum exploration reward is set (significantly) below 1. Consequently, a high reward is assigned only when the robot achieves a high degree of certainty about the location of the human.

For the constraints, as the robot can only interact with passable cells, the fuzzy variable "passability" has been introduced. For the fuzzy variables "passability", "human detection reward", and "exploration reward", a static model with a single input suffices. For instance, for "passability" the membership function that is shown in Figure 5.4 (top left) is used, where the input is the likelihood that the state of a cell is "obstacle". "Human detection reward" depends on the probability of successfully locating a person, i.e., the robot should distinguish between actual victims and observations that are wrongly reported as humans. The membership function used for fuzzy variable "human detection reward" is illustrated in Figure 5.4 (top right). Without prior knowledge about the location of humans, the robot should be encouraged to explore cells with higher uncertainty. Thus, the membership function for "exploration reward" receives the uncertainty of a cell as input (see Figure 5.4, bottom left).

For the fuzzy variable "uncertainty" a multi-dimensional membership function is used. Although some papers [12, 14, 25] model the uncertainty as a function of the cell detectability only, this approach faces limitations. Notably, it does not account for conflicting measurements. In fact, if new measurements conflict with existing cell knowledge, the inference about the uncertainty should differ from cases where observations align. This prevents robots from prematurely considering a cell as explored when recent measurements do not provide a clear conclusion. To account for the consistency of the measurements obtained for a cell, a dynamic fuzzy model for uncertainty with two inputs is deployed: the most recent certainty degree and current measurement consistency degree (see the left plot in Figure 5.5). "Measurement consistency" is an auxiliary fuzzy variable that reflects whether the new measurement confirms or contradicts the existing knowledge (see the right plot in Figure 5.5). Its membership function receives two inputs: detectability of the cell and the likelihood of a measurement. When the measurement consistency is below 0.5, the measurement is considered not aligned

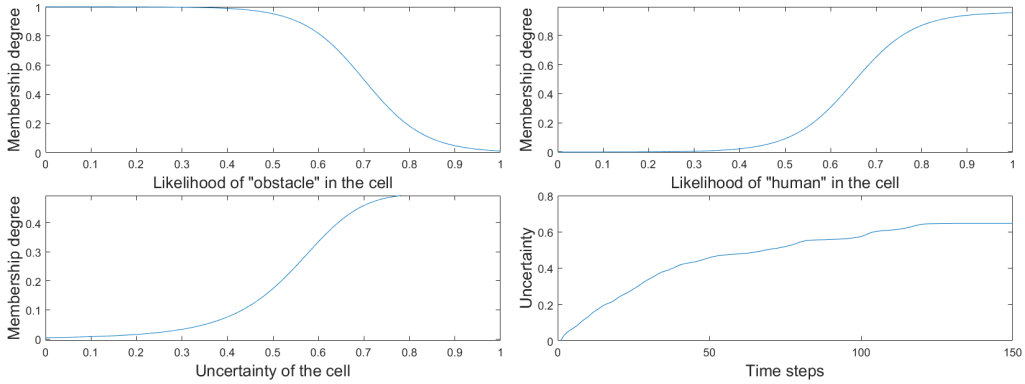


Figure 5.4: Top left: Membership function for the passability of a cell, given the likelihood that the state of the cell is "obstacle". Top right: Membership function for human detection reward for a cell, given the likelihood that the state of the cell is "human". Bottom left: Membership function for exploration reward for a cell, given the uncertainty of the cell within $[0,1]$. Bottom right: Evolution of the uncertainty degree for a cell; the uncertainty degree rises over time as long as the cell is not visited/observed.

with prior knowledge. This appears to have a small to no impact on the new measurements on the updated uncertainty degree (see the left plot in Figure 5.5).

Initially, the membership degrees of uncertainty for all the cells are set to 1, which gradually decreases by gathering measurements for the cells. Figure 5.4 (bottom right) represents the evolution of the uncertainty degree of a cell when no measurements are received for the cell. The uncertainty degrees smaller than 0.648 increase slowly until reaching the maximum value of 0.648.

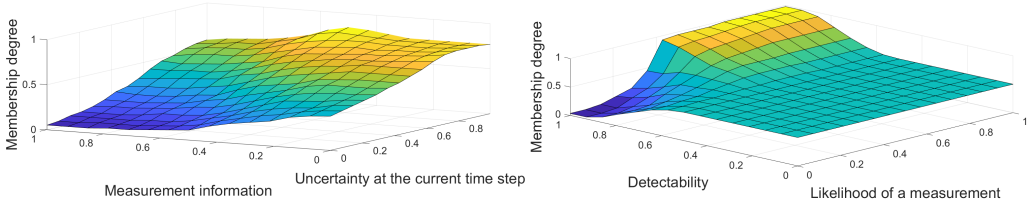


Figure 5.5: Left: Membership function for "uncertainty" when the cell is observed, given the most recent value for the uncertainty degree and the measurement consistency degree. Right: Membership function for "measurement consistency", given the detectability of the cell and the likelihood for a measurement.

In the implementation of the case study, I use $\gamma = 0.965$, $\delta^{\max} = 5$, $w^{\text{goal}} = 20$, $w^{\text{con}} = 5$, $w^{\text{agg}} = 1$, and $n^{\text{p}} = 5$ in (5.2),(5.3),(5.7),(5.8).

RESULTS FOR CASE STUDY 1

The results of the MPC controller with a stochastic cost function and the FLMPC controller are compared, starting with computational complexity. Although both methods use the same number of particles and iterations in the optimization solver, the time needed to find an optimal solution differs significantly. On average, MPC with a stochastic cost function takes more than 60 times longer than FLMPC. This difference arises because grading a single trajectory in MPC with a stochastic cost function demands substantial computation. Updating the probability and certainty maps is particularly time-consuming compared to solving the optimization problem (5.4).

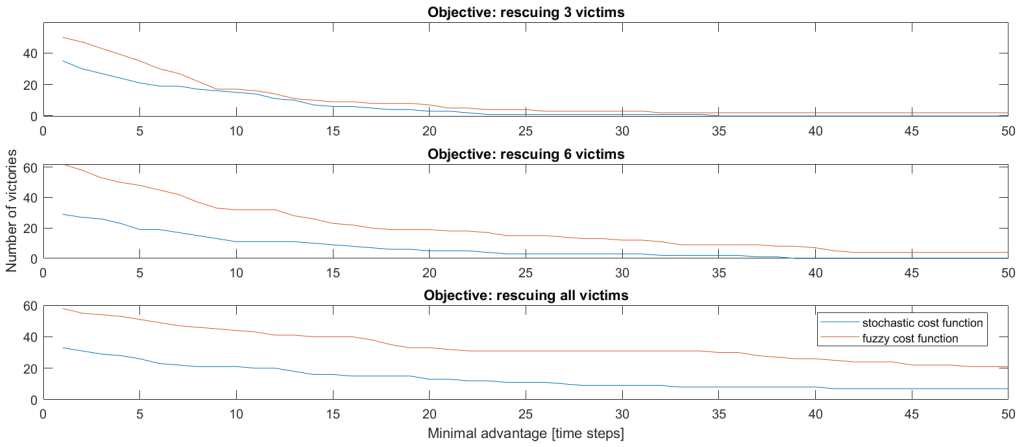


Figure 5.6: Number of times that each controller is the winner for a specific milestone, shown above each figure, and its advantage. For instance, in 20 simulations, FLMPC wins in saving 7 victims with an advantage of at least 24 time steps.

I compare the performance of the controllers in terms of both speed and consistency in rescuing the humans. For a fair comparison, simulations for 100 environmental conditions are performed to account for the randomness related to initializing the positions, and the initial decisions made by the robots are characterized by a high degree of randomness, a consequence of their limited environmental knowledge. Comparing the mean completion times directly for the environmental setups is problematic, since the time required to finish a simulation is influenced by the generated environment and initial conditions. What seems like a short time in one scenario may be long in another, making a simple mean duration comparison ineffective. Instead, I assess how often each controller rescues people faster than the other controller and the time difference between them. In a simulation, I define a winner as the controller that first reaches a chosen milestone (e.g., saving 6 out of 10 humans). The advantage is the absolute time difference between both controllers in reaching the milestone. If one controller reaches the milestone before the simulation ends while the other one does

not, the advantage is considered infinite.

Figure 5.6 illustrates the win frequency of each controller and the advantage gained. It also presents results for two intermediate goals, next to the goal of rescuing all victims. The plots show that FLMPC outperforms MPC with a stochastic cost function in 6 out of 10 cases. Moreover, FLMPC achieves **four times more victories** with a significant advantage of at least 35 time steps compared to MPC with a stochastic cost function. This indicates that FLMPC not only wins more frequently, but is also much more efficient at locating and rescuing victims in significantly less time

5.4.2. CASE STUDY 2: COMPARISON BETWEEN SINGLE-LEVEL FLMPC AND BI-LEVEL PC-FLMPC

To highlight the importance of a bi-level PC-FLMPC approach, this section examines the same problem as in the first case study but in a larger environment of 40000 cells, with 200 cells in each direction and a simulation limit of 3000 time steps. The environment is randomly generated, while the number of robots remained 3. A bi-level PC-FLMPC system is compared to a single-level FLMPC system that replicates the C-FLMPC level without guidance from the P-FLMPC level.

Per time step and for each robot, the C-FLMPC optimization problem considers the cells within an area of 51 cells $\times$ 51 cells, centered on the robot. This means that each robot makes decisions based on less than 7% of the cells. This is expected to impact the global performance of the single-level FLMPC system. The only distinction between the single-level C-FLMPC system in this case study and the FLMPC system from the first case study (see Section 5.4.1) lies in how the fuzzy uncertainty map is updated. If a cell is not observed, its uncertainty degree increases 100 times more slowly than before to prevent a rapid loss of certainty in the early stages of exploration.

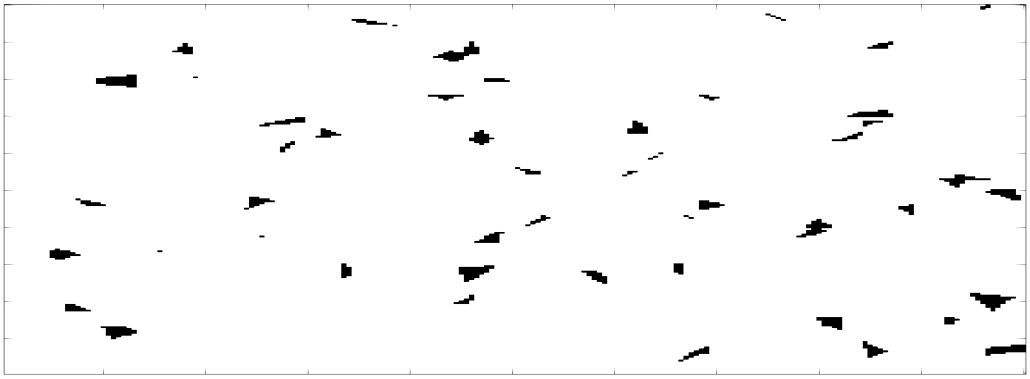


Figure 5.7: A randomly generated environment for the second case study: In this environment, white is used to represent empty spaces, black is used to represent the obstacles, green is used to represent humans, and blue is used to represent the robots.

BI-LEVEL PC-FLMPC

To implement the bi-level PC-FLMPC architecture, the environment is divided into 25 square-shaped sub-areas, including 40 cells per side (i.e., each sub-area is of the same size as the entire environment in the first case study). Then fuzzy clusters are initialized by including one of these sub-areas, where the membership degree $\mu_{i,0}^{\text{cluster}}(c, \boldsymbol{e}_0)$ of each cell c of this sub-area in the i^{th} cluster is initialized to 1. To enhance flexibility, allowing robots to exit their designated fuzzy cluster when beneficial, 3 cells in the environment from the edge of the square-shaped sub-area are also included in the fuzzy cluster, with a membership degree given by:

$$\mu_{i,0}^{\text{cluster}}(c, \boldsymbol{e}_0) = \left(1 - \frac{\delta_{c,i,0}^{\text{cluster}}}{4}\right)^2 \tag{5.16}$$

where $\delta_{c,i,0}^{\text{cluster}}$ is the distance of the cell from the edge of fuzzy cluster i at time step 0.

For long-term planning via the P-FLMPC system (see Section 5.3.4), the state of each cluster should be determined. $s^{\text{exp}} = 0.22$ and $s^{\text{unexp}} = 0.4$, are used and the score of the clusters is initialized slightly below 0.5, since the cells are initially unexplored and there is no information about victims. If the passability of a cell is below 0.1, the cell is treated as an obstacle within the fuzzy cluster. If there is a row of such cells, the fuzzy cluster is divided into sub-clusters (see Figure 5.1). Moreover, $w^{\text{cluster}} = 5$ in (5.10a), $w^{\text{goal,P}} = 1$ as is explained in Section 5.3.4, $\gamma = 0.98$ in (5.11b), and $\sigma = 60$ in (5.12b) are used.

General solvers struggle with solving the optimization problem of P-FLMPC, mainly due to the constraint that each fuzzy cluster must be assigned to exactly one robot. The problem closely resembles the vehicle routing problem [38] or the multiple traveling salesman problem [39], but unlike traditional formulations of these problems that focus on minimizing time or distance, the P-FLMPC optimization problem involves a multi-objective cost function. To efficiently solve this problem, a GA is implemented to offer fast convergence and extensibility, inspired by the one in [40]. This algorithm introduces customized mutation and crossover operators for the multiple traveling salesman problem. An existing MATLAB implementation given in [41] is adapted for the P-FLMPC multi-objective optimization problem that ensures to optimize the assignment of robots with respect to both exploration efficiency and rescue success rates.

RESULTS FOR CASE STUDY 2

The results for 4 representative simulations of large-scale search and rescue missions (called mission 1, ..., mission 4) are summarized in Table 5.3, giving the completion time for each mission under both the single-level FLMPC system and the bi-level PC-FLMPC system. For all these cases, both controllers finish the mission (i.e., locate all the victims) before the given time budget (i.e., 3000 time steps).

The bi-level PC-FLMPC method demonstrates superior performance in 2 out of the 4

simulations (i.e., mission 1 and mission 3). A significant observation is that the bi-level PC-FLMPC system consistently outperforms the single-level FLMPC system whenever the latter struggles to locate the last victim. This suggests that the coordination offered to the bi-level PC-FLMPC by the P-FLMPC plays a crucial role in eliminating inefficiencies in the later stages of the mission, leading to a more balanced and globally optimized exploration strategy compared to the single-level C-FLMPC system. In fact, the single-level FLMPC system, due to its inability to account for information across the entire environment, tends to adopt a fast and greedy exploration approach, which is not always penalized. In contrast, the bi-level PC-FLMPC system considers the entire environment, leading to a more deliberate and cautious exploration strategy, ensuring that no victims are overlooked. The single-level FLMPC system may locate humans faster in some missions, but this is not guaranteed and mainly depends on how well exploration aligns with the environmental setup.

Table 5.3: Comparison of the completion time of each mission (the values are given in time steps).

	Mission 1	Mission 2	Mission 3	Mission 4
Single-level FLMPC	2869	2059	2688	2222
PC-FLMPC	2016	2435	2447	2664
Improvement due to adding the parent level	29.73%	−18.26%	8.97%	−19.89%

To illustrate this problem, two cases from Table 5.3 are analyzed: Mission 1 and Mission 4, where the bi-level PC-FLMPC system, respectively, wins with the highest advantage and loses with the highest disadvantage.

In mission 1, as shown in Figure 5.8 (top left), the single-level FLMPC system leads to chaotic search behavior due to its greedy exploration. Despite significant time passing, large portions of the environment remain unexplored. Furthermore, by the end of the mission, only one robot continues searching, while the other two robots are trapped within areas that have already been explored. Consequently, when humans are located in the remaining unexplored area, it takes substantial extra time to rescue them. Conversely, the bi-level PC-FLMPC system follows a more structured and coordinated approach (see the bottom left plot in Figure 5.8). This organized search pattern ensures that the locations of victims have less impact on the overall search time.

In mission 4, despite the unfortunate circumstances, the bi-level PC-FLMPC system ensures that most of the environment is explored (see the uncertainty map just before discovering the last victim in the bottom right plot of Figure 5.8). This significantly reduces the impact of random exploration on the worst-case mission time. Moreover, within the remaining unexplored region, two robots are actively searching, ensuring that even in the worst-case mission for the bi-level PC-FLMPC system, the additional time required to complete the mission is only marginally higher than the actual mission time. In contrast, while the

single-level FLMPC system completes the mission faster (see the top right plot in Figure 5.8), it remains highly sensitive to the locations of the victims. In case victims are instead located in the large portions of the environment that remain unexplored, the mission takes considerably longer. This highlights a key drawback for the single-level FLMPC system, i.e., its worst-case mission time is significantly higher than the bi-level parent-child FLMPC system, making its performance less reliable in unpredictable scenarios.

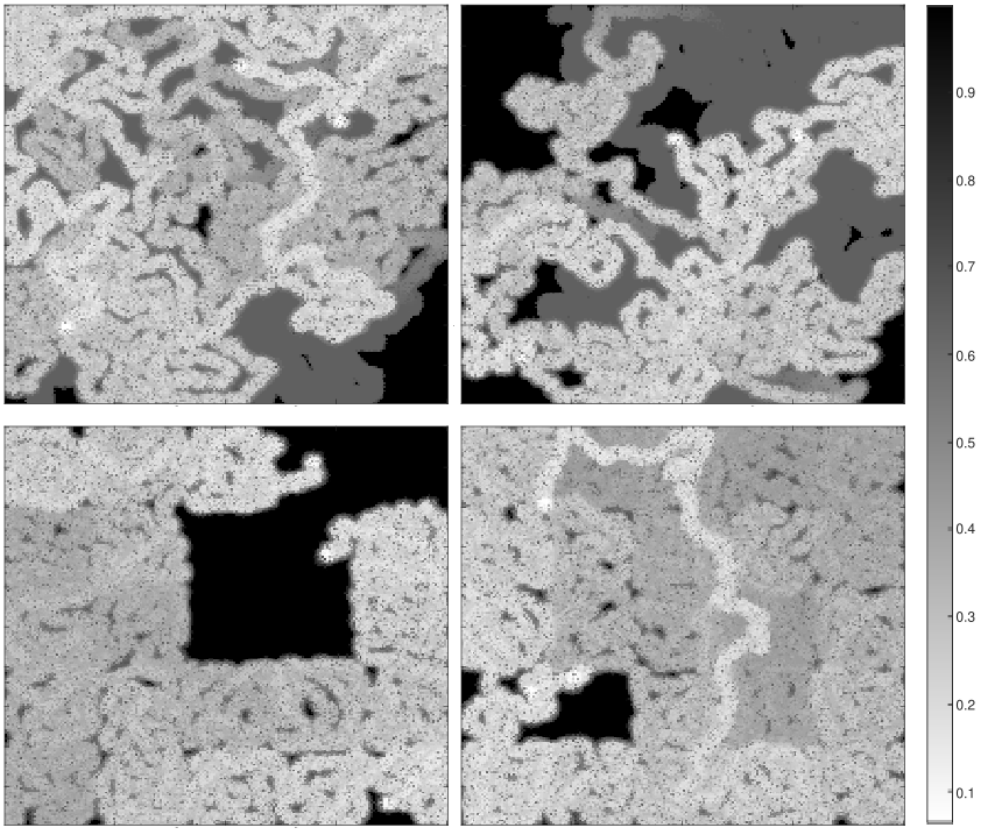


Figure 5.8: End-of-mission uncertainty map for mission 1 under the single-level FLMPC system (top left), mission 4 under the single-level FLMPC system (top right), mission 1 under the bi-level PC-FLMPC system (bottom left), and mission 4 under the bi-level PC-FLMPC system (bottom right). The brightness of the pixels represents the uncertainty.

Remark. In some additional simulations, the bi-level PC-FLMPC system fails to complete the mission within the time budget. This occurred because certain fuzzy clusters are prematurely marked as "explored", with many cells gaining high certainty with a lot of low-detectability (and sometimes incorrect) measurements.

5.5. CONCLUSIONS AND FUTURE WORK

This Chapter introduces a novel control concept based on fuzzy optimization, called FLMPC, and leveraging FLMPC, presents a multi-robot autonomous exploration approach for SaR in unknown environments. While conventional MPC methods commonly employ stochastic cost functions to handle uncertainty in unknown environments, Dynamic fuzzy maps are proposed to systematically process and structure high-level environmental information. Unlike stochastic cost functions, which require extensive sampling and recalculations, by providing a structured way for capturing the uncertainties via fuzzy maps and by incorporating these fuzzy maps into the objective function of FLMPC, the computational efficiency of decision-making and the success rate of multi/single-robot SaR missions are enhanced, while maintaining robust adaptability to dynamic environments. As demonstrated by the results of the first case study, replacing the stochastic cost function not only improves the performance by reducing the mission completion time, but also significantly decreases the computational complexity (by up to 60 times) by pre-processing environmental data before optimization.

To mitigate the inherent limitations of MPC in exploring large-scale unknown environments, i.e., struggling to (effectively) incorporate multi-level environmental information in its optimization-based decision-making, bi-level PC-FLMPC architecture is introduced. The C-FLMPC layer focuses on short-term, reactive decision-making, while the P-FLMPC layer provides long-term strategic guidance for robots. Such a structured coordination ensures a balanced and globally (close-to) optimal exploration, and reduces the impact of randomness on worst-case mission time, resulting in enhanced reliability in SaR missions via robots. The results of the second case study show that bi-level PC-FLMPC ensures a more balanced exploration and reduces randomness and delays in finding all the victims. While single-level FLMPC is in some setups faster, it risks inefficiencies and struggling to find victims in less accessible places, whereas the bi-level PC-FLMPC improves consistency and reliability in mission completion.

Future research on FLMPC should focus on optimizing computational efficiency, improving parameter tuning, and enhancing adaptability to dynamic environments. Developing specialized software for fuzzy optimization and integrating heuristic algorithms will reduce the computation time and enhance the optimality. A structured or data-driven approach to parameter tuning, in the absence of data scarcity and randomness in data, will eliminate reliance on trial and error. Additionally, making the bi-level PC-FLMPC framework more dynamic by refining the procedure of fuzzy cluster formation, enabling real-time cell re-allocation, and adapting the scoring method for the fuzzy clusters will enhance the efficiency and adaptability of bi-level PC-FLMPC. In particular, the limitation discussed in Section 5.4.2, i.e., premature association of the state "explored" to certain fuzzy clusters due to low detectability of high-reward cells, should be addressed to further enhance the robustness and reliability of the bi-level PC-FLMPC system. It is hypothesized that careful tuning of the parameters (e.g., s^{exp} and

w^{score}) will address the issue by compromising the mission speed. This hypothesis should be evaluated via extensive simulations. The impact of a more dynamic approach, considering flexibility in the shape of fuzzy clusters, on improving the area coverage and effectiveness of the bi-level PC-FLMPC in finding all the victims should also be investigated. Finally, implementing the bi-level PC-FLMPC system in realistic SaR operations, including dynamic environments, multiple tasks, and diverse robots, should be researched.

REFERENCES

[1] F. Surma and A. Jamshidnejad. “Fuzzy-logic-based model predictive control: A paradigm integrating optimal and common-sense decision making”. In: *Applied Soft Computing* 193 (2026). DOI: https://doi.org/10.1016/j.asoc.2026.114817.

[2] A. Mesbah. “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research”. In: *IEEE Control Systems Magazine* 36 (2016). DOI: 10.1109/MCS.2016.2602087.

[3] *EM-DAT The international disaster database*. URL: https://public.emdat.be/ (visited on 09/22/2024).

[4] C. Fischer and H. Gellersen. “Location and Navigation Support for Emergency Responders: A Survey”. In: *IEEE Pervasive Computing* 9 (2010). DOI: 10.1109/MPRV.2009.91.

[5] S. F. Ochoa and R. Santos. “Human-centric wireless sensor networks to improve information availability during urban search and rescue activities”. In: *Information Fusion* 22 (2015). DOI: 10.1016/j.inffus.2013.05.009.

[6] A. Bock, Å. Svensson, A. Kleiner, J. Lundberg, and T. Ropinski. “A Visualization-Based Analysis System for Urban Search & Rescue Mission Planning Support”. In: *Computer Graphics Forum* 36 (2017). DOI: 10.1111/cgf.12869.

[7] *Robots come to the rescue after Fukushima Daiichi nuclear disaster*. URL: https://www.cbsnews.com/news/robots-fukushima-daiichi-nuclear-disaster-60-minutes-2021-07-11/ (visited on 09/22/2024).

[8] M. Lyu, Y. Zhao, C. Huang, and H. Huang. “Unmanned Aerial Vehicles for Search and Rescue: A Survey”. In: *Remote Sensing* 15 (2023). DOI: 10.3390/rs15133266.

[9] G. De Cubber, D. Doroftei, Y. Baudoin, D. Serrano, K. Berns, C. Armbrust, K. Chintamani, R. Sabino, S. Ourevitch, and T. Flamma. “Search and Rescue Robots Developed by the European ICARUS Project”. In: *7th International Workshop on Robotics for Risky Environments (RISE 2013) – Extreme*. IEEE, 2013, pp. 1–4. DOI: 10.1109/SSRR.2013.6719323.

[10] J. Whitman, N. Zevallos, M. Travers, and H. Choset. “Snake Robot Urban Search After the 2017 Mexico City Earthquake”. In: *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2018, pp. 1–6. DOI: 10.1109/SSRR.2018.8468633.

[11] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science Robotics* 7 (2022). DOI: 10.1126/scirobotics.abk2822.

[12] C. de Koning and A. Jamshidnejad. “Hierarchical Integration of Model Predictive and Fuzzy Logic Control for Combined Coverage and Target-Oriented Search-and-Rescue via Robots with Imperfect Sensors”. In: *Journal of Intelligent & Robotic Systems* 107 (2023). DOI: 10.1007/s10846-023-01833-2.

[13] M. D. Phung and Q. P. Ha. “Motion-encoded particle swarm optimization for moving target search using UAVs”. In: *Applied Soft Computing* 97 (2020). DOI: 10.1016/j.asoc.2020.106705.

[14] Z. Wang, J. Guo, W. Zou, and S. Li. “Cooperative search for moving targets with the ability to perceive and evade using multiple UAVs”. In: *Intelligence & Robotics* 3 (2023). DOI: 10.20517/ir.2023.30.

[15] R. E. Bellman and L. A. Zadeh. “Decision-Making in a Fuzzy Environment”. In: *Management Science* 17 (1970). DOI: 10.1287/mnsc.17.4.B141.

[16] L. Zadeh. “Fuzzy sets”. In: *Information and Control* 8 (1965). DOI: 10.1016/S0019-9958(65)90241-X.

[17] R. Yager. “On ordered weighted averaging aggregation operators in multicriteria decisionmaking”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 18 (1988). DOI: 10.1109/21.87068.

[18] H.-J. Zimmermann. “Fuzzy programming and linear programming with several objective functions”. In: *Fuzzy Sets and Systems* 1 (1978). DOI: 10.1016/0165-0114(78)90031-3.

[19] J. da Costa Sousa and U. Kaymak. “Model predictive control using fuzzy decision functions”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31 (2001). DOI: 10.1109/3477.907564.

[20] K. Belarbi and F. MEGRI. “A Stable Model-Based Fuzzy Predictive Control Based on Fuzzy Dynamic Programming”. In: *IEEE Transactions on Fuzzy Systems* 15 (2007). DOI: 10.1109/TFUZZ.2006.890656.

[21] S. Teniou and K. Belarbi. “A stabilizing model fuzzy predictive control scheme for nonlinear deterministic systems”. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2016, pp. 724–729. DOI: 10.1109/FUZZ-IEEE.2016.7737759.

[22] Y. Shan, J. Hu, K. W. Chan, and S. Islam. “A Unified Model Predictive Voltage and Current Control for Microgrids With Distributed Fuzzy Cooperative Secondary Control”. In: *IEEE Transactions on Industrial Informatics* 17 (2021). DOI: 10.1109/TII.2021.3063282.

[23] M. Alanezi, H. Bouchekara, T. Apalara, M. S. Shahriar, Y. Shaaban, M. Javaid, and M. Khodja. “Dynamic Target Search Using Multi-UAVs Based on Motion-Encoded Genetic Algorithm With Multiple Parents”. In: *IEEE Access* 10 (2022). DOI: 10.1109/ACCESS.2022.3190395.

[24] H. Zhang, H. Ma, B. W. Mersha, X. Zhang, and Y. Jin. “Distributed cooperative search method for multi-UAV with unstable communications”. In: *Applied Soft Computing* 148 (2023). DOI: 10.1016/j.asoc.2023.110592.

[25] Y. Yang, M. Polycarpou, and A. Minai. “Multi-UAV Cooperative Search Using an Opportunistic Learning Method”. In: *J. Dyn. Sys., Meas., Control.* 129 (2007). DOI: 10.1115/1.2764515.

[26] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. 1st. The MIT Press, 2005. ISBN: 0262201623.

[27] "An improved sand cat swarm optimization for moving target search by UAV". In: *Expert Systems with Applications* 238 (2024), p. 122189. DOI: 10.1016/j.eswa.2023.122189.

[28] P. Trodden and A. Richards. "Multi-Vehicle Cooperative Search Using Distributed Model Predictive Control". In: *AIAA Guidance, Navigation and Control Conference and Exhibit* (2008). DOI: 10.2514/6.2008-7138.

[29] S. Carabaza, E. Besada, J. Lopez-Orozco, and J. de la Cruz. "Ant Colony Optimization for Multi-UAV Minimum Time Search in Uncertain Domains". In: *Applied Soft Computing* 62 (2017). DOI: 10.1016/j.asoc.2017.09.009.

[30] P. Lanillos, J. Yañez-Zuluaga, J. J. Ruz, and E. Besada. "A Bayesian Approach for Constrained Multi-Agent Minimum Time Search in Uncertain Dynamic Domains". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO 2013)*. ACM, 2013, pp. 391–398. DOI: 10.1145/2463372.2463417.

[31] A. Jamshidnejad and B. De Schutter. "A combined probabilistic-fuzzy approach for dynamic modeling of traffic in smart cities: Handling imprecise and uncertain traffic data". In: *Computers and Electrical Engineering* 119 (2024). DOI: 10.1016/j.compeleceng.2024.109552.

[32] K. M. Passino and S. Yurkovich. *Fuzzy Control*. 1st. Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 020118074X.

[33] R. Negenborn, B. De Schutter, and J. Hellendoorn. "Multi-agent model predictive control for transportation networks: Serial versus parallel schemes". In: *Engineering Applications of Artificial Intelligence* 21 (2008). DOI: 10.1016/j.engappai.2007.08.005.

[34] *particleswarm*. URL: www.mathworks.com/help/gads/particleswarm.html (visited on 09/22/2023).

[35] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[36] F. Surma. *Implementation Regarding Publication "Fuzzy-Logic-based model predictive control: A paradigm integrating optimal and common-sense decision making"*. 2025. DOI: 10.4121/319168f0-bc62-4051-84c2-f32718c05386.v1.

[37] *Delft High Performance Computing Centre (DHPC) DelftBlue Supercomputer (Phase 2)*. URL: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2 (visited on 07/19/2024).

[38] S. Nanda Kumar and R. Panneerselvam. "A Survey on the Vehicle Routing Problem and Its Variants". In: *Intelligent Information Management* 04 (2012). DOI: 10.4236/iim.2012.43010.

[39] O. Cheikhrouhou and I. Khoufi. "A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy". In: *Computer Science Review* 40 (2021). DOI: 10.1016/j.cosrev.2021.100369.

[40] A. Király and J. Abonyi. "A Novel Approach to Solve Multiple Traveling Salesmen Problem by Genetic Algorithm". In: *Computational Intelligence in Engineering*. Springer Berlin Heidelberg, 2010, pp. 141–151. ISBN: 978-3-642-15220-7. DOI: 10.1007/978-3-642-15220-7_12.

[41] A. Király. *Multiple Traveling Salesmen Problem - Genetic Algorithm, using multi-chromosome representation*. 2024. URL: https://www.mathworks.com/matlabcentral/fileexchange/48133-multiple-traveling-salesmen-problem-genetic-algorithm-using-multi-chromosome-representation.

APPENDIX A5: MPC WITH STOCHASTIC COST FUNCTION

In Section 5.4.1, to correctly assess the behavior of FLMPC, MPC with a stochastic cost function is implemented, based on the design found in [14]. However, as the problems in the Section 5.4.1 and [14] are different, MPC with a stochastic has to be updated.

As briefly explained in Section 5.4.1, MPC uses a cost function (5.17) that is a weighted summation of four terms as shown by. For the sake of simplicity in notation, the symbol "·" is used as an input of an objective function. However, it should be noted that "·" is a placeholder for the current state of all robots, the trajectory of inputs for all robots, the current probability and certainty maps.

Search effectiveness term (5.17c), and uncertainty term (5.17d) are both preserves. Moreover, human reward (5.17e) and obstacle avoidance term (5.17f) are added. Each of these terms is multiplied by a discount factor $\beta(\cdot)$, which decreases over time in the prediction (as shown by equation (5.17b)).

The probability term (5.17c) depends on the detectability and the prediction of the probability of finding a human, which is estimated by assuming future measurements. In the original work [14], it is assumed that robots would receive information that a cell is empty. However, this assumption becomes problematic when there are multiple possible states for each cell. For this reason, in this implementation, it is assumed that the robot would observe the most likely outcome according to the current belief.

The uncertainty term (5.17d) instead returns higher rewards while passing through less explored cells. $z_c(q)$ is the predicted uncertainty of a cell. To reduce computation, only the uncertainty of cells observed by a robot is increased, without decreasing it for unobserved cells.

A rescued human reward term (5.17e) returns a reward if there is a high probability of rescuing a human, where $g(\cdot)$ is a binary function with 2 arguments (1 vector and 1 integer) that returns 1 if the highest element of the vector has an index equal to the integer. In (5.17e), it is tested whether the state "human" is most likely according to the belief.

Finally, the obstacle term (5.17f) is added to penalize the system where a robot could enter a cell with a high probability of containing an obstacle.

$$J(\cdot) = J_E(\cdot) + 0.5J_H(\cdot) + 2J_R(\cdot) + \frac{J_R(\cdot)}{1500} \tag{5.17a}$$

$$\beta(\kappa) = \exp(\frac{-2\kappa}{L}) \tag{5.17b}$$

$$J_E(\cdot) = \frac{1}{n^{\text{rob}}} \sum_{i=1}^{n^{\text{rob}}} \sum_{q=1}^{n^{\text{path}}} (\beta(q) \prod_{c \in \mathcal{N}(c_{i,q})} (1 - b_{c,2,q} r(||c - c_{i,q}||)) \tag{5.17c}$$

$$J_H(\cdot) = \frac{1}{n^{\text{rob}}} \sum_{i=1}^{n^{\text{rob}}} \sum_{q=1}^{n^{\text{path}}} (\beta(q) \prod_{c \in \mathcal{N}(c_{i,q})} (1 - b_{c,2,q}) z_c(q))) \tag{5.17d}$$

$$J_R(\cdot) = \frac{1}{n^{\text{rob}}} \sum_{i=1}^{n^{\text{rob}}} \sum_{q=1}^{n^{\text{path}}} (\beta(q)(1 - g(\boldsymbol{b}_{c_{i,q},q}, 2) \sqrt{b_{c_{i,q},2,q}}) \tag{5.17e}$$

$$J_O(\cdot) = \frac{1}{n^{\text{rob}}} \sum_{i=1}^{n^{\text{rob}}} \sum_{q=1}^{n^{\text{path}}} (g(\boldsymbol{b}_{c_{i,q},q}, 2) \beta(q) \exp(20 * b_{c_{i,q},3,q})) \tag{5.17f}$$

In the course of the mission, both the MPC with stochastic cost function and the FLMPC employ the same methodology for updating the probability map based on the received measurements. Nevertheless, MPC with stochastic cost employs a distinct methodology for updating uncertainty maps. It is a common practice to employ two models for the evolution of uncertainty [12, 14]. One model is employed for each cell measurement, while the other is utilized when a cell is not measured during the entire timestep. In the case of an unmeasured cell, the update function defined in (5.18) is used to calculate the next uncertainty. This function is predicated on the assumption that the constant e_1 is a positive value below 1. In the implementation, it is set to $e_1 = 0.99$.

$$z_{k+1} = 1 - e_1(1 - z_k) \tag{5.18}$$

If a cell is visited, it is typical to utilize a model of equation (5.19), wherein e_2 represents a positive constant that is less than one. In this implementation, e_2 is not constant and is equal to undetectability, that is, 1 minus detectability, to represent that the robot gains more knowledge about a cell the closer it is to it.

$$z_{k+1} = e_2 z_k \tag{5.19}$$

5

6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This chapter first summarizes the findings based on the results presented in Chapters 2–5 through a comparative analysis. The research questions introduced in Chapter 1 are then revisited and addressed in light of the findings. For each proposed control framework, key limitations are identified and future research directions are outlined. Additionally, potential application domains are proposed for each method.

6.1. COMPARISON OF PROPOSED CONTROLLERS

The focal objective of this thesis is to tackle fundamental challenges in the design of Model Predictive Control (MPC) for control systems operating in dynamic environments that are characterized by unpredictable or initially unknown uncertainties. To this end, the proposed control frameworks were evaluated with respect to three critical dimensions that influence the performance of frameworks and architectures based on MPC: optimality, robustness, and computational efficiency. Table 6.1 presents a comparative overview of the control approaches developed in this thesis. A direct comparison is drawn between each control approach and either the baseline, from which it was derived, or the state-of-the-art control method it was benchmarked against.

Optimality With respect to optimality, the SDD-TMPC approach exhibits optimality levels above those of conventional TMPC, and on par with deterministic MPC, which, however, fails to guarantee robust constraint satisfaction under uncertainty. The ASDD-TMPC method performs slightly below SDD-TMPC, reflecting a trade-off between optimality and its reduced computational cost. The PC-MPC architecture, which is explicitly designed to reduce online computational burden, nevertheless achieves optimality levels comparable to conventional MPC and TMPC in deterministic and non-deterministic scenarios, respectively. This suggests that the PC-MPC architecture effectively balances performance and scalability. Finally, FLMPC delivers

Table 6.1: Comparison of the proposed control methods with respect to the three central criteria considered in this PhD thesis: optimality, robustness, and computational efficiency

Control approach	**Optimality**	**Robustness (constraint satisfaction)**	**Online computational complexity**
SDD-TMPC (Chapter 2)	Higher than conventional TMPC; comparable to deterministic MPC	Guaranteed	Higher than both TMPC and deterministic MPC
ASDD-TMPC (Chapter 3)	Slightly lower than SDD-TMPC	Known probabilistic guarantees	Negligible
PC-MPC (Chapter 4)	Comparable to MPC and TMPC in, respectively, deterministic and non-deterministic settings	Guaranteed	Lower than MPC and TMPC in, respectively, deterministic and non-deterministic settings — particularly in large-scale cases
FLMPC (Chapter 5)	Similar to SMPC	Probabilistic guarantees	Significantly lower than SMPC

optimality levels similar to SMPC, while significantly reducing online computational complexity. This, however, is not the only achievement of FLMPC, as it also handles problems where Robust MPC (RMPC) and SMPC are not applicable because of the nature of or lack of knowledge about uncertainties.

Robustness With regard to constraint satisfaction, both SDD-TMPC and the PC-MPC architecture imply guaranteed robustness, meaning that, under valid modeling assumptions, constraints are guaranteed never to be violated during operation. These guarantees align with traditional robust control principles and are particularly crucial in safety-critical applications. In contrast, ASDD-TMPC provides probabilistic guarantees, where the likelihood of constraint violation can be explicitly quantified. This justifies the "known" qualifier in the table, as the level of constraint violation risk is analytically characterizable based on system uncertainty. FLMPC, although more susceptible to constraint violations, is explicitly designed for environments characterized by unbounded uncertainties, where classical robust constraint satisfaction is fundamentally infeasible. Instead, FLMPC prioritizes computational efficiency and practical feasibility in settings where strict guarantees are unattainable.

Computational efficiency Among the proposed methods, only SDD-TMPC introduces an increased computational burden. This stems from its dynamic adaptation mechanism, which reflects its design objective, formulated in response to *Research Question* 1, that prioritizes optimality and robustness over computational efficiency. In contrast, all other proposed control approaches either maintain comparable computational demands or offer substantial improvements. In particular, ASDD-TMPC achieves negligible computational cost through approximation, while

the PC-MPC architecture leverages hierarchical decomposition to reduce the burden in spatially and temporally large-scale problems.

6.2. ANSWERS TO RESEARCH QUESTIONS

In the introduction chapter of this thesis, three central research questions were formulated to guide the development of the thesis. The results presented across the successive chapters now enable conclusive answers to each of these research questions, as is discussed next.

> *Research Question 1:*
> **To what extent can the boundaries be pushed to enhance optimality and robustness in design of MPC-based systems, assuming computational complexity is not a limiting factor?**

To address this question, SDD-TMPC was developed. TMPC extends conventional MPC by prioritizing robustness to uncertainties, often at the expense of optimality. In turn, SDD-TMPC builds upon TMPC by restoring the lost optimality, while preserving robustness. However, this comes at the cost of increased computational complexity, due to the introduction of a dynamic adaptation mechanism in the control design. Consequently, SDD-TMPC may prove impractical for systems with limited computational resources or with strict real-time requirements.

Nevertheless, this limitation does not render the approach useless. On the contrary, SDD-TMPC has been shown to serve as an optimal foundation for the development of Approximate MPC (AMPC)-based strategies thriving robustness guarantees, thereby yielding ASDD-TMPC. Furthermore, future developments in computing hardware and optimization algorithms are expected to eventually mitigate the current computational challenges that are not limited to SDD-TMPC, but to many advanced MPC-based approaches. A useful analogy is the perceptron [1], a rudimentary Neural Network (NN) model proposed in the 1950s, whose full potential was not realized until several decades later, following major breakthroughs in hardware and algorithm design.

> *Research Question 2:*
> **How can the computational complexity of an MPC-based controller be reduced, depending on the extent to which requirements of the controlled system tolerate compromises in optimality and/or robustness?**

This question was addressed by distinguishing two scenarios: one, in which robustness can be relaxed, and one, where robustness must be preserved.

In the first case, when constraint satisfaction is not critical to control, the ASDD-TMPC approach offers a promising solution. ASDD-TMPC builds on SDD-TMPC by integrating a Spiking Neural Network (SNN), leveraging the fact that SNNs are capable of approximating MPC, due to

their universal function approximation properties, as well as their ability to capture input–output mappings of optimal control laws through supervised learning with affordable computational effort. While SDD-TMPC provides a theoretically sound base, due to its ability to interpret approximation errors as state-dependent disturbances, the incorporation of SNN enables a drastic reduction in online computational effort. ASDD-TMPC is especially effective when high levels of optimality are still desired, but some relaxation in robustness is acceptable. However, this approach requires significant offline computation to generate the training data necessary for the SNN-based approximation.

In the second scenario, where robustness remains critical, the PC-MPC architecture was developed to provide a more suitable solution. The PC-MPC architecture reduces online computational demand through hierarchical decomposition of the optimization problem, while maintaining robust constraint satisfaction. It is capable of extending both conventional MPC and TMPC, and has also shown the flexibility to incorporate more complex formulations, such as FLMPC, with minimal structural modifications.

The PC-MPC architecture is particularly advantageous in large-scale systems, where resource constraints exist, but excessive compromises in robustness or performance are not acceptable — in contrast to ASDD-TMPC, which assumes more tolerance to constraint violations. Moreover, the PC-MPC architecture offers a way to mitigate one of the fundamental challenges of MPC, namely the reliance on a finite prediction horizon. This issue becomes increasingly problematic for systems with large scales or complex dynamics, where it may lead to performance degradation, recursive infeasibility, or even instability.

Research Question 3:
How can uncertainty be effectively modeled when both robust and stochastic representations are inadequate, while still maintaining a balanced trade-off between performance, safety, and computational tractability?

The proposed solution, i.e., FLMPC, leverages Fuzzy Logic (FL). Similarly to stochastic representations, membership functions can be utilized to capture unbounded disturbances. When combined with MPC, this approach enables more efficient optimization compared to stochastic methods. The most computationally intensive step — analyzing the impact of uncertainties on the system state and/or the environmental state across a prediction horizon — is performed prior to the optimization phase, as opposed to being performed within the optimization loop. This is possible due to the nature of FL, which enables modeling a spectrum of uncertainties rather than a specific value for it, enhancing the reliability of the out-of-the-loop predictions. Moreover, unlike stochastic approaches, FLMPC does not require continuous updates of probability distributions during the optimization in order to maintain an accurate representation of the environment. This significantly reduces computational overhead of FLMPC.

However, a key drawback of this approach lies in the difficulty, and in some cases impossibility,

of deriving rigorous mathematical guarantees for FLMPC, such as formal proofs of stability or quantified probabilities of constraint satisfaction. Consequently, unlike other controllers explored in this thesis, the practicality of FLMPC should be validated primarily through implementation and experimentation. This is a known challenge for various FL-based control systems, i.e., even simpler controllers using a single Fuzzy Inference System (FIS) are notoriously difficult to analyze from a theoretical standpoint.

Additionally, the process of tuning FLMPC (and other FL-based controllers) is often time-consuming, and there is currently no established methodology to determine the most effective set of hyper-parameters for such controllers. Consequently, the full potential of FLMPC remains an open avenue for research.

A broader question that arises in the context of this thesis is: *what determines whether a controller is desirable?* Throughout this thesis, the development of new controllers was supported by numerous examples and comparative analyses to shed light on this aspect. The findings of this thesis acknowledge that no such universal definition of a desirable controller exists. In industrial applications, criteria such as the time required for tuning during deployment, ease of adaptation to system expansions, safety (closely tied to robustness), cost-effectiveness, and computational efficiency all weigh as heavily as the classical criteria of optimality and constraint satisfaction. Consequently, rather than prescribing a single framework for characterizing control actions, this thesis offers a range of possible approaches, each tailored to specific problem settings.

6.3. FUTURE RESEARCH

This thesis presented novel contributions in the field of MPC, yet there remains room for further developments. While each chapter concluded with specific research directions relevant to the respective control approach, this section focuses on broader avenues that aim at enhancing multiple controllers simultaneously or on exploring their promising integrations.

Both SDD-TMPC and FLMPC relied on Particle Swarm (PS) [2] to solve their respective optimization problems. Although this algorithm consistently resulted in satisfactory trajectories, performance is expected to improve through the development of specialized solvers tailored to these novel control frameworks. A related challenge affected the training of SNN, since conventional gradient descent methods cannot be directly applied for training, due to the non-differentiability of SNNs. Progress in this field has been achieved now through surrogate gradient descent [3], which has addressed the original non-differentiability issue.

The PC-MPC architecture has the potential to reduce the computational demands of SDD-TMPC. However, the current formulation of this architecture is based on the use of rigid tubes — incompatible with the dynamic mechanism involved in designing the tube for SDD-TMPC. Developing a special extension to the current architecture — analogous to the Parent-Child Fuzzy-Logic-based MPC (PC-FLMPC) architecture introduced in Chapter 5 — will be crucial to handle this challenge.

This thesis primarily evaluated the performance of the developed controllers in simulations under ideal conditions, where all assumptions were satisfied. A key subsequent step is to investigate the behavior of the control strategies in realistic scenarios to identify and address any unforeseen issues that may arise when assumptions break down.

The FL-based models employed by SDD-TMPC were designed, potentially to be tunable using Reinforcement Learning (RL) [4] after initialization with expert knowledge. However, introducing time-varying models require adaptation of the stability proofs currently established for SDD-TMPC. The integration of adaptive control theory [5] offers a promising route to extend the scope of SDD-TMPC to cover such dynamic settings.

Similarly, ASDD-TMPC currently relies on offline training and requires prior knowledge of the environment. Future research should focus on extending ASDD-TMPC to incorporate sensor data for real-time environmental mapping [6], enabling ASDD-TMPC to adapt dynamically. Furthermore, the integration of SNN with fast, energy-efficient neuromorphic sensors [7] represents a promising research avenue.

Finally, FLMPC should be extended to use time-varying fuzzy maps in predictions per control iteration. For instance, a location near a fire may be safe in the short-term, but hazardous over a longer horizon. The utilization of time-varying fuzzy maps within the decision-making loop offers a powerful means of modeling such dynamic environments.

These extensions will collectively enable SDD-TMPC, ASDD-TMPC, and FLMPC to operate effectively in environments that evolve both spatially and temporally.

6.4. POSSIBLE IMPLEMENTATIONS AND IMPACTS

MPC remains a widely used control strategy, provided that a sufficiently accurate model is available and that the computational power allows for a horizon large enough to capture relevant dynamics.

The SDD-TMPC framework, owing to its flexibility, extends conventional TMPC, when prediction errors can be modeled as state-dependent disturbances. This thesis presented three examples of state-dependent disturbances: environmental disturbances (Section 2.4.1), modeling inaccuracies (Section 2.4.2), and disturbances affecting control inputs (Section 3.2.4). Across these cases, SDD-TMPC consistently improved the performance.

Nevertheless, the challenge of high computation time remains unresolved, thereby impeding the feasibility of online implementation. Future advances in computational hardware or algorithmic efficiency can mitigate this issue. In the meantime, as demonstrated in Chapter 3, SDD-TMPC can serve as an offline strategy to generate datasets or as a benchmark to assess the behavior of other controllers and to ascertain whether their performance is (near) optimal.

To date, SNN has been demonstrated in various control applications, including for quadrotors [8], robotic manipulators [9], and autonomous driving [10]. ASDD-TMPC is capable of enhancing any of these systems achieving closer to optimal behavior, although at the expense of investing

time to generate training datasets offline.

The PC-MPC architecture can extend any MPC or TMPC framework designed for systems with bounded nonlinearities, such as quadrotors [11], helicopters [12], and autonomous bicycles [13]. For such systems, the PC-MPC architecture is particularly advantageous due to the large-scale, dynamic environments involved, as well as the requirement for long-term planning. The inherent flexibility of the PC-MPC architecture allows it to be adapted to a wide range of problems, as demonstrated in Chapter 5 through its application in multi-robot systems. Consequently, the PC-MPC architecture holds considerable promise for implementation in real world for numerous systems.

SMPC can be applied to problems where the probability distribution of the uncertainty is known. Its use has been demonstrated in contexts, such as coordination of flying robots [14], microgrid control [15], and path planning [16]. Meanwhile, FLMPC introduces an innovative approach to modeling environmental uncertainties. Although its implementation in Chapter 5 focused on multi-robot systems in unknown environments, FLMPC can be employed in any application where SMPC is viable, as well as in those that due to unboundedness of uncertainties or lack of knowledge about them, neither robust nor stochastic approaches are applicable. Determining application fields and problem types where FLMPC offers the most significant advantages over SMPC and RMPC remains an open research question.

The Search-and-Rescue (SaR) domain exemplifies a context, in which all the proposed controllers can be highly effective. When deploying robots for SaR, multiple controllers often operate at various hierarchical levels. When robots enter hazardous areas, a totally risk-free trajectory may not exist. The robot should then optimize its decisions in the face of unbounded uncertainties. FLMPC can be deployed to generate short-term, near-optimal, relatively safe plans. However, scalability remains a challenge for FLMPC in large environments. This issue can be mitigated by incorporating the PC-MPC architecture and by appropriately configuring the required number of the Parent MPC (P-MPC) components, tailored to the specific problem.

Despite meticulous planning, the success of SaR missions ultimately depends on reliable low-level controllers, which directly interact with the controlled system. SDD-TMPC has demonstrated robustness and accuracy in executing its assigned goals. Nevertheless, given the time-critical nature of SaR operations, the role of SDD-TMPC may be limited to generating datasets for ASDD-TMPC. While ASDD-TMPC lacks guaranteed robustness, its risk can be limited, making its use in certain high-risk tasks justified, particularly when such risks are borne via expendable robots. In such cases, integration with FLMPC, particularly within a PC-MPC architecture, can help to handle and mitigate various risks, including those arising from shortcoming of ASDD-TMPC in robustness.

BIBLIOGRAPHY

[1] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65 (1958). DOI: 10.1037/h0042519.

[2] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[3] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu. “Training Spiking Neural Networks Using Lessons From Deep Learning”. In: *Proceedings of the IEEE* 111 (2023). DOI: 10.1109/JPROC.2023.3308088.

[4] F. Fathinezhad, V. Derhami, and M. Rezaeian. “Supervised fuzzy reinforcement learning for robot navigation”. In: *Applied Soft Computing* 40 (2016). DOI: 10.1016/j.asoc.2015.11.030.

[5] I. D. Landau, R. Lozano, M. MŚaad, and A. Karimi. *Adaptive Control: Algorithms, Analysis and Applications*. 2nd. Springer, 2011. ISBN: 978-0-85729-663-4.

[6] N. S.-Y. Dumont, P. M. Furlong, J. Orchard, and C. Eliasmith. “Exploiting semantic information in a spiking neural SLAM system”. In: *Front Neurosci* 17 (2023). DOI: 10.3389/fnins.2023.1190515.

[7] F. Liao, F. Zhou, and Y. Chai. “Neuromorphic vision sensors: Principle, progress and perspectives”. In: *Journal of Semiconductors* 42 (2021). DOI: 10.1088/1674-4926/42/1/013105.

[8] R. K. Stagsted, A. Vitale, J. Binz, A. Renner, L. Bonde Larsen, and Y. Sandamirskaya. “Towards neuromorphic control: A spiking neural network based PID controller for UAV”. In: *Proceedings of Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, 2020, pp. 1–8. DOI: 10.15607/RSS.2020.XVI.074.

[9] D. Marrero, J. Kern, and C. Urrea. “A Novel Robotic Controller Using Neural Engineering Framework-Based Spiking Neural Networks”. In: *Sensors* 24 (2024). DOI: 10.3390/s24020491.

[10] R. Halaly and E. Ezra Tsur. “Autonomous driving controllers with neuromorphic spiking neural networks”. In: *Front Neurorobot* 17 (2023). DOI: 10.3389/fnbot.2023.1234962.

[11] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone. “Robust online motion planning via contraction theory and convex optimization”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5883–5890. DOI: 10.1109/ICRA.2017.7989693.

[12] G. Bertolani, A. D. Ryals, F. Giulietti, and L. Pollini. “Adaptive Attitude Control of an Unmanned Helicopter”. In: *Proceedings of the Aerospace Europe Conference 2021 (AEC 2021)*. Aerospace Europe /CEAS, 2021, pp. 1–16. URL: https://aec2021.meil.pw.edu.pl/.

[13] H. E. T. Yiqi Gao Andrew Gray and F. Borrelli. “A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles”. In: *Vehicle System Dynamics* 52 (2014). DOI: 10.1080/00423114.2014.902537.

[14] N. Kantas, J. Maciejowski, and A. Lecchini-Visintini. “Sequential Monte Carlo for Model Predictive Control”. In: *Lecture Notes in Control and Information Sciences* 384 (1970). DOI: 10.1007/978-3-642-01094-1_21.

[15] C. A. Hans, P. Sopasakis, A. Bemporad, J. Raisch, and C. Reincke-Collon. “Scenario-based model predictive operation control of islanded microgrids”. In: *54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 3272–3277. DOI: 10.1109/CDC.2015.7402711.

[16] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams. “A Probabilistic Particle-Control Approximation of Chance-Constrained Stochastic Predictive Control”. In: *IEEE Transactions on Robotics* 26 (2010). DOI: 10.1109/TRO.2010.2044948.

[17] F. Surma and A. Jamshidnejad. “Recursive feasibility without terminal constraints via parent–child MPC architecture”. In: *Results in Control and Optimization* 22 (2026). DOI: https://doi.org/10.1016/j.rico.2025.100653.

[18] F. Surma and A. Jamshidnejad. “Fuzzy-logic-based model predictive control: A paradigm integrating optimal and common-sense decision making”. In: *Applied Soft Computing* 193 (2026). DOI: https://doi.org/10.1016/j.asoc.2026.114817.

[19] R. Osypiuk and F. Surma. “Uncertainty of Aircraft Localization with Multilateration and Known Altitude”. In: *Electronics* 14 (2025). DOI: 10.3390/electronics1412242

[20] F. Surma and A. Jamshidnejad. “State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances”. In: *International Journal of Robust and Nonlinear Control* 35 (2025). DOI: 10.1002/rnc.7558.

[21] F. Surma and A. Jamshidnejad. “Approximate SDD-TMPC with Spiking Neural Networks: An Application to Wheeled Robots”. In: *IFAC-PapersOnLine* 58 (2024). DOI: 10.1016/j.ifacol.2024.09.050.

[22] F. Surma, T. P. Kucner, and M. Mansouri. “Multiple Robots Avoid Humans To Get the Jobs Done: An Approach to Human-aware Task Allocation”. In: *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–6. DOI: 10.1109/ECMR50962.2021.9568843.

ACKNOWLEDGEMENTS

To my supervisor, Anahita Jamshidnejad:

I would like to express my sincere gratitude for the guidance, scholarly insight, and unwavering support that have been instrumental throughout the course of this research. Even in moments of disagreement, each discussion deepened my understanding and challenged me to think more critically. Overcoming every challenge along the way has ultimately enabled me to produce a thesis of this quality.

To my parents, Alicja and Jacek Surma:

Thank you for your unwavering emotional support and for being a constant source of willpower throughout this journey. Your wise advice helped me navigate my life. Your care, understanding, and encouragement were essential in helping me maintain my mental well-being and continue moving forward.

To my copromotor, Hans Hellendoorn:

I am grateful for your role on my supervisory committee. I appreciate you taking the time to attend key milestones, ensuring the process stayed on track, and contributing to the final review. Your input helped bring this thesis to completion.

CURRICULUM VITæ

Filip Surma

27-04-1996 Born in Świnoujście, Poland.

EDUCATION

2009–2012 Gimnazjum No. 16 in Szczecin (lower secondary school)

2012–2015 Highschool No. 2 in Szczecin

2015–2019 Undergraduate studies in Automation Systems and Robotics at the West Pomeranian University of Technology in Szczecin

2019–2020 Master of Science in Robotics at the University of Birmingham

WORK EXPERIENCE

2017–2018 Junior Software Developer at Gryftec

2020–2021 Research assistant at Aerobits

2021–2025 PhD candidate

2025– current Robotics Software Engineer and owner of SURM-AI

LIST OF PUBLICATIONS

1. REVIEWED

1. F. Surma and A. Jamshidnejad. “Recursive feasibility without terminal constraints via parent–child MPC architecture”. In: *Results in Control and Optimization* 22 (2026). DOI: https://doi.org/10.1016/j.rico.2025.100653

2. F. Surma and A. Jamshidnejad. “Fuzzy-logic-based model predictive control: A paradigm integrating optimal and common-sense decision making”. In: *Applied Soft Computing* 193 (2026). DOI: https://doi.org/10.1016/j.asoc.2026.114817

3. R. Osypiuk and F. Surma. “Uncertainty of Aircraft Localization with Multilateration and Known Altitude”. In: *Electronics* 14 (2025). DOI: 10.3390/electronics14122420

4. F. Surma and A. Jamshidnejad. “State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances”. In: *International Journal of Robust and Nonlinear Control* 35 (2025). DOI: 10.1002/rnc.7558

5. F. Surma and A. Jamshidnejad. “Approximate SDD-TMPC with Spiking Neural Networks: An Application to Wheeled Robots”. In: *IFAC-PapersOnLine* 58 (2024). DOI: 10.1016/j.ifacol.2024.09.050

6. F. Surma, T. P. Kucner, and M. Mansouri. “Multiple Robots Avoid Humans To Get the Jobs Done: An Approach to Human-aware Task Allocation”. In: *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–6. DOI: 10.1109/ECMR50962.2021.9568843

ADVANCES IN MODEL PREDICTIVE CONTROL UNDER UNCERTAINTY