INTEGRATED MODEL PREDICTIVE AND HUMAN-INSPIRED CONTROL FOR SEARCH-AND-RESCUE ROBOTICS

PERCEPTION, PLANNING, AND MAPPING

MIRKO BAGLIONI

Integrated Model Predictive and Human-Inspired Control for Search-and-Rescue Robotics

Perception, Planning, and Mapping

Integrated Model Predictive and Human-Inspired Control for Search-and-Rescue Robotics

Perception, Planning, and Mapping

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. H. Bijl,
chair of the Board for Doctorates,
to be defended publicly on
Thursday, 16 April 2026, at 10:00 o'clock

by

Mirko Baglioni

This dissertation has been approved by the (co)promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. J. Hellendoorn	Delft University of Technology, *promotor*
Dr. A. Jamshidnejad	Delft University of Technology, *copromotor*
Independent Members:	
Prof. dr. M. A. Neerincx	Delft University of Technology
Prof. dr. M. T. J. Spaan	Delft University of Technology
Prof. dr. G. Cavone	Università Roma Tre, Italy
Prof. dr. I. Papamichail	Polytechneio Kritis
Dr. M. Lazar	Eindhoven University of Technology

Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Keywords: Model Predictive Control, Fuzzy Logic Control, Hierarchical Control, Camera-based Perception, Multi-robot Systems, Search-and-Rescue Robots

Printed by: Ridderprint

Cover by: M. Baglioni

ISBN 978-94-6518-283-4

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

CONTENTS

SUMMARY

This PhD thesis addresses the optimal control and AI-based control of Search-and-Rescue (SaR) robots. The work is motivated by the need to improve the efficiency of SaR operations after disasters, using robots. In fact, the main benefits of using SaR robots are reduced cost, improved speed of response, increased search performance, extended reachability to otherwise inaccessible places, and fewer risks for the SaR crew. Robots can optimize the mission plans and safely explore the environment through systematic mathematical approaches. Therefore, novel control approaches are needed to enable robots to perform SaR operations autonomously and time-efficiently, and this is the main objective of this thesis.

The main contributions of this PhD thesis are the following:

1. We propose novel mission planning frameworks and architectures for ground or flying SaR robots based on Model Predictive Control (MPC), Fuzzy Logic Control (FLC), and other control approaches, in some cases combined to exploit the advantages of multiple methods.

2. We integrate our architectures with models for moving targets and dynamic obstacles, and we leverage robust control formulations to deal with uncertainties and perception approaches to map the SaR environment and track targets.

3. We validate our approaches by comparing them to other state-of-the-art approaches in case studies with simulations and in some cases with real-life experiments in the lab.

Each paper-based chapter addresses an open challenge in the field and provides new contributions, as follows (while Chapters 1 and 8 consist of an introduction and a conclusion, respectively).

Chapter 2: MPC for Combined Target Tracking and Area Coverage in Path Planning [1] MPC has been used in control approaches for SaR robotics, but the applications mainly relate to tracking a reference trajectory. In contrast, optimizing the coverage of the environment area, possesses two main advantages: Facing uncertainties about the known targets location, and finding additional targets in the unexplored areas. Hence, we contribute by designing an MPC approach that determines online, which of the two tasks the robot should address, between target-oriented and coverage-oriented objectives. We adopt a dynamic force-based evacuation model to represent the movements of the target victims, and a robust tube-based MPC formulation, in order to deal with environmental uncertainties included in the victims locations and in the robot model.

Chapter 3: Robust MPC for Motion Planning and Motion Tracking in the Presence of Dynamic Obstacles [2] Another main challenge for autonomous navigation of SaR

robots is avoiding obstacles, that may be dynamic and move according to nonlinear trajectories. To tackle these challenges, we propose a control architecture that firstly leverages an efficient heuristic path planning approach to generate a nominal obstacle-free path towards the targets (i.e., determined assuming there are no uncertainties) based on predicting the potential movements of the obstacles, and secondly, integrates a robust tube-based MPC formulation for an optimal motion tracking system that closely follows the nominal trajectory while guaranteeing robustness to bounded uncontrollable disturbances.

Chapter 4: Camera-based Object Detection Approaches for Target Tracking and Mapping with Flying and Ground Robots Collaboration [3] Since the majority of natural disasters affects low-income countries, it is valuable to make SaR operations more affordable. We tackle this challenge by using color cameras, that have the advantage of having wider applications and affordable prices, and we use camera images taken by flying robots and ground robots, for mapping unknown SaR environments. Firstly, we extend the pose estimation module of the object detection algorithm YOLO (You Only Look Once), in order to estimate the coordinates of the unobserved key points of the body of a victim from an image, and then leverage YOLO to estimate the distance of the victim from the ground robot. Secondly, YOLO is employed in streams captured from a flying and a ground robot, along with a localization algorithm, in order to map the detected points into real-world coordinates, and to fuse them using a Kalman filter, in order to estimate and track the trajectory of a moving victim. Thirdly, an algorithm is proposed for estimating the elevation of the terrain, using YOLO and homography estimation.

Chapter 5: Bi-level Centralized and Decentralized Control Architecture Based on FLC and MPC for Environment Mapping [4] Architectures for SaR multi-robot systems bring important challenges: Centralized systems have high computational complexity and risk of failure; Decentralized systems may perform worse in the presence of interdynamics among robots; Distributed systems require the exchange of information that may be costly or even impossible. Therefore, we integrate in a bi-level structure based on a decentralized FLC layer and a centralized MPC layer that can help in solving these challenges. In addition, this architecture includes global optimality and predictive decision making of MPC and time-efficient, human-inspired decision making of FLC in one control system, in order to increase the performance in mapping the SaR environment, in the presence of fire spread.

Chapter 6: Hierarchical and Optimal Control of Multi-Unmanned Aerial Vehicle (UAV) Systems Achieving Best Viewpoints for Wildfire Suppression [5] Flying robots (or UAVs) are promising for suppression of wildfires, within SaR missions, but their autonomous control and coordination pose scientific challenges: These UAVs are required to perform multiple tasks (navigating, providing the best views of the fire extinguishing UAVs, monitoring fires and other robots, avoiding collisions). Therefore, we employ Task Hierarchical Control (THC) to perform a parallel execution of multiple tasks with different priorities, Path Integral Control (PIC), for tasks that demand optimality, and the state-of-the-art best viewpoints method, to determine the optimal viewpoints of the fire extinguishing UAVs. In addition, we achieve an occlusion-free Field of View (FoV) of auxiliary UAVs, with the FoV centered on the fire extinguishing UAV, by introducing a novel line-of-sight obstacle avoidance method.

Chapter 7: FLC and Explainable AI for Human-in-the-loop Decision Making in Firefighting Scenarios In firefighting scenarios, robots are usually teleoperated. Therefore, on the one hand, there is a need for automatic control approaches to increase their autonomy, in order to reduce the workload for the human user. On the other hand, firefighters need both to maintain moral responsibility and to be aware of the environmental information. Hence, in order to combine these two requirements, we design a control architecture where both the robot and the human stay in the loop, by combining FLC, to determine temporary navigation destinations for the robot, and Explainable Artificial Intelligence (XAI), to support human-in-the-loop decision making with communication between human and robot where the human selects the preferred target destination using a graphical user interface.

SAMENVATTING

Dit proefschrift gaat over de optimale besturing en AI-gebaseerde besturing van Search-and-Rescue (SaR) robots. Het werk wordt gemotiveerd door de noodzaak om de efficiëntie van SaR operaties na rampen te verbeteren, met behulp van robots. De belangrijkste voordelen van het gebruik van SaR robots zijn lagere kosten, verbeterde reactiesnelheid, betere zoekprestaties, grotere bereikbaarheid naar anders ontoegankelijke plaatsen en minder risico's voor de SaR bemanning. Robots kunnen de missieplannen optimaliseren en de omgeving veilig verkennen door middel van systematische wiskundige benaderingen. Daarom zijn er nieuwe besturingsbenaderingen nodig om robots in staat te stellen autonoom en tijdbesparend SaR-bewerkingen uit te voeren, en dit is het hoofddoel van dit proefschrift.

De belangrijkste bijdragen van dit proefschrift zijn de volgende:

1. We stellen nieuwe missieplanningskaders en architecturen voor grond of vliegende SaR-robots op basis van Model Predictive Control (MPC), Fuzzy Logic Control (FLC) en andere besturingsbenaderingen, in sommige gevallen gecombineerd om de voordelen van meerdere benaderingen te benutten.

2. We integreren onze architecturen met modellen voor bewegende doelen en dynamische obstakels, en we maken gebruik van robuuste controleformuleringen om met onzekerheden en perceptiebenaderingen om de SaR-omgeving in kaart te brengen en doelen te volgen.

3. We valideren onze benaderingen door ze te vergelijken met andere state-of-the-art benaderingen in casestudy's met simulaties en, in sommige gevallen, met real-life experimenten in het lab.

Elk op een artikel gebaseerd hoofdstuk behandelt een open uitdaging in het veld en levert nieuwe bijdragen, als volgt (terwijl hoofdstukken 1 en 8 respectievelijk uit een inleiding en een conclusie bestaan).

Hoofdstuk 2: MPC voor het Combineren van het Volgen van Doelen en Gebiedsdekking in Padplanning [1] MPC is gebruikt in controlebenaderingen voor SaR robotica, maar de toepassingen hebben voornamelijk betrekking op het volgen van een referentietraject. Aan de andere kant heeft het optimaliseren van de dekking van het milieugebied twee belangrijke voordelen: het onder ogen zien van onzekerheden over de locatie van de bekende doelen en het vinden van extra doelen in de onontgonnen gebieden. Daarom dragen we bij door een MPC-aanpak te ontwerpen die online bepaalt welke van de twee taken de robot moet uitvoeren, tussen doelgerichte en dekkingsgerichte doelstellingen. We gebruiken een dynamisch, op kracht gebaseerd evacuatiemodel om de bewegingen van de doelslachtoffers weer te geven, en een robuuste op tubes gebaseerde

MPC-formulering, om om te gaan met milieuonzekerheden die zijn opgenomen in de locaties van de slachtoffers en in het robotmodel.

Hoofdstuk 3: Robuuste MPC voor Bewegingsplanning en het Volgen van Bewegingen in de Aanwezigheid van Dynamische Obstakels [2] Een andere belangrijke uitdaging voor autonome navigatie van SaR robots is het vermijden van obstakels, die dynamisch kunnen zijn en volgens niet-lineaire trajecten kunnen bewegen. Om deze uitdagingen aan te gaan, stellen we een controlearchitectuur voor die ten eerste gebruikmaakt van een efficiënte heuristische padplanningsbenadering om een nominaal obstakelvrij pad naar de doelen te genereren (d.w.z. bepaald ervan uitgaande dat er geen onzekerheden zijn) op basis van het voorspellen van de mogelijke bewegingen van de obstakels, en ten tweede integreert een buisgebaseerde MPC-formulering voor een optimaal bewegingsvolgsysteem dat het nominale traject nauwlettend volgt en tegelijkertijd de robuustheid tot begrensd garandeert oncontroleerbare verstoringen.

Hoofdstuk 4: Cameragebaseerde Objectdetectiebenaderingen voor het Volgen en in Kaart Brengen van Doelen met Samenwerking tussen Vliegende en Grondrobots [3] Aangezien de meeste natuurrampen lage-inkomenslanden treffen, is het waardevol om SaR-operaties betaalbaarder te maken. We pakken deze uitdaging aan door kleurencamera's te gebruiken, die het voordeel hebben dat ze bredere toepassingen en betaalbare prijzen hebben, en we gebruiken camerabeelden die zijn gemaakt door vliegende robots en grondrobots, voor het in kaart brengen van onbekende SaR omgevingen. Eerst breiden we de pose-schattingsmodule van het objectdetectie-algoritme YOLO (You Only Look Once) uit, om de coördinaten van de niet-waargenomen sleutelpunten van het lichaam van een slachtoffer uit een afbeelding te schatten, en vervolgens YOLO te gebruiken om de afstand van het slachtoffer tot de grondrobot te schatten. Ten tweede wordt YOLO gebruikt in stromen die zijn vastgelegd door een vliegende en een grondrobot, samen met een lokalisatie-algoritme, om de gedetecteerde punten in kaart te brengen in real-world coördinaten, en om ze samen te smelten met behulp van een Kalman-filter, om de baan van een bewegend slachtoffer te schatten en te volgen. Ten derde wordt een algoritme voorgesteld voor het schatten van de hoogte van het terrein, met behulp van YOLO en homografieschatting.

Hoofdstuk 5: Gecentraliseerde en Gedecentraliseerde Besturingsarchitectuur op Twee Niveaus op Basis van FLC en MPC voor het in Kaart Brengen van de Omgeving [4] Architecturen voor SaR multi-robotsystemen brengen belangrijke uitdagingen met zich mee: Gecentraliseerde systemen hebben een hoge rekencomplexiteit en het risico op mislukking; Gedecentraliseerde systemen kunnen slechter presteren in de aanwezigheid van interdynamiek tussen robots; Gedistribueerde systemen vereisen de uitwisseling van informatie die kostbaar of zelfs onmogelijk kan zijn. Daarom integreren we in een structuur op twee niveaus op basis van een gedecentraliseerde FLC-laag en een gecentraliseerde MPC-laag die kan helpen bij het oplossen van deze uitdagingen. Bovendien omvat deze architectuur wereldwijde optimaliteit en voorspellende besluitvorming van MPC en tijdefficiënte, door mensen geïnspireerde besluitvorming van FLC in één besturingssysteem, om de prestaties te verbeteren bij het in kaart brengen van de SaR-omgeving, in de aanwezigheid van brandverspreiding.

Hoofdstuk 6: Hiërarchische en Optimale Controle van Multi-Unmanned Aerial Vehicle (UAV) Systemen het Bereiken van de Beste Gezichtspunten voor het Onderdruk-

ken van Natuurbranden [5] Vliegende robots (of UAVs) zijn veelbelovend voor het onderdrukken van bosbranden, binnen SaR missies, maar hun autonome controle en coördinatie vormen wetenschappelijke uitdagingen: Deze UAVs zijn nodig om meerdere taken uit te voeren (navigeren, het beste zicht geven op de brand, UAVs blussen, branden en andere robots monitoren, botsingen vermijden). Daarom gebruiken we Task Hierarchical Control (THC) om een parallelle uitvoering van meerdere taken met verschillende prioriteiten uit te voeren, Path Integral Control (PIC), voor taken die optimaliteit vereisen, en de state-of-the-art best viewpoints-methode, om de optimale gezichtspunten van de blusser te bepalen UAVs. Bovendien bereiken we een occlusievrije Field of View (FoV) van hulpUAVs, waarbij de FoV gecentreerd is op de brandblussing UAV, door een nieuwe methode voor het vermijden van obstakels in het zicht te introduceren.

Hoofdstuk 7: FLC en Verklaarbare AI voor Human-in-the-loop Besluitvorming in Brandbestrijdingsscenario's In brandbestrijdingsscenario's worden robots meestal op afstand bediend. Daarom is er enerzijds behoefte aan automatische controlebenaderingen om hun autonomie te vergroten, om de werklast voor de menselijke gebruiker te verminderen. Aan de andere kant moeten brandweerlieden zowel morele verantwoordelijkheid behouden als zich bewust zijn van de milieu-informatie. Om deze twee vereisten te combineren, ontwerpen we daarom een besturingsarchitectuur waarbij zowel de robot als de mens op de hoogte blijven, door FLC te combineren, om tijdelijke navigatiebestemmingen voor de robot te bepalen, en Explainable Artificial Intelligence (XAI), om human-in-the-loop besluitvorming te ondersteunen met communicatie tussen mens en robot waarbij de mens de gewenste doelbestemming selecteert met behulp van een grafische gebruikersinterface.

List of Acronyms

AI Artificial Intelligence

APF Artificial Potential Function

ASR Air-Sea Rescue

BIBO Bounded-Input-Bounded-Output

CNN Convolutional Neural Network

FIS Fuzzy Inference System

FL Fuzzy Logic

FLC Fuzzy Logic Control

FoV Field of View

GPS Global Positioning System

HL-RRT* Horizon-based Lazy Rapidly-exploring Random Tree

HP+TMPC integrated Heuristic motion Planning and robust TMPC

IMU Inertial Measurement Unit

LiDAR Light Detection And Ranging

LoS Line-of-Sight

M2PFC Multi-agent Model-Predictive Fuzzy logic Control

MPC Model Predictive Control

PIC Path Integral Control

PID Proportional-Integral-Derivative

robust TMPC Robust Tube-based Model Predictive Control

RRT* Rapidly exploring Random Tree

SaR Search-and-Rescue

SLAM Simultaneous Localization And Mapping

THC Task Hierarchical Control

TSK Takagi-Sugeno-Kang

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

USaR Urban Search-and-Rescue

USV Unmanned Surface Vehicle

UUV Unmanned Underwater Vehicle

XAI Explainable Artificial Intelligence

YOLO You Only Look Once

1

INTRODUCTION

This chapter presents the motivation for advancing autonomous control in Search-and-Rescue (SaR) robotics, highlighting the critical role of intelligent systems in improving the effectiveness of disaster response operations. The chapter outlines the structure of the thesis and introduces the core methodologies applied or leveraged throughout the thesis, encompassing both theoretical developments and their application to SaR scenarios. In addition, a comprehensive discussion of the key open challenges that continue to hinder fully autonomous SaR operations is discussed. Finally, the chapter summarizes the main scientific contributions of the thesis and their significance within the broader landscape of robotic control and disaster management.

1

MOTIVATION

NATURAL and man-made disasters continue to claim countless lives and cause significant disruption to society and the environment [6], [7]. A defining characteristic of such disasters (whether earthquakes, hurricanes, floods, fires, or pandemics) is their unpredictability and unavoidability. While prevention is often impossible, the impact of such disasters can be mitigated through effective response strategies. Therefore, there is a persisting demand for more efficient disaster management protocols.

One critical aspect of disaster response is *Search-and-Rescue (SaR)*, which involves a range of coordinated tasks, most importantly, saving lives of victims and mitigating further harm [8]. In other words, locating and rescuing victims remains paramount [9].

However, real-world events have exposed various limitations in current SaR practices, particularly in initiating timely, wide-reaching, and safe operations [10], [11]. In order to address these challenges, SaR operations are increasingly integrating robotic systems [12], [13].

Robots contribute across various tasks, including mapping unknown terrain, locating trapped victims, and traversing and scanning hazardous areas that are inaccessible to humans [14], [15]. Their sensors enable rapid and autonomous environmental assessment, while their expendability and capability of navigating dangerous areas improve safety and operational reach. Furthermore, delegating high-risk tasks to robots allows human responders to focus on critical efforts, such as providing medical care and psychological support for victims [16], [17].

Key benefits of deploying robots in SaR include faster response, improved search performance, increased coverage of zones that are inaccessible or difficult to reach, enhanced safety for the SaR personnel, and reduced cost [18].

While human rescuers rely on intuition and experience to deal with uncertain SaR situations, robots are capable of employing formal mathematical methods to optimize their mission plans and to make safe and timely decisions in exploring the SaR environment. Therefore, advanced control approaches are essential to enable autonomous, time-efficient robotic behavior during SaR operations [19].

Optimal control techniques, such as Model Predictive Control (MPC) [20] and Path Integral Control (PIC) [21], offer systematic frameworks to balance competing objectives, such as minimizing the mission time, maximizing the area coverage, conserving energy, and avoiding hazards.

In parallel, heuristic control techniques, such as Fuzzy Logic Control (FLC) [22] and other strategies based on Artificial Intelligence (AI), can be employed to mimic human-like decision making for SaR robots. These methods are especially useful for complex or nonlinear systems and are computationally efficient, making them well-suited for real-time applications in unstructured SaR environments.

Post-disaster environments are often reshaped by structural damage and collapse, and by dynamic hazards. Thus, robust and adaptive control methods are crucial for ensuring autonomous, safe navigation in dynamic, unpredictable, and partially known environments [23]–[25]. Robust control techniques, such as Robust Tube-based Model Predictive Control (robust TMPC) [26], allow SaR robots to mitigate uncertainties arising from dynamic changes (e.g., spreading fire or moving obstacles), model mismatches, and external disturbances (e.g., uneven terrain or debris) in SaR environments.

Besides control, perception systems play a vital role in SaR robotics. These systems enable robots to interpret environmental features (e.g., hazards, victims, obstacles) through sensor data, and to inform their decisions about which areas to explore or which targets to prioritize [27].

PROBLEM SETTINGS

This thesis addresses SaR scenarios that are both indoor (partially collapsed buildings in urban contexts) and outdoor (wilderness contexts with fires). Robots, either ground or flying, are used to support and improve the SaR operations. Due to the disasters, there are victims in the SaR scenarios, that the robots should detect, reach and track when appropriate. Static obstacles, and, in some cases, dynamic obstacles are represented by debris or collapsed walls in indoor scenarios, and trees or spreading fire in outdoor scenarios. The robots should avoid collisions with such obstacles, in addition to other robots and target victims (that may be dynamic). Moreover, the robots are equipped with various sensors to gather information from the SaR environment, such as to detect victims or obstacles, determine the distance from them, and in general map the environment. Uncertainties about the elements of the environment (moving victims, dynamic obstacles, spreading fire) are present in these contexts and should be taken into account by the control systems. These control systems are based on various control techniques (optimal, heuristic, model-based, AI-based, robust, hierarchical), depending on the specific scenario considered in each chapter of this thesis, and are used to formulate new mission planning, path planning, exploration, and navigation approaches, and to control and coordinate multi-robots systems. In addition, control and AI-based techniques are used for perception, tracking, mapping, sensor fusion and human-robot interaction. Therefore, the aim of this thesis is to formulate, analyze, and assess through simulations and experiments the proposed control systems and perception approaches.

RESEARCH OBJECTIVES AND RESEARCH QUESTIONS

The overarching objective of the research presented in this thesis is:

To develop and validate novel mission planning frameworks and architectures based on model-based, optimal, and human-inspired control approaches, as well as efficient and reliable vision-based perception and sensor fusion techniques, in order to enhance the autonomy, robustness, computational efficiency, and multi-robot coordination of SaR robots that operate in uncertain dynamic environments, while supporting human-in-the-loop decision making where appropriate.

Building on this objective, the following research questions (RQs) are addressed:

RQ1. How can optimal control approaches, particularly, MPC, be extended to enable mission planning strategies that balance the competing objectives of target tracking and area coverage in uncertain SaR environments?

RQ2. How can optimal and human-inspired control methods be generalized to handle dynamic environments involving moving targets, uncertain obstacle behav-

ior, and unmodeled disturbances, while providing guarantees of robustness and safety?

RQ3. How can vision-based perception be advanced and fused across heterogeneous ground and aerial robots to enable scalable, cooperative mapping and target tracking within SaR environments under limited sensing and communication constraints?

RQ4. How can optimal and human-inspired control approaches be systematically integrated into hybrid architectures to exploit their complementary strengths, ensuring both real-time responsiveness and long-term optimality in SaR missions?

RQ5. How can the proposed control and perception approaches be validated through rigorous simulation and real-world experiments, and how can their performance be benchmarked against state-of-the-art approaches under realistic SaR scenarios?

RQ6. How can explainable and trustworthy human-in-the-loop decision making be integrated with autonomous SaR robotics to balance human oversight and robot autonomy in time-critical missions?

OUTLINE

Table 1.1 provides a high-level overview of the thesis structure, organized by topic. The primary focus of this thesis is autonomous control of SaR robots. Within this domain, several key topics are explored across the chapters, categorized along six main dimensions, as illustrated in Table 1.1:

- Control methodology;
- Control architecture;
- Robotic task;
- Robot type;
- System scale;
- SaR environment.

These aspects are addressed throughout Chapters 2 to 7.

After an extensive investigation of fully autonomous robotic control, Chapter 7 introduces the concept of human-in-the-loop control, emphasizing human-robot coordination and collaboration without compromising robotic autonomy.

Another important dimension covered by the thesis is robotic perception, with a focus on video-based sensing. The proposed techniques support tasks, such as object detection and best viewpoints achievement, which are explored in Chapters 4 and 6, respectively.

Chapter #	Autonomous robotic control																				Human-in-the-loop robotic control
	Control																		Perception		
	Approach					Control architecture		Task					Robot type		System scale		SaR environment		Video-based perception		
	Conventional MPC	Robust MPC	Fuzzy logic control	Integrated FLC and MPC	Task hierarchical control and Path integral control	Single-level	Multi-level (hierarchical)	Path planning	Mission planning	Target tracking	Exploration	Mapping	Unmanned ground vehicle	Unmanned aerial vehicle	Single-agent	Multi-agent	Dynamic environments	Dynamic targets	Object detection	Best viewpoints	
Chapter 2	X					X		X					X		X			X			
Chapter 3		X					X		X				X		X		X				
Chapter 4										X		X	X	X		X		X	X		
Chapter 5				X			X		X		X	X		X		X	X				
Chapter 6					X		X	X		X				X		X		X		X	
Chapter 7			X			X					X		X		X						X

Table 1.1: High-level overview of the thesis structure by topic.

1

FUNDING ACKNOWLEDGEMENTS

This research has been supported jointly by the TU Delft AI Labs & Talent programme - as a part of the AI*MAN lab research - and by the NWO Talent Program Veni project "Autonomous drones flocking for search-and-rescue" (18120), which has been financed by the Netherlands Organization for Scientific Research (NWO). University of Texas at Austin has collaborated in the work presented in Chapter 6 through the project within the XPRIZE Wildfire competition.

BACKGROUND

This section is structured into two main parts. First, it provides an overview of key fundamental methodologies relevant to the research presented in this thesis. The second part discusses the main state-of-the-art approaches in control and perception for SaR robotics.

OVERVIEW OF THE MAIN METHODOLOGIES

This section presents a discussion on core methodologies underlying the fundamental research topics of the thesis, particularly for robotic control and perception.

Robotic Control: The following sections introduce methods for robotic control, with a particular focus on exploration, environmental mapping, and path planning.

Model Predictive Control (MPC): MPC is an advanced control method that computes optimal control inputs based on a predictive model of the controlled system [20]. At each control time step, MPC solves an optimization problem to minimize (or maximize) a cost (or objective) function subject to various constraints.

MPC is widely used across a large number of applications and has several key variants, including *nonlinear MPC*, *robust MPC*, *stochastic MPC*, and *distributed MPC*. The underlying model of the controlled system may be linear or nonlinear, time-varying or time-invariant, continuous-time or discrete-time, and centralized or distributed.

Constraints in the optimization problem may be imposed on the inputs, states, or outputs, and can be either hard (i.e., strictly enforced) or soft (i.e., relaxed, allowing limited violations to prevent infeasibility or performance degradation). A specific class of constraints, primarily used in the variant known as stochastic MPC, is probabilistic in nature — commonly referred to as *chance constraints*.

The cost function may reflect multiple, potentially competing objectives, such as time efficiency, energy conservation, and risk avoidance. Moreover, it is typically represented as a combination of stage and terminal costs. These allow the controller to handle trade-offs systematically.

MPC has been largely employed in several industrial processes, for power electronics, energy, or manufacturing, and for controlling engines, vehicles, airplanes, spacecrafts or missiles, as examples [28]. In the context of SaR robotics, it is used to optimize the actions of the robots, such as in path planning [29], mostly to track trajectories and avoid obstacles, or for multi-robot coordination [30].

MPC is used in the control approaches formulated in Chapters 2, 3 and 5.

Robust Tube-based Model Predictive Control (Robust TMPC): A particularly relevant variant of MPC for uncertain systems or environments is robust MPC, which maintains control performance in the presence of system uncertainties [31]. While stochastic MPC handles uncertainties using a probabilistic formulation and chance constraints [32], robust MPC deals with uncertainties deterministically.

Such uncertainties may stem from model inaccuracies [33], external disturbances [34], or noisy state measurements [35]. Issues originating from uncertainties are especially prominent in SaR missions, where robust control methods are essential [36], [37].

Among the most widely used robust MPC formulations, there are *min-max MPC* and *robust TMPC* [26], [38]. In min-max MPC, the controller optimizes for the worst-case realization of uncertainty. In contrast, robust TMPC consists of a nominal MPC controller and an ancillary feedback controller that keeps the actual state trajectory of the system within a bounded region (called the *tube*) of the admissible state space, around the nominal state trajectory.

Compared to min-max MPC, robust TMPC is computationally less demanding, particularly because it does not solve a problem for all admissible states, which in min-max MPC leads to a complexity that increases exponentially with the prediction horizon, while in robust TMPC it increases rather quadratically [39]. It is also less conservative, since it does not ignore information on the uncertainty realization as min-max MPC does by considering its maximum [40]. In SaR contexts, robust TMPC has demonstrated strong performance, as it provides robustness to bounded uncertainties, enables balanced trade-offs among competing goals of SaR missions (such as maximizing the area coverage and minimizing the mission time), enforces state and input constraints systematically, and offers real-time stability guarantees [25], [41], [42].

Robust TMPC is used in the control approaches formulated in Chapters 2 and 3.

Fuzzy Logic Control (FLC) and AI-based Decision Making: FLC is a heuristic control technique that incorporates human knowledge into control systems, typically in the form of natural language rules translated into quantified values [22]. Due to its heuristic-based nature, FLC can also contribute to the growing field of Explainable Artificial Intelligence (XAI), i.e., a branch of AI that includes methods that enable AI algorithms to produce output and results that are understandable, interpretable, and reliable for human users [43].

The process begins with *fuzzification*, which maps crisp measurements into fuzzy sets using membership functions. These fuzzy values are then processed through a *Fuzzy Inference System (FIS)* that applies control rules, deploying fuzzy operations, to generate fuzzy outputs (representing the control decisions in FLC systems). Finally, a *defuzzification* step converts the fuzzy outputs back into crisp values used to actuate the controlled system [44].

A key advantage of FLC is its ability to deal with highly nonlinear systems. By leveraging expert-defined linguistic rules, FLC eliminates the need for precise mathematical models of the controlled system or any data-driven training. This results in computationally efficient decision making and makes FLC a transparent and interpretable approach [45].

For these reasons, FLC is particularly effective in SaR scenarios, where it has been

successfully deployed to control robots in uncertain, unstructured environments [46]–[49]. FLC is used in the control architectures of Chapters 5 and 7. In addition to classical FLC, hybrid approaches have emerged that combine fuzzy logic with other techniques. This includes:

- Neuro-fuzzy controllers, which hybridize fuzzy logic with neural networks to incorporate learning capabilities;[50]
- Hybrid FLC and genetic algorithms, enabling optimization of fuzzy rules and membership functions [51], [52];
- Hierarchical architectures fusing FLC with MPC, particularly in multi-agent systems, to introduce predictability and to facilitate coordinated control [25].

Beyond FLC, other intelligent control approaches are increasingly employed for autonomous control of robots, including in SaR. These approaches include:

- Behavior-based (reactive) AI [53], which decomposes the behavior of robots into modular, reactive rules that respond directly to sensor inputs, so that the robot exhibits rather complex behavior despite little internal model of the environment, by correcting and adapting its actions. In particular for SaR scenarios, this allows for fast, adaptive responses in dynamic environments without requiring a full environmental model.
- Reinforcement learning and deep (reinforcement) learning [54], [55], which allow robots to learn optimal policies through interacting with their environment. An important advantage of these techniques is that they do not require prior knowledge of the robot dynamics, but the controller is instead learned from interaction experience, and therefore, it can implicitly deal with uncertainties and disturbances. In SaR missions, such methods are used for autonomous decision making, e.g., for adaptive exploration strategies, dynamic path planning, and coordination of multi-robot systems under uncertainty.
- Evolutionary algorithms, such as ant colony optimization [56] and particle swarm optimization [57], which are inspired by natural processes and are deployed in SaR missions for multi-robot path planning and coverage tasks. Evolutionary algorithms are especially useful in high-dimensional, dynamic, or partially known environments, where traditional optimization methods may struggle. This happens in particular for multi-robot systems, where the robots are treated as agents in the swarm or colony.
- Probabilistic techniques, such as particle filtering (also known as Monte-Carlo methods) [58] and information-based control [58], which handle uncertainties in robot localization, interpretation of sensor data, and decision making. They are based on maximizing the mutual information with respect to a target, rather than minimizing a distance or maximizing the covered area. In SaR scenarios, probabilistic techniques support tracking victims and mapping the environment by incorporating noisy, partial, or ambiguous data streams over time.

Multi-agent and Hierarchical Control Methods: Multi-agent systems are typically governed using one of the following control architectures: *Centralized*, *decentralized*, or *distributed* [59].

In centralized control, a single controller has global knowledge and computes actions for all agents. This control architecture can achieve globally optimal solutions, but is computationally expensive and susceptible to single-point failure [60].

In decentralized control architectures, each agent operates independently without exchanging information. This is computationally lighter than centralized control and robust to communication failures, but may lead to sub-optimal performance, when agents are dynamically inter-dependent [61].

Distributed control lies between centralized and decentralized control architectures: agents share local information to solve local control problems in coordination. This enables scalable and fault-tolerant control systems, albeit with increased algorithmic complexity and communication overhead [62].

An advanced class of architectures is *hierarchical* control [63], where control is structured across multiple layers, e.g., distributed control at the lowest, local level and centralized supervision at the highest, global level [25], [64], [65]. Interactions across layers of hierarchical architectures can be unilateral or bilateral.

Task Hierarchical Control (THC) [66] introduces hierarchy in the task space, allowing to execute multiple objectives via various tasks, with explicit priority handling. A relevant example of using THC in the context of multi-agent systems is the deployment of multiple Unmanned Aerial Vehicles (UAVs) for wildfire suppression tasks [67]. Based on task distribution, these UAVs may be categorized as follows:

- *Fire extinguishing UAVs* [68], which tele-operatedly or semi-autonomously navigate to fire locations to suppress the fire using water or retardants;
- *Fire detecting UAVs* [69], [70], which detect and report fire outbreaks using onboard sensors;
- *Auxiliary UAVs*, as introduced in this thesis (Chapter 6), should autonomously and closely accompany fire extinguishing UAV to maintain clear, obstacle-free viewpoints, thereby enhancing situational awareness of human operators. These UAVs typically operate in coordinated swarms.

Robotic Perception: The next sections discuss methods for robotic perception, focusing on interpreting sensor data through algorithms for object detection, state estimation, and visual awareness of the environment.

Convolutional Neural Networks (CNNs): CNNs [71] are deep learning algorithms that play a dominant role in computer-vision-based perception. In the context of SaR operations, the most relevant tasks that require computer-vision-based perception include object detection, object recognition, and semantic segmentation, all being crucial for interpreting data from cameras to distinguish hazards, targets, and key environmental features [72].

CNNs are preferred over classical machine learning and other deep learning approaches, due to their ability to autonomously extract hierarchical features, as well as their inherent translation invariance and computational efficiency [73].

Beyond robotics, CNNs are employed in a wide range of applications, such as for face recognition, autonomous driving, self-service supermarkets, and intelligent healthcare solutions [74].

YOLO (You Only Look Once) for Object Detection in Images: Among state-of-the-art object detection algorithms, YOLO [27] stands out as one of the most promising real-time object detection frameworks. It has evolved through multiple versions, each introducing significant improvements in terms of accuracy and speed of detection [75].

Built on CNNs, YOLO performs object detection from an image in a single forward pass by integrating feature extraction, candidate boxes extraction and object classification methods into one neural network. Therefore, it determines bounding boxes and probabilities for associated classes of an object, enabling fast inference.

YOLO has been successfully employed across various domains, including robotics, where fast and reliable object detection is crucial [76], [77]. In the context of SaR robotics, YOLO enables reliable detection of victims, obstacles, and other key environmental features from sensor-generated imagery [78], [79].

The capability of YOLO to operate in real time with high detection accuracy and moderate computational requirements makes it particularly suited for embedded robotic platforms deployed in time-sensitive and resource-constrained SaR missions [75].

YOLO is used in all the perception approaches proposed in Chapter 4.

Kalman Filters: Kalman filters are recursive algorithms for estimating unknown state variables of a system from noisy and uncertain measurements. Kalman filter operates in two alternating phases: In the *prediction phase* a model of the system is employed to estimate the next value of the state variables, whereas in the *update phase* this estimate is refined using sensor measurements. These phases are then repeated alternately.

Kalman filters are widely used for *sensor fusion*, i.e., the integration of data from multiple sensors, whether on a single robot or distributed across multiple robotic platforms [80]. This is particularly relevant for SaR missions, especially those involving multi-robot teams.

Kalman filters are also extensively applied to state estimation tasks [81], [82], such as for robot localization by fusing measurements from various sensors, e.g., IMU (Inertial Measurement Unit) and GPS (Global Positioning System) receivers.

The Kalman filter is used in the tracking approach of Chapter 4.

Best Viewpoints for Support Robots: In certain contexts, autonomous robots, particularly UAVs, are equipped with cameras to provide real-time visual feedback for human operators. This is particularly useful in applications such as wildfire suppression, where support (also called *auxiliary*) UAVs follow and monitor fire extinguishing robots to transmit visual data to human operators, as described in more detail in Chapter 6 of this thesis.

A fundamental requirement for such support tasks is to determine the best viewpoints for auxiliary UAVs to monitor main (e.g., fire extinguishing) robots. The work in [83] presents a state-of-the-art method for selecting such best viewpoints to support certain robotic tasks, including object manipulation and navigating narrow passages.

Overview of Search-and-Rescue (SaR) Robotics

This section provides an overview of key topics and state-of-the-art developments related to the application domain of this PhD thesis, namely Search-and-Rescue (SaR) robotics.

SaR Environments: SaR operations can be categorized according to the type of disaster environment [14]:

- *Urban Search-and-Rescue (USaR):* Operations take place in constrained environments, such as collapsed buildings or vehicles [18], [84];
- *Wilderness SaR (WiSaR):* These operations involve locating victims — often mobile — in open spaces, such as mountains, valleys, or forests [85], [86];
- *Air-Sea Rescue (ASR)* or *Maritime SaR:* Operations are focused on rescuing victims in aquatic contexts, including flood zones or maritime accidents [87], [88].

SaR Tasks: Robots are deployed in SaR missions to support various tasks, including:

- *Search*, which involves detecting victims or potential hazards [89];
- *Reconnaissance and mapping*, which aims at enhancing situational awareness of the SaR environment, using AI and robotics to map the environment [84];
- *Rubble removal*, including robots lifting debris heavier than what could be moved manually [90];
- *Structural inspection*, which concerns assessing building interiors and exteriors to predict collapses or to ensure safety for human entry [91];
- *Medical intervention*, enabling remote communication between medical staff and victims, as well as delivering medical supplies [92];
- *Tele-presence*, which provides audio-visual access for human crews and medical staff from a distance [93];
- *Wireless network extension*, where robots act as mobile beacons or repeaters [94];
- *Logistics support*, automating the transportation of tools, equipment, and supplies [95].

Robot Technologies in SaR: Depending on the specific SaR scenario and the involved tasks, different types of robots may be employed, each suited to particular operational requirements.

First, *Unmanned Aerial Vehicles (UAVs)*, also referred to as *flying robots* or *drones*, are commonly used in outdoor environments. UAVs are particularly suited for rapidly covering large areas, traversing inaccessible or hazardous terrain, and high maneuverability that allows them to reach targets closely [14].

Second, *Unmanned Ground Vehicles (UGVs)*, also known as *ground robots*, are deployed in confined areas with limited space. These robots offer stable camera perspectives for precise imaging and are capable of carrying heavy payloads required for various SaR tasks [96]–[98]. Among UGVs, *mobile robots* are particularly valuable in SaR applications, due to their ability to operate in unstructured environments and their versatility with diverse locomotion systems (e.g., wheels, tracks, legs) [99].

Third, *Unmanned Surface Vehicles (USVs)* operate on the surface of the water and are suitable for cluttered marine environments. USVs can autonomously navigate on the surface of water and may be equipped with mechanical features, e.g., robotic grippers or arms, for object retrieval or flotation device deployment. The stable platform of USVs also makes them suited for launching UAVs or for acting as a mobile base station in maritime missions. These robots can also be used for monitoring coastal areas, detecting sea mines, or handling chemicals which can cause health problems [100], [101].

Fourth, *Unmanned Underwater Vehicles (UUVs)* can dive underwater and provide information, such as live videos, temperature, pressure, or depth. UUVs are essential in underwater SaR tasks for locating sunken objects, inspecting submerged structures, or detecting victims in shipwreck scenarios, especially where human divers cannot safely operate. Other tasks that these robots can perform are underwater mapping or leak detection [102], [103].

Humanoid robots are also used for SaR, even if less than other types of robots, due to their slower speed, lower levels of autonomy, and more difficult mobility. Nevertheless, their potential is high with regard to their ability to emulate closely human mobility and dexterity, and being able, for example, to drive a vehicle, operate hand tools such as when opening closed doors, climbing stairs, and moving over any type of terrain [104]. However, they are currently mostly used in manipulation tasks, such as casualty extraction or injured victims transportation [105], or to provide imagery for victim detection as other ground robots can do [106]. Other special types of robots include *creeping* and *serpentine robots*, that are designed for maneuvering through highly constrained spaces and for reaching locations otherwise inaccessible to conventional robots, even if at a slower speed. These behaviors can be achieved thanks to the many more degrees of freedom that they possess, compared to other robots [107].

In various cases, multi-robot systems (e.g., combinations of aerial, ground, and marine robots) are deployed in SaR operations [85], [106]. These systems are particularly useful in large-scale disaster environments, as they allow for broader coverage, increased robustness in case of individual robot failure, and decomposition of complex tasks into simpler sub-tasks that can be performed concurrently [108]. Deployment of multi-robot systems requires advanced coordination and perception mechanisms, including multi-robot path planning and sensor fusion [72].

Sensor Technologies in SaR: SaR robots are equipped with a large variety of sensors, primarily used for mapping the environment, detecting humans, and tracking targets [93]. These sensors include:

- *LiDAR (Light Detection And Ranging) sensors* are range finders based on laser (i.e., light) that calculate distances and are widely used for environment mapping.
- *RaDAR (Radio Detection And Ranging) sensors* and *ultrasonic range finders (SoNARs)* function similarly to LiDARs, but rely on radio and ultrasonic signals, respectively. Their advantage lies in their robustness to lighting conditions and the color or shape of objects [98].
- *Cameras* are among the most affordable sensors and provide rich visual information, including the shape, size, and color of objects. Various types of cameras exist, including:
  - RGB (color) cameras for standard imaging [109],
  - depth cameras for range sensing [110], and
  - infrared or thermal cameras, which detect heat and are especially useful in smoke-filled or dark environments [111].
- *Gas sensors* detect changes in gas concentration and are often employed in fire scenarios [98]. A sub-category of gas sensors, *CO_2 sensors*, can monitor the breathing cycle of a victim, assisting with detecting health status or whether the victim is alive [111].
- *Temperature sensors* are used to detect human body heat, also relevant for monitoring the health status of victims.
- *Humidity sensors* are useful in firefighting contexts, since humidity influences how strongly fuel ignites.
- *Heat sensors* detect live beings, including humans, by measuring emitted infrared radiation.
- *Vibration sensors* are deployed on the ground, indoors or outdoors, to detect movements of victims.
- *IMUs (Inertial Measurement Units)* and *GPS (Global Positioning System) receivers* are essential for robot localization.
- *Microphones* and *speakers* facilitate verbal communication between rescuers and victims [111].

Perception, Planning, and Mapping Methodologies used in SaR: The main methodologies used by SaR robots can be categorized according to the three main topics covered by this thesis, i.e., perception, planning, and mapping, as explained next.

Perception in SaR: *Robotic perception* is a crucial field that enables SaR robots to gather information from the surroundings and interpret it, thus enabling more complex behaviors such as planning, navigation, and mapping.

As cameras are the most common SaR sensors, the majority of the techniques used in perception rely on computer vision, and usually such algorithms are based on deep learning [72].

Semantic segmentation is the process where every part of the image is labeled, e.g., as road, building, sky, etc. [112]. In particular, all the adjacent pixels that form a specific area are assigned the same class label and recognized as objects. These techniques are mostly based on CNNs [113], [114].

Object detection deals with detecting instances of specific objects of a certain class, e.g., victims, specific obstacles, or other particular SaR targets [115]. In fact, with respect to segmentation, only the objects of interest are labeled. These algorithms involve determining bounding boxes for the candidate object in the images, and usually a connected task is to localize these boxes to determine the object position. As for segmentation, most of the object detection techniques are based on CNNs [116], [117].

Since deep learning techniques require a training phase before being employed, datasets are also developed specifically for SaR [118], [119].

When perception relies on data gathered from various sources, e.g., LiDAR data and camera images, it is referred to as *multimodal perception*. In these cases, *sensor fusion* techniques are employed to combine the information [120]. In SaR, the most important data fusion applications concern images and depth information [72]. The algorithms for sensor fusion are usually based on machine learning [121] or Kalman filters [122]. In multi-robot systems, data fusion is also required to combine the environmental information gathered from the sensors placed on different robots [123].

In addition to perceiving the environment, robots must also gather information about their own states. In particular, *localization* of the robot, i.e., determining its position and orientation in the environment, is a special topic where the robot should perceive information about itself rather than the surroundings. It is based on sensors such as IMUs and GPS receivers, and can be performed using state estimation techniques, such as Kalman filters and their variants [19], computer-vision-based methods, such as point cloud processing [96], or probabilistic methods, such as in Monte-Carlo localization, which uses particle filters [106].

Planning in SaR: *Robot planning* is what allows SaR robots to be autonomous, being responsible for the decision in real-time about what they should do next. As every typology of robots, SaR robots require appropriate control methodologies to operate autonomously, as opposed to being teleoperated manually [19]. A middle ground is semi-autonomous control, which allows sharing tasks between the robot and human operators [19].

In robot motion planning, the approaches can be distinguished between *path planning*, when a geometric path is designed without any specified time law, and *trajectory planning*, where a time that the robot should reach and follow is assigned [124]. In simpler cases, *point-to-point control* may be performed, where the robot moves directly towards a target location without following a computed trajectory or path, but rather only

pursuing the final position (and, in case, orientation) [111].

Therefore, to navigate autonomously, robots rely on path planning techniques, which can be broadly divided into global and local path planning, depending on the level of knowledge about the environment. *Global path planning* consists of selecting a complete path using a pre-existing map. *Local path planning*, instead, addresses real-time changes in dynamic environments, focusing on obstacle and collision avoidance and reactive path adjustments. Common approaches for path planning include A* and its variants, such as D* [96], ant colony optimization [125], genetic algorithms [126], FLC [127], and sampling-based methods, such as Rapidly exploring Random Tree (RRT*) [128].

Once a path is generated, control approaches are further required to track planned trajectories or to follow designated paths. These techniques ensure that the robot maintains the desired position, orientation, and (angular and linear) velocities over time (kinematics control). For robots with multiple degrees of freedom, tracking the planned trajectories involves coordinating the corresponding actuators (through kinetics control) for obtaining the desired kinematics.

Common control approaches used for tracking planned paths include Proportional-Integral-Derivative (PID) control [19] and MPC [129], [130]. Recent strategies leverage reinforcement learning [131] or machine learning [132] to control complex components, such as flippers in legged or tracked SaR robots. In multi-robot systems, path planning is typically more complex, as it should consider coordination among multiple agents and targets [133].

In connection with motion planning, *obstacle avoidance* is a fundamental component of a path planning approach. The obstacle avoidance task can be divided into static and dynamic obstacle avoidance, where generally in SaR both cases should be considered, since the SaR environments are usually dynamic [134]. These approaches are often based on deep reinforcement learning [135], [136] or particle swarm optimization [137].

Mapping in SaR: *Mapping* is the process of generating a spatial representation, called a *map*, of the SaR environment.

In unknown or dynamic environments, SaR robots must first perform *exploration*, i.e., understanding the surrounding environment by gaining information through the sensors of the robots, within bounded time [138]. A commonly used strategy for this is frontier-based exploration [139], although alternative approaches, including (deep) reinforcement learning, have also been proposed [140].

The goal of the exploration phase is to generate a map of the SaR environment, that can be subsequently used for navigating inside the known map. For mapping, techniques such as OctoMap, based on octrees, are commonly used due to their memory efficiency and suitability for 3D space representation [106].

While the robot can rely on a known map, more efficient techniques are used to generate maps while navigating, without the need of previous knowledge, that often in SaR context is also not usable since the environment is usually reshaped after a disaster. This process, where a robot simultaneously plans its paths while building a map, is known as *active mapping* [141].

In addition, to navigating effectively and localizing themselves, SaR robots typically employ *Simultaneous Localization And Mapping (SLAM)* algorithms. These techniques

allow robots to build a map of the environment, while simultaneously estimating their own position within the environment. Common SLAM algorithms include Hector [142], which fuses 2D laser scans, or Gmapping [106], based on particle filtering.

Human-robot Interaction in SaR: In real-world SaR, robots often operate in human-robot teams. This requires the integration of robotics and AI technologies for effective human-robot interaction and team cooperation [84].

Such approaches are used to distribute robots and humans in space, time, capability, and role [143]. In such cases, the robots are no longer fully autonomous, but rather rely on adjustable autonomy with human users. For example, approaches are also designed to allow SaR operators to send commands to multi-robot systems, without choosing a specific robot [144].

The human-robot interaction field also involves developing interfaces, such as graphical user interfaces, to improve human-robot collaboration and allow a better switch between teleoperation and autonomous modes [145].

Simulation Models Used in SaR: In SaR missions, especially in dynamic, open spaces, victims may be mobile. Therefore, models of victim behavior are used to simulate their movements and interactions in open areas [86], as well as crowd evacuation models that replicate how individuals exit buildings during emergencies [146], [147]. Additionally, dynamic pedestrian models simulate the motion of people in crowded environments to inform SaR simulations about realistic pedestrian behavior and motion patterns [148], [149].

In firefighting scenarios, simulation models of fire dynamics may also be essential to predict the spread of fire in both indoor and outdoor settings. These models typically account for influencing factors, such as wind conditions and terrain slope [150]. A common framework for microscopic modeling of such phenomena is the cellular automaton approach, which discretizes the environment into a grid to simulate local interactions and fire propagation [151].

Such simulation models support planning, risk assessment, and the development of effective SaR strategies under complex and evolving conditions.

KEY CHALLENGES ADDRESSED IN THE THESIS AND KEY CONTRIBUTIONS

This section outlines the main research challenges that are addressed in this thesis and indicates which chapters are dedicated to each. After each challenge, the central contributions of this PhD thesis are presented, given for each corresponding chapter.

Challenge 1: Formulating Model Predictive Control (MPC)-based Path Planning for Simultaneous Target Tracking and Area Coverage: While MPC has been used in Search-and-Rescue (SaR) robotics, it has primarily focused on tracking pre-defined trajectories generated by other controllers [152], [153]. However, leveraging MPC for exploration in SaR environments, where reference trajectories are unavailable or should be generated and adapted online, remains under-explored.

In typical target-oriented SaR, MPC is used to reach or follow given targets. However, this approach faces two main limitations: First, uncertainty in target positions may lead

to incorrect or empty locations. Second, in most missions, not all target locations are known or estimated in advance, highlighting the significance of balancing target tracking with environmental exploration.

Addressing these open challenges requires MPC formulations capable of dynamically balancing the two mentioned competing objectives. Developing such autonomous MPC-based systems constitutes a key contribution of this thesis.

Chapter 2 introduces the proposed novel MPC-based formulation for mission planning of SaR robots in dynamic environments. The MPC problem includes both target-oriented and coverage-oriented objectives and constraints, while no reference trajectory is provided in advance. We adopt a dynamic force-based evacuation model to represent the movements of the victims, which are the targets that SaR robots should visit at least once. We incorporate the uncertainties regarding the exact position of moving victims and the evolution of these uncertainties into the MPC prediction model. A robust tube-based variant of the proposed MPC formulation is adopted to deal with environmental uncertainties in victims locations and robot motion model. We validate the approach through simulations benchmarked against four state-of-the-art methods and real-life laboratory experiments.

Challenge 2: Ensuring Robust MPC-based Navigation in Presence of Dynamic Obstacles: A critical challenge in autonomous navigation for SaR robots is safe obstacle avoidance, particularly in dynamic environments with moving obstacles. State-of-the-art approaches, especially those using MPC, often assume static obstacles or rely on ad-hoc reactive solutions that lack generalizability and robustness guarantees [154]–[156].

To tackle this challenge, this thesis employs a hierarchical control architecture: (i) Nominal trajectories are generated offline, considering only static obstacles and deterministic dynamics. (ii) A robust MPC-based tracking controller ensures that the robot safely follows these nominal trajectories online, while compensating for dynamic obstacles and bounded disturbances. Such two-tiered approaches ensure both safety and efficiency in time-sensitive SaR deployments.

Chapter 3 presents this hierarchical control approach for motion planning and motion tracking for navigation in environments with dynamic obstacles. We develop an architecture that integrates a heuristic path planner for real-time obstacle avoidance with Robust Tube-based Model Predictive Control (robust TMPC) for trajectory tracking under disturbances and model mismatches. Comparative evaluations demonstrate superior performance in safely reaching specified targets over existing methods, including state-of-the-art versions of Rapidly exploring Random Tree (RRT*) and Artificial Potential Function (APF) for path planning.

Challenge 3: Achieving Reliable Camera-based Perception for Target Tracking and Collaborative Mapping: Affordable SaR robotics is crucial, especially in resource-limited contexts [157]. While conventional sensors, e.g., LiDAR (Light Detection And Ranging), RaDAR (Radio Detection And Ranging), and thermal cameras offer high accuracy, their cost is prohibitive. In contrast, RGB cameras offer a cost-effective solution for perception of SaR robotics, with broad applicability, including mapping, object detection, and tracking, integrating computer vision and deep learning techniques [158], [159].

These cameras may be mounted on robots to capture images or video streams of the environment and to extract meaningful inferences through computer vision techniques. When deployed on flying robots, which offer rapid aerial coverage, challenges include camera instability and limited effectiveness in confined spaces. In contrast, ground robots provide stable, close-range imagery and can access narrow or cluttered environments, but they typically move at a slower pace than flying robots.

This thesis integrates complementary strengths of both platforms by fusing camera views captured by flying and ground robots. This enables more accurate, real-time perception for both target tracking and area mapping.

Chapter 4 of the thesis details the integration of camera-based perception from flying and ground robots for target tracking and area mapping. We extend the pose estimation module of YOLO (You Only Look Once) for estimating unobserved key points of detected human bodies, as well as the distance of the victim from the ground robot that has captured its image. Sensor fusion from the streams of aerial and ground robots is performed for trajectory tracking and terrain elevation estimation via homography. For all these three parts, experiments have been designed and performed in real world, all validating the precision and reliability of the proposed perception techniques.

Challenge 4: Scaling Multi-robot Mapping while Maintaining Efficiency and Robustness A core objective in SaR missions is to reduce the mission time while increasing the area coverage [160]. Robots should alleviate human workload, rather than increasing it, by efficiently operating in a fully-autonomous or semi-autonomous way. Achieving this requires scalable and efficient multi-agent control methods that optimize speed and spatial coverage in real time.

This thesis introduces the Multi-agent Model-Predictive Fuzzy logic Control (M2PFC) framework, a novel bi-level control architecture that integrates the optimal decision making capabilities of MPC with adaptive responsiveness of Fuzzy Logic Control (FLC). This integration allows high-level constrained optimization, while maintaining low-level reactive control under the uncertainties inherent to SaR environments.

We address the limitations of centralized, decentralized, and distributed schemes — respectively, computational bottlenecks, poor coordination, and communication heaviness — by integrating their strengths into a two-layer design that offers both scalability and robustness.

Chapter 5 presents the M2PFC framework, an architecture based on FLC and MPC, offering efficient multi-robot mapping of unknown SaR environments. We propose a bi-level control framework that systematically integrates predictive optimality of MPC and time-efficient, human-inspired responsiveness of FLC, using event-triggered tuning for decentralized control and global coordination maintenance. The performance of the M2PFC framework is validated for autonomous multi-robot path planning in SaR missions, considering a grid-based dynamic environment involving propagating fire. In addition to performance analysis, we execute sensitivity analyses with regard to various control parameters and assessment of different design configurations.

Challenge 5: Coordinating Multiple Unmanned Aerial Vehicles (UAVs) to Maintain Best Viewpoints During Wildfire Suppression Flying robots — also called UAVs — are par-

ticularly promising for wildfire suppression in SaR missions, due to their maneuverability, wide Field of View (FoV), and capacity to quickly cover large areas [14], [15]. Autonomous coordination of these UAVs poses several scientific challenges, particularly for auxiliary UAVs, which should navigate autonomously to maintain best viewpoints of ongoing suppression and to monitor dynamic fire behavior and other robots, while avoiding collisions.

To address this challenge, this thesis integrates optimization-based and reactive control approaches [25], [161], [162]. In particular, we employ *Task Hierarchical Control (THC)* [66] for prioritized multi-task execution, *Path Integral Control (PIC)* [21] for tasks that demand optimality, and best viewpoint planning methods [83] for effective visual coverage.

Chapter 6 introduces the hierarchical and optimal control framework for coordinated multi-UAV systems that maintain the best viewpoints during wildfire suppression missions. We achieve sustained, occlusion-free Fields of View (FoVs) for auxiliary UAVs, centered on a target fire extinguishing UAV, by introducing a novel Line-of-Sight (LoS) obstacle avoidance method combined with best-viewpoint planning, THC, and PIC. Case studies through computer-based simulations, modeling a 3D environment, validate the effectiveness of the proposed approaches in wildfire scenarios.

Challenge 6: Enabling Trustworthy Human-in-the-loop Decision Making for Robot-assisted Firefighting In robot-assisted firefighting, maintaining a balance between robot autonomy and meaningful human control is critical. While increasing robot autonomy reduces operator workload, it risks diminishing moral responsibility and situational awareness of humans. Conversely, excessive reliance on human input may slow down the operation and reduce the benefits of automation.

Achieving effective collaboration between humans and autonomous robots requires robots to make decisions transparently and in ways that humans can trust and interpret. This introduces key challenges, including how to support real-time, explainable decision making by robots balanced with human preferences, how to determine what information to communicate with humans, and how to support trustworthy collaboration between humans and robots.

This thesis addresses these challenges by leveraging FLC to autonomously determine temporary navigation targets for robots which behavior is shown using Explainable Artificial Intelligence (XAI) methods [43]. This provides interpretable explanations that assist decision making by human operators.

Chapter 7 presents this human-in-the-loop framework for firefighting scenarios. We design an architecture that integrates FLC-based robot autonomy with XAI-enabled human oversight. The system proposes navigation targets with interpretable justifications. This allows human operators to retain meaningful control and situational awareness in firefighting missions.

2

A Novel MPC Formulation for Dynamic Target Tracking with Increased Area Coverage for Search-and-Rescue Robots

Robots are increasingly deployed for search-and-rescue (SaR), in order to speed up rescuing the victims in the aftermath of disasters. These robots require effective mission planning approaches to determine time and space-efficient trajectories that steer them faster towards (moving) victims, while dealing with uncertainties. Model predictive control (MPC) is an effective optimization-based control approach that has been used to steer robots along reference trajectories determined by higher level controllers. Determining the trajectory of the robots directly via MPC has the advantage of optimizing multiple SaR criteria while handling the constraints. We, thus, introduce a path planning approach based on MPC for indoor SaR robots that allows the robot to systematically chase the moving victims, when no reference trajectory is provided. The proposed approach combines target-oriented and coverage-oriented search, and allows for systematic handling of environmental uncertainties, by deploying a robust tube-based version of the introduced MPC formulation. In addition, we model the movements of the victims for MPC, by adopting an existing evacuation model. We present a case study, using Gazebo, MATLAB, and ROS, where the performance of the proposed MPC controller is evaluated compared to four state-of-the-art methods (two target-oriented methods based on MPC and A and two heuristic algorithms for area coverage). The results show that, while robust to uncertainties, our approach overall outperforms the other methods, with regards to victim detection, area coverage, and mission time.*

Parts of this chapter have been published in Journal of Intelligent & Robotic Systems **110**, 140 (2024) [1].
M. Baglioni contributed to the methodology, performed the simulations, assessed the results, and wrote the chapter. A. Jamshidnejad contributed to methodology, results, and editing as PhD supervisor.

2.1. INTRODUCTION

RECENTLY Search-and-Rescue (SaR) robots are increasingly used in dangerous and complicated stages of SaR [12], [13], [15], [18]. Robots are expendable and can access locations that are inaccessible to humans. By taking over dangerous tasks, robots allow humans to contribute to other tasks, e.g., providing medical and emotional support for victims [16], [17]. Novel control approaches are needed to enable robots to autonomously and time-efficiently search for trapped people [19].

Model Predictive Control (MPC) [20] is an optimization-based control approach that explicitly incorporates various state and input constraints, and finds balanced trade-offs between various objectives of SaR missions. Furthermore, robust versions of MPC have been established (see, e.g., [26], [31], [38]) that, if adopted for SaR robots, will maintain their performance in presence of uncertainties [36], [37], [96], [133]. While a well-known challenge of MPC methods is the demanding computations, a vast number of literature exists that propose alternative approximate MPC methods to tackle this challenge (see, e.g., [163], [164]).

Although MPC has been used for SaR robotics, the applications mainly concern tracking a reference trajectory that is determined by another controller [152], [153]. However, exploiting the advantages of MPC for systematic exploration of SaR environments when no reference trajectory is available or when it is desired to determine this trajectory directly via MPC remains limited (see Section 2.1.1). Additionally, reference-tracking MPC is used for target-oriented SaR, whereas optimizing the area coverage, next to moving towards the target victims, possesses two main advantages: First, the estimated positions for the target victims are usually prone to uncertainties. Therefore, a solely reference point tracking method may lead the robot to places without victims and thus, the robot should include exploration to its tracking task. Second, area coverage is essential for various SaR missions, in order to find unknown victims or other important targets (e.g., explosives). Running the area coverage and target tracking as two separate or sequential missions will pose additional challenges regarding real-time communication and data exchange and coordinating the robots, and will result in increased mission time, overpopulating the area with robots, etc. Designing an MPC system that determines on the go, according to a constrained (multi-objective) optimization problem, which of the two tasks the robot should take will eliminate those issues.

In this chapter, we propose a novel formulation based on MPC for autonomous decision making of SaR robots, when no reference trajectories exist and the path should be planned by MPC itself. Our main motivation for determining the trajectory of the robots directly via MPC is to optimize the SaR mission with respect to time and the area coverage, especially the coverage of areas that potentially include victims, and to systematically handle the hard constraints via SaR robots. The area coverage problem is dynamic, due to the possibility of movement of the victims or obstacles. Moreover, our main aim is to adopt MPC for systematic combined target and coverage-oriented SaR, while handling various state and control constraints. Satisfying constraints, e.g., avoiding obstacles, fire, and collision, reaching specific terminal targets, and incorporating the limits of the actuators and the kinematics of the robots, is crucial for SaR robots.

The proposed generalized MPC-based framework can be adopted for mission planning of robots for various environments, targets, robots, and unmodeled disturbances in

SaR missions. This framework will exploit the unique characteristics of MPC for mission planning of SaR robots. Moreover, various models for the SaR environment, dynamic targets (e.g., victims), and robots and their actuators can be plugged into MPC as prediction models, which significantly expands the applicability and generalizability of our proposed framework.

The main contributions of this chapter include:

- A novel MPC-based formulation for mission planning of SaR robots in dynamic environments is introduced. While MPC for SaR robots has mainly been used for (robust) reference tracking, which leads the robot to specific targets (target-oriented SaR), we formulate an economic MPC problem that includes both target-oriented and coverage-oriented objectives and constraints. This will significantly improve the efficiency and effectiveness of search-and-rescue, without having a reference trajectory.
- We adopt a dynamic force-based evacuation model to represent the movements of the targets (victims). We also incorporate the uncertainties regarding the exact position of the victims and the evolution of these uncertainties into the model, which will be used by MPC as prediction model. The proposed control approach is not limited to this particular model and may be integrated with different dynamic models for victim/target movement.
- We design and run a case study (implemented in MATLAB, ROS, and Gazebo) to assess the performance of the proposed MPC method with respect to four different state-of-the-art methods, that are dedicated target-oriented or coverage-oriented approaches. For the case study, we adopt a robust tube-based version of the proposed MPC formulation, in order to deal with environmental uncertainties included in victims locations and in the robot model. We also include the results of real-life experiments in the lab.

The remainder of the chapter is structured as the following. Section 2.1.1 provides a background discussion about the control of SaR robots for various SaR objectives. In Section 2.2, we describe the proposed methodologies, including the MPC formulation and the dynamic model for the movement of the victims. In Section 2.3, we show and discuss the results of a case study that compares our approach with four different state-of-the-art methods for simulated indoor SaR missions. Finally, Section 2.4 concludes the chapter and proposes topics for future research. Moreover, Table 2.1 shows the commonly used mathematical notations.

2.1.1. BACKGROUND

Considering the main objectives of their control systems, SaR missions may be categorized as *coverage-oriented* [165], [166] and *target-oriented* [96], [133]. In coverage-oriented SaR, exploring the SaR environment and finding the victims [167]–[170] is the focus of the control system, whereas in target-oriented SaR the control system aims at reaching a specific target (e.g., an exit) [171]. The most commonly used coverage-oriented approaches for mission planning of SaR robots include heuristics techniques (e.g., bug algorithms [172], potential fields methods [173], fuzzy logic control [48], and

Table 2.1: Table of frequently used mathematical notation

	MPC
Variable	Description
$A_i(k)$	Area that is expected to include victim i at time step k
$C^{\mathrm{p}}(k)$	Circular area representing the perception field of the SaR robot at time step k
k^{c}	Control time step
k^{s}	Simulation time step
$\mathscr{M}^{\mathrm{Evac}}$	Mapping corresponding to "Evac" crowd evacuation model
N^{p}	MPC prediction horizon
$\boldsymbol{r}^{\mathrm{rob}}(k)$	Position of the robot at time step k
T^{c}	Control sampling time
$v^{\mathrm{rob}}(k)$	Linear velocity of the robot at time step k
$\omega^{\mathrm{rob}}(k)$	Angular velocity of the robot at time step k
$x^{\mathrm{rob}}(k)$	Coordinate x of the robot at time step k
$y^{\mathrm{rob}}(k)$	Coordinate y of the robot at time step k
$\theta^{\mathrm{rob}}(k)$	Orientation angle of the robot at time step k
π^{c}	Threshold used by MPC for intersection of the areas $C^{\mathrm{p}}(k)$ and $A_i(k)$ to stop chasing victim i
	Evacuation model
Variable	Description
$\boldsymbol{F}_i(k)$	Total external force on victim i at time step k
$\boldsymbol{F}^{\mathrm{s}}_{ij}(k)$	Social force between victims i and j at time step k
$\boldsymbol{F}^{\mathrm{c}}_{ij}(k)$	Contact force between victims i and j at time step k
$\boldsymbol{F}^{\mathrm{a}}_{ij}(k)$	Attraction force between victims i and j at time step k
$\boldsymbol{F}^{\mathrm{s}}_{io^{\mathrm{s}}}(k)$	Social force between victim i and static obstacle o^{s} at time step k
$\boldsymbol{F}^{\mathrm{c}}_{io^{\mathrm{s}}}(k)$	Contact force between victim i and static obstacle o^{s} at time step k
$\boldsymbol{F}^{\mathrm{a}}_{io^{\mathrm{d}}}(k)$	Attraction force between victim i and dynamic obstacle o^{d} at time step k
I_i	Moment of inertia of victim i
m_i	Mass of victim i
$\boldsymbol{r}^{\mathrm{v}}_i(k)$	Position of victim i at time step k
$T_i(k)$	Total external torque on victim i at time step k
$T^{\mathrm{s}}_i(k)$	Torque of the social force on victim i at time step k
$T^{\mathrm{c}}_i(k)$	Torque of the contact force on victim i at time step k
$T^{\mathrm{m}}_i(k)$	Torque of the motive force on victim i at time step k
$\boldsymbol{v}^0_i$	Velocity vector field of victim i
$\boldsymbol{v}^{\mathrm{v}}_i(k)$	Linear velocity of victim i at time step k
$\eta_i(k)$	Fluctuation torque component on victim i at time step k
$\theta^{\mathrm{v}}_i(k)$	Orientation angle of victim i at time step k
$\boldsymbol{\xi}_i(k)$	Fluctuation force component on victim i at time step k
τ_i	Time relaxation parameter of victim i

particle swarm optimization [137]). Graph-based [174], [175] and optimization-based control approaches, including MPC, are often used in target-oriented SaR. An extensive literature survey shows that the use of MPC in coverage-oriented SaR is rather limited (see, e.g., [25], [30], [176]–[178]). In particular, MPC is used for reference tracking [152], i.e., when a reference path that steers the robot to a specific known destination is available.

We first discuss some state-of-the-art coverage-oriented approaches. Although the focus of our chapter is on single robot control, in order to properly cover various control methods, we consider papers with both single and multiple robot systems. Since most approaches decompose the environment into cells, a main classification of coverage-oriented methods includes exact and approximate decomposition methods. Using an exact decomposition, Agarwal and Akella in [179] consider a multi-robot system for exploring a given 2-dimensional area, where the robots have constraints on their flight time and battery life. The problem is transformed into a line-coverage one, which is then solved by a graph-based method. In [180], a coverage-oriented path planning method based on approximate cellular decomposition is proposed that is shown to reduce the coverage overlap. Their approach, however, does not account for dynamic obstacles and uncertainties. In [181] a bio-inspired ant colony optimization algorithm is proposed, where variable velocities are introduced for the ants, in order to find sub-optimal paths, while reducing the overall path planning time. This approach resulted in improved computational efficiency and reduced repetition in covering the same areas. In [182] four methods are proposed for discrete-space path planning of Unmanned Aerial Vehicles (UAVs) in SaR missions. Using fuzzy logic, a map of the search area is generated where places with a higher risk and a higher probability of finding the victims are specified. The four methods for path planning are based on artificial intelligence, and thus systematic obstacle avoidance and optimization of the control inputs have not been considered. In [183] a distributed particle swarm optimization algorithm for path planning of a multi-robot SaR system is proposed. The robots use artificial potential functions to avoid static obstacles in the environment and to reach the victims. With respect to classical particle swarm optimization methods, through the introduction of a repulsive force among the particles and robots, the performance is improved considering exploration of the area and avoiding collisions between the robots. However, uncertainties in the SaR environment and dealing with moving obstacles have not been considered in [183]. In [184], a coverage path planning algorithm for multiple UAVs is proposed, for a SaR scenario with wind. The authors perform sensitivity analysis with different sizes of the scenario and number of UAVs, and they show good performance even with large scenarios and a large number of UAVs, for which the algorithm speed may also decrease. However, obstacles are not considered, and in addition they make strong assumptions on the wind directions in connection with the UAVs direction, so robustness to real-world uncertainties may not be achieved. In [11], a case study is designed for multiple UAVs that explore a SaR environment. They address search and coverage control, where they show how much coverage of the area they can achieve with cooperative or non-cooperative methods, i.e., when exchange of information among robots is present or not. They validate their method in both computer-based simulations and laboratory experiments, also in the presence of uncertainties. However, they make high simplifications on the robots

models, including constant flying altitude and speed, therefore the applicability in real-world scenarios might be limited. Other papers that have considered heuristic control of SaR robots include [185]–[189], which particularly focus on reinforcement learning. A main challenge with reinforcement-learning-based approaches, however, is that these methods do not systematically incorporate the constraints of a SaR mission. To summarize, optimality (as a balanced trade-off) with respect to multiple competing criteria and incorporation of constraints are not systematically handled by these approaches.

In [25] a control architecture including an MPC layer is presented for combined target-oriented and coverage-oriented mission planning of SaR robots. The architecture is suitable for multi-robot systems and the main task of MPC is coordination of the robots, rather than directly steering them based on the solutions of a constrained multi-objective optimization problem. The robots are steered by individual local heuristic fuzzy logic controller towards various targets. In summary, the common aspect of this chapter and [25] is in adoption of MPC for systematic maximization of the coverage of (partially) unknown SaR environments. However, while in [25] MPC is a supervisory layer that only acts when local fuzzy logic controllers fail to cover the area above a given threshold, in this chapter a reformulation of the objective function and constraints of MPC is proposed for direct steering of SaR robots. The resulting framework is generalizable, as explained earlier, and exploits the unique advantages of MPC for various SaR scenarios.

Since the architecture in [25] follows the same objective as this chapter, i.e., approaching more target victims, while maximizing the area coverage, we compare the performance of our approach with that architecture.

In [30] distributed MPC is used for discrete-space and discrete-time path planning of a cooperative multi-drone system that searches an outdoor environment. The objective of MPC is to determine the speeds and roll angles of the drones in order to maximize their reward for visiting the cells of an environment, which is prone to changes due to the wind flow. In [190], Carron and Zeilinger propose a coverage control method based on nonlinear MPC for a multi-robot system with nonlinear dynamics and state and input constraints. Moreover, a distributed coverage control problem is solved in [176] via MPC. The approaches in both papers converge to a centroidal Voronoi configuration. Our problem differs from that in these papers, since we seek a trade-off between systematic chasing of dynamic targets in uncertain environments (which is not considered in [190] and [176]) and maximizing the area coverage dynamically. In [177] and [178], Ibrahim et al. present control approaches for path planning of robots for area coverage, based on MPC. Compared to their bi-level discrete-space approach, we employ only one level of MPC for a continuous-space problem. The last two papers introduce dynamic obstacles within the environment that the robot should avoid. This is different from our problem where some dynamic victims are assigned as targets for a robot and the robot should estimate the future evolution of their movement, including possible uncertainties, and chase them time and space-efficiently.

Next, we give an overview on papers that consider target-oriented SaR. The number of papers in this category that use MPC for path planning is much more vast compared to coverage-oriented SaR. In [191], a path planning approach based on decentralized robust MPC is proposed for cooperative, collision-free navigation of multiple wheeled robots in an unknown, static, and cluttered environment towards static tar-

gets with known positions. One MPC layer is used to determine way-points according to nominal trajectories for the robots, and a second MPC layer uses a robust approach to track these trajectories despite uncertainties. Farrokhsiar et al. develop a two-layer robust tube-based MPC controller in [192] for motion planning of a unicycle robot that should reach a static target with a known position in a cluttered environment. Their results show that the robust tube-based MPC controller stabilizes the position of the robot around the planned nominal trajectory in dynamic cluttered environments.MPC is also used in [193] for mission planning of an autonomous ground robot in an unknown environment with static obstacles that should reach a static target with a known position, while maintaining a safe distance from the obstacles. The robot has access only to information within its perception field, and accordingly determines intermediary goal positions and moves towards them using an MPC controller with two possibly competing objectives, i.e., reduction of the mission time and the energy consumption. While the results show good performance and constraint satisfaction in an environment with static obstacles, dynamic obstacles and targets, and uncertainties in the position of the target(s) remain topics for future research. In [153] MPC is used in an outdoor mission with an unmanned aerial vehicle for tracking dynamic targets with known initial positions and speeds, considering the wind flow, in an obstacle-free environment. Within recent papers in the category of target-oriented SaR missions that are not based on MPC, in [194] an exploration approach based on reinforcement learning for multi-agent SaR is proposed, where the agents should reach multiple targets. The authors show that their proposed method achieves higher success rate, compared to a frontier-based method and other reinforcement learning approaches, because they are able to better track unknown dynamic targets, but they do not show the performance compared to standard planning approaches such as artificial potential fields or sampling-based approaches. In addition, obstacles are not considered, limiting the applicability to uncluttered environments. In [195], a path planning algorithm for UAVs based on Rapidly exploring Random Tree (RRT*) is proposed, involving a modified sampling strategy that, by increasing the amount of samples in regions with higher densities of obstacles and vice versa, adapts to specific characteristics of the environment to enhance efficiency of target searching. This modified approach demonstrates improved success rates and reduced computational complexity compared to conventional path planning methods based on earlier versions of RRT*, particularly in scenarios with high obstacle densities. However, a notable limitation of this approach is restricted realism and applicability, due to neglecting the dynamics of the UAV that otherwise would allow to improve maneuvering efficiency, however this is challenging because it cannot be inserted as a constraint on the way-points as it is currently done. In [196], a mission planning approach is proposed for UAVs that perform search in a dynamic environment with uncertainties. Similarly to the approach in this chapter, they allow for switching between tasks of area coverage and target surveillance, but, nevertheless, the model of dynamic targets does not achieve the detailed modeling that we achieve in this chapter in order to systematically track and chase the targets.

In connection to the topic of this chapter, we also consider works on search and pursuit-evasion in mobile robotics [197]. In [198], a distributed algorithm for target tracking by a group of pursuers is presented, introducing a modified version of the Voronoi

tassellation that generates collision-free areas. The pursuers are capable of tracking the dynamic target even with an uncertain position. A drawback of the approach is that the presence of a barricade of obstacles does not allow reaching the target. In this field, there are also works that use MPC. In fact, in [199] an MPC approach is proposed and compared to a game-theoretic method. It is shown that MPC can solve this problem with less information required and with a similar performance. In more detail, in [200] the same authors show that MPC requires less information to solve pursuer-evasion games, where in particular the position of the opponent is required, but its orientation and dynamics are not. In [201] De Simone et al. introduce a pursuer-evader problem for humanoids in presence of obstacles. Firstly, unicycle models are used to generate reference velocities, and lastly, MPC is used for tracking them for stable gait generation.

Finally, other related topics are robot exploration and target-driven navigation. The first group include studies that discuss and propose exploration approaches, many of which use learning-based techniques, that can be reinforcement learning [202] or deep learning [203]. In particular, in [140] deep reinforcement learning is combined with the more traditional approach of frontier-based exploration for autonomous exploration of unknown cluttered environments by a SaR robot, while in [204] an approach based on reinforcement learning is used that can leverage structural regularities of the environment, robustness to errors in state-estimation, and flexibility with respect to input modalities, or also in [205] a multi-robot exploration strategy in presence of communication dropouts is presented. The second group includes papers on navigation approaches guided by target information, where many use reinforcement learning [206], [207], foundation models based on deep learning [208], [209], or solve optimization problems [210]. In more detail, [206] proposes a navigation approach to reach a target only using vision information based on two networks that respectively explore the environment and locate the target, or in [207] an attention-based method is proposed to learn to navigate by leveraging an episodic memory that embeds previously-visited states, while in [211] a frontier-based exploration method that exploits visual information to navigate towards unseen semantic objects is presented.

2.2. PROPOSED METHODOLOGIES

In this section, we explain our proposed methodology, including the problem definition, assumptions and models, and mathematical formulations.

2.2.1. PROBLEM STATEMENT

The SaR control problem is formulated within a discrete-time and continuous-space framework. We consider an autonomous ground robot that follows the standard unicycle model (see Section 2.2.2 for details). This robot should explore a partially known 2-dimensional SaR environment $\mathscr{E}$, while chasing specific moving targets. Figure 2.1 illustrates an example of a SaR environment and its static and dynamic elements. More specifically, the SaR environment contains a time-invariant set $\mathscr{O}^{\mathrm{s}}$ of sub-areas occupied by static and closed obstacles, e.g., debris and stones with generally arbitrary shapes (see the dashed cyan shapes in Figure 2.1; set $\mathscr{O}^{\mathrm{s}}$ also includes a safety margin for the robot around each obstacle), and a time-varying set $\mathscr{O}^{\mathrm{d}}(k)$ of sub-areas occupied by moving

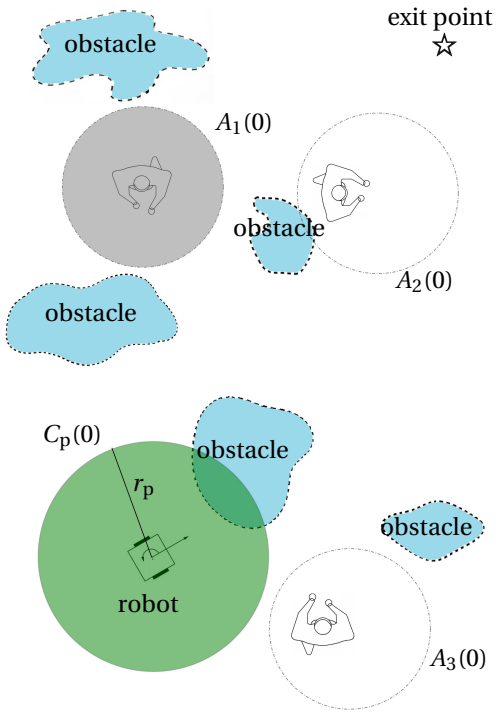


Figure 2.1: Search-and-rescue environment at initial time step $k = 0$ including the SaR robot and its perception field $C^p(0)$ (illustrated with a solid green circle), three victims and their approximate initial areas $A_1(0), A_2(0), A_3(0)$ that represent the uncertain locations of the victims (illustrated with dash-dotted grey circles in case of target victims, or with dash-dotted uncolored circles in case of non-target victims), static obstacles (illustrated with dashed cyan circles), and the exit (illustrated by a star symbol).

objects at time step k, e.g., victims ($\mathcal{O}^d(k)$ includes a safe margin that allows the robot to get as close as needed to its target victims, without crashing into or disturbing them).

The assumptions used are given below:

A1 The robot is associated with a set $\mathcal{V}$ of N^v moving target victims and is initially given a set $\mathcal{A} = \{A_1(0), \ldots, A_{N^v}(0)\}$ of N^v closed circular areas that represent the approximate initial location of each target victim (i.e., $\boldsymbol{r}_i^v(0) \in A_i(0)$, with $i = 1, \ldots, N^v$). Victim i is assumed to initially be inside area $A_i(0)$. Before the robot detects a target victim, it only has knowledge about the approximate initial area $A_i(0)$ of the victim. Thus the robot predicts the evolution and transition of areas $A_i(k)$ in time according to a victims movement model.

A2 An initial exploration phase of the environment has already resulted in knowledge about the approximate initial location of the victims and the position, size, and number of the static obstacles. In real world, the bounded uncertainties corresponding to the areas where victims are located may be estimated based on the likely whereabouts of the victims (e.g., offices, canteens, bedrooms), depending on the functionality and expected number of inhabitants of the building, size and

position of the rooms, time of the day.

A3 From every position in the environment, where the robot may be placed, there is a feasible path to the final target of the robot (i.e., the exit).

A4 The robot has perfect knowledge of its own states at every time step.

A5 The robot is equipped with a sensor with a circular perception area $C^{\mathrm{p}}(k)$ of fixed radius r^{p} (see the solid green circle in Figure 2.1). The center of the perception field always coincides with the position $\left(x^{\mathrm{rob}}(k), y^{\mathrm{rob}}(k)\right)$ of the center of gravity of the robot. The sensor has perfect perception. At every time step, the sensor provides a map of the sub-area that is encountered by $C^{\mathrm{p}}(k)$, including the shape, size, and location of the perceived obstacles[1]. The control system assumes that the target victim i has been detected when the surface of the intersection area for $C^{\mathrm{p}}(k)$ and $A_i(k)$ has reached a specific threshold, given as a percentage of $A_i(k)$.

The robot uses a dynamic model to predict the evolution of the approximate areas $A_1(k),\ldots,A_{N^{\mathrm{v}}}(k)$ in time (see Section 2.2.2 for details), which may be influenced by a motive force field, social forces among the victims, and repulsion forces with respect to static obstacles.

The aim of this chapter is to develop a dynamic and online mission planning control system for the robot that steers the robot within the free space $\mathscr{E}\setminus\left(\mathscr{O}^{\mathrm{s}}\cup\mathscr{O}^{\mathrm{d}}(k)\right)$ towards its final destination (an exit represented by a star symbol in Figure 2.1) in the shortest possible time, while the robot maximizes a trade-off between its area coverage and the possibility of visiting more target victims from the set $\mathscr{V}$, continuously satisfying the hard state and input constraints. Therefore, the SaR problem should be formulated as a combined target-oriented and coverage-oriented multi-objective MPC problem that incorporates all the constraints (see Section 2.2.2 for details).

2.2.2. MPC FOR COMBINED DYNAMIC TARGET CHASING AND AREA COVERAGE

We first explain the models used for the mobile robot and for movement of the victims. Then we detail the MPC formulation.

KINEMATICS MODEL FOR THE GROUND ROBOT

For the robot, we consider the following kinematics model, which is a standard unicycle model:

$$x^{\mathrm{rob}}(k^{\mathrm{s}}+1) = x^{\mathrm{rob}}(k^{\mathrm{s}}) + v^{\mathrm{rob}}(k^{\mathrm{s}})\cos\left(\theta^{\mathrm{rob}}(k^{\mathrm{s}})\right)T^{\mathrm{s}} + \delta_x(k^{\mathrm{s}}) \tag{2.1a}$$

$$y^{\mathrm{rob}}(k^{\mathrm{s}}+1) = y^{\mathrm{rob}}(k^{\mathrm{s}}) + v^{\mathrm{rob}}(k^{\mathrm{s}})\sin\left(\theta^{\mathrm{rob}}(k^{\mathrm{s}})\right)T^{\mathrm{s}} + \delta_y(k^{\mathrm{s}}) \tag{2.1b}$$

where $[\boldsymbol{r}^{\mathrm{rob}\top}(k^{\mathrm{s}}), \theta^{\mathrm{rob}}(k^{\mathrm{s}})]^{\top}$ is the state vector of the robot at simulation time step k^{s}, with $\boldsymbol{r}^{\mathrm{rob}}(k^{\mathrm{s}}) = [x^{\mathrm{rob}}(k^{\mathrm{s}}), y^{\mathrm{rob}}(k^{\mathrm{s}})]^{\top}$ including the x and y coordinates corresponding to the center of gravity of the robot and the angular position of the robot with respect to

[1] Sensor fusion and robustness to sensor inaccuracies are out of the scope of this chapter and are thus assumed to be provided for the robot.

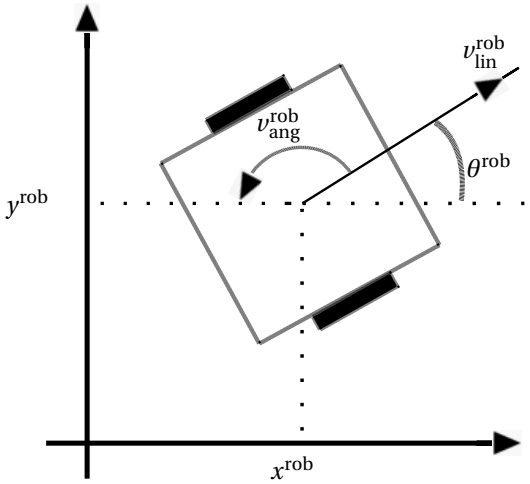


Figure 2.2: Two-wheeled ground mobile robot schematic.

the horizontal axis (see Figure 2.2). Moreover, $v^{\text{rob}}(k^{\text{s}})$ is the linear velocity of the robot at simulation time step k^{s} and T^{s} is the simulation sampling time. The model mismatch with respect to the real robot is formulated via bounded uncertainties $\delta_x(k^{\text{s}})$ and $\delta_y(k^{\text{s}})$; these uncertainties can represent the non-smoothness of the terrain in a SaR environment, that makes the robot real position deviate from the nominal one. With this model, we assume that the linear and rotational motions of the robot are decoupled, i.e., at the beginning of every simulation sampling time, the robot makes a rotation according to $\theta^{\text{rob}}(k)$ (with its maximum possible angular speed and in a negligible time) and continues its movement according to $v^{\text{rob}}(k^{\text{s}})$. Note that the linear velocity $v^{\text{rob}}(k^{\text{s}})$ and the angular position $\theta^{\text{rob}}(k^{\text{s}})$ are control inputs (i.e., optimization variables of the MPC).

DYNAMIC MODEL FOR MOVEMENT OF THE VICTIMS

In order to model the movement of the victims and thus estimate the evolution of their approximate areas, we consider the crowd evacuation model FDS+Evac [146], which is a well established model used, e.g., in [212]–[214].

Remark 1. *Our approach is independent of the selected crowd evacuation model, and thus a different model can be used. For this reason, the application is not limited to SaR. However, in order to properly use FDS+Evac model, in case moving obstacles exist in the SaR environment, the robot should be aware of them and should have a model of their motion.*

In FDS+Evac model, the movements including the position and orientation of humans in a disaster scene are driven by (physical and virtual) forces and torques, such as the contact forces and the gravity, as well as the psychological forces, based on the social force model by Helbing and Molnar [215], that are exerted by the environment. The movement trajectories of the victims in this model are formulated in continuous time and continuous space. The equations of motion for target victim i ($i = 1, \ldots, N^{\text{v}}$)

are given by:

$$m_i \frac{\mathrm{d}^2 \boldsymbol{r}_i^{\mathrm{v}}(t)}{\mathrm{d}t^2} = \boldsymbol{F}_i(t) + \boldsymbol{\xi}_i(t) \tag{2.2}$$

$$I_i \frac{\mathrm{d}^2 \theta_i^{\mathrm{v}}(t)}{\mathrm{d}t^2} = T_i(t) + \eta_i(t) \tag{2.3}$$

where m_i is the mass of target victim i, $\boldsymbol{r}_i^{\mathrm{v}}(t) = [x_i^{\mathrm{v}}(t), y_i^{\mathrm{v}}(t)]^\top$ is the position of the target victim at time instant t with $x_i^{\mathrm{v}}(t)$ and $y_i^{\mathrm{v}}(t)$ the x and y coordinates, $\boldsymbol{F}_i(t)$ is the total measured external force on target victim i at time instant t, $\boldsymbol{\xi}_i(t)$ is a random fluctuation force at time instance t, I_i is the moment of inertia of target victim i, $\theta_i^{\mathrm{v}}(t)$ is the angular position (i.e., heading) of the target victim at time instant t, $T_i(t)$ is the total measured external torque on the target victim at time instant t, and $\eta_i(t)$ is a random fluctuation torque at time instant t. The fluctuation components $\boldsymbol{\xi}(t)$ and $\eta_i(t)$ are part of the uncertainties that affect the control system of the robot.

The total measured external force and torque on target victim i are given by:

$$\begin{aligned}\boldsymbol{F}_i(t) = \frac{m_i}{\tau_i}(\boldsymbol{v}_i^0 - \boldsymbol{v}_i^{\mathrm{v}}(t)) + \sum_{\substack{j=1 \\ j\neq i}}^{N^{\mathrm{v}}} \left(\boldsymbol{F}_{ij}^{\mathrm{s}}(t) + \boldsymbol{F}_{ij}^{\mathrm{c}}(t) + \boldsymbol{F}_{ij}^{\mathrm{a}}(t)\right) + \\ \sum_{o^{\mathrm{s}} \in \bar{\mathcal{O}}^{\mathrm{s}}} \left(\boldsymbol{F}_{io^{\mathrm{s}}}^{\mathrm{s}}(t) + \boldsymbol{F}_{io^{\mathrm{s}}}^{\mathrm{c}}(t)\right) + \sum_{o^{\mathrm{d}} \in \bar{\mathcal{O}}^{\mathrm{d}}(t)} \boldsymbol{F}_{io^{\mathrm{d}}}^{\mathrm{a}}(t)\end{aligned} \tag{2.4}$$

$$T_i(t) = T_i^{\mathrm{s}}(t) + T_i^{\mathrm{c}}(t) + T_i^{\mathrm{m}}(t) \tag{2.5}$$

In (2.4), the first term on the right-hand side corresponds to the motive force on the target victim whose intended velocity at time instant t (i.e., the time derivative of $\boldsymbol{r}_i^{\mathrm{v}}(t)$) is $\boldsymbol{v}_i^{\mathrm{v}}(t)$, and $\boldsymbol{v}_i^0$ is the velocity vector field, which steers the target victim towards an exit of the building. The relaxation time parameter τ_i is an indicative of how fast the target victim reaches the intended speed. The second term in (2.4) corresponds to the interactions among victim i and other target victims, including social, contact, and attraction forces. The third term corresponds to the interactions between target victim i and static obstacles in the SaR environment, including the social and contact forces. The last term represents other interactions among the victim and the SaR environment, including the attraction forces with respect to the moving or propagating obstacles (e.g., fire-human repulsion). In (2.5), the three terms on the right-hand side are the torques corresponding to the social, contact, and motive forces, respectively. For a more detailed description of the force and torque components in (2.4) and (2.5), see [146]. In our model, the forces and torques exerted by the other agents are instead exerted by each victim area $A_i(k^{\mathrm{s}})$, which location is indeed available for the robot. Since our SaR problem is formulated in discrete time, a discretized version of (2.2)-(2.5) formulated as a first-order difference equation for the state vector $\boldsymbol{s}_i^{\mathrm{v}}$ can be used, i.e.,

$$\boldsymbol{s}_i^{\mathrm{v}}(k^{\mathrm{s}}+1) = f^{\mathrm{v}}\left(\boldsymbol{s}_i^{\mathrm{v}}(k^{\mathrm{s}}), \boldsymbol{F}_i^{\mathrm{total}}(k^{\mathrm{s}})\right) \tag{2.6}$$

where:

$$\boldsymbol{s}_i^{\mathrm{v}}(k^{\mathrm{s}}) = \left[\boldsymbol{r}_i^{\mathrm{v}}(k^{\mathrm{s}})^\top, \dot{\boldsymbol{r}}_i^{\mathrm{v}}(k^{\mathrm{s}})^\top\right]^\top \tag{2.7}$$

$$\boldsymbol{F}_i^{\mathrm{total}}(k^{\mathrm{s}}) = \left[\frac{\boldsymbol{F_i}(k^{\mathrm{s}})^\top}{m_i}, \frac{\boldsymbol{\xi_i}(k^{\mathrm{s}})^\top}{m_i}\right]^\top \tag{2.8}$$

and:

$$f^{\mathrm{v}}(\zeta_1, \zeta_2) = e^{AT^{\mathrm{s}}}\zeta_1 + \zeta_2 B \int_0^{T^{\mathrm{s}}} e^{Az}\, dz \tag{2.9}$$

with:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ \frac{1}{m_i} & \frac{1}{m_i} \end{bmatrix} \tag{2.10}$$

and for further details, see [216].

Remark 2. *For the ease of computations, after each evolution of $A_i(k^{\mathrm{s}})$ one may consider the convex hull of the resulting area. However, when the degree of non-convexity is large (i.e., the ratio of the surface of the convex hull and the surface of the original area is larger than a threshold), we allow the area to separate into different sub-areas and then consider the convex hulls of these sub-areas. This means that there may be more than one potential area in the SaR environment that embeds that particular victim. As soon as the victim is detected by the robot, all these sub-areas will be removed from the memory and prediction module of the robot. Moreover, whenever the surface of $A_i(k^{\mathrm{s}}) \cap A_j(k^{\mathrm{s}})$ for two different victims i and j is larger than a specific percentage of the surface of the smaller area, the two victims are assumed to approach each other to move on as a team. Therefore, their approximate areas will be merged. With regard to the evolution of the areas $A_i(k^{\mathrm{s}})$, their motion is obtained by considering each point of the border of $A_i(k^{\mathrm{s}})$ moving based on* (2.2)-(2.5).

Remark 3. *In large-scale implementations, to improve the computation time whenever $C^{\mathrm{p}}(k^{\mathrm{s}}) \cap A_i(k^{\mathrm{s}}) \neq \emptyset$, but the victim is not yet detected (see Figure 2.3), the intersection will be excluded from the approximate area $A_i(k^{\mathrm{s}})$.*

FORMULATION OF MPC FOR COVERAGE-ORIENTED SAR

The main idea of robust tube-based MPC approach is to decompose the control problem into two problems [217], where first a deterministic nominal MPC problem (considering the nominal models and no disturbances) is solved to determine a desired state sequence and the corresponding control input sequence within the prediction window of the MPC. Next, a feedback-based control problem is formulated that determines a control input increment per control time step in order to keep the realized state sequence as close as possible to the desired one determined by the nominal MPC. For the SaR problem, we formulate the nominal MPC problem (2.11).

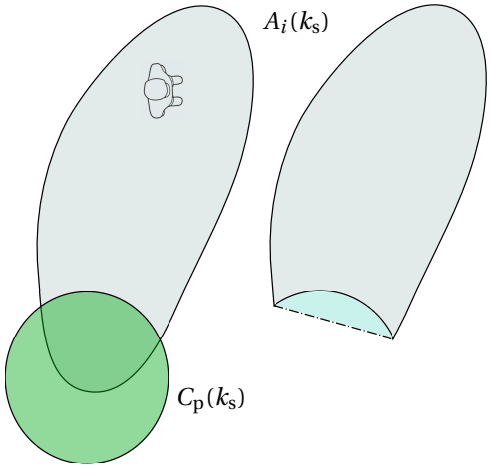


Figure 2.3: When the perception field of the SaR robot intersects with the proximate area $A_i(k^s)$ of a victim, but does not detect the victim in the intersected area, the intersection will be excluded from the modelled approximate area for the robot to improve the computational efficiency.

$$\min_{\tilde{\boldsymbol{v}}(k^c)} J^{nom}\left(k^c, \tilde{\boldsymbol{r}}^{rob}(k^c), \tilde{\boldsymbol{v}}(k^c)\right) \tag{2.11a}$$

s.t. for $\kappa = k^c, \ldots, k^c + N^p - 1$

$$A_i(\kappa+1) = \mathscr{M}^{Evac}\left(A_i(\kappa)\right), \text{ for } i = 1, \ldots, N^v \tag{2.11b}$$

$$\text{if } |C^p(\kappa) \cap A_i(\kappa)| \geq \pi^c |A_i(\kappa)|, \text{ then } A_i(\bar{\kappa}) = \emptyset, \text{ for } \bar{\kappa} = \kappa+1, \ldots, \kappa+N^p \tag{2.11c}$$

$$C^p(\kappa+1) = \mathscr{T}\left(C^p(\kappa), \left[\cos\left(\theta^{rob}(\kappa)\right), \sin\left(\theta^{rob}(\kappa)\right)\right]^\top v^{rob}(\kappa)\right) \tag{2.11d}$$

$$v^{rob}(\kappa) \leq v^{rob,max} \tag{2.11e}$$

$$\Delta v^{rob}(\kappa) \leq \Delta v^{rob,max} \tag{2.11f}$$

$$\Delta\theta^{rob}(\kappa) \leq \omega^{rob,max}\tau, \text{with } \tau \ll T^c \tag{2.11g}$$

$$x^{rob}(\kappa) \text{ and } y^{rob}(\kappa) \text{ evolve according to nominal version of kinematics model by (2.1)} \tag{2.11h}$$

$$\boldsymbol{v}(\kappa) = [v^{rob}(\kappa), \theta^{rob}(\kappa)]^\top \tag{2.11i}$$

$$\boldsymbol{r}^{rob\top}(\kappa+1) \in \mathscr{E} \setminus \left(\mathscr{O}^s \cup \mathscr{O}^d(k)\right) \tag{2.11j}$$

$$J^{nom}\left(k^c, \tilde{\boldsymbol{r}}^{rob}(k^c), \tilde{\boldsymbol{v}}(k^c)\right) = J^{nom,target}\left(k^c, \tilde{\boldsymbol{r}}^{rob}(k^c), \tilde{\boldsymbol{v}}(k^c)\right) + J^{nom,coverage}\left(k^c, \tilde{\boldsymbol{r}}^{rob}(k^c), \tilde{\boldsymbol{v}}(k^c)\right) \tag{2.12}$$

$$J^{\text{nom,target}}\Big(k^{\text{c}}, \tilde{\boldsymbol{r}}^{\text{rob}}(k^{\text{c}}), \tilde{\boldsymbol{v}}(k^{\text{c}})\Big) =$$

$$\sum_{\kappa=k^{\text{c}}}^{k^{\text{c}}+N^{\text{p}}-1} \Big(w_0 \Big(v^{\text{rob}}(\kappa) \Big)^{-1} + \tag{2.13a}$$

$$w_1 \big| C^{\text{p}}(\kappa) \cap A_1(\kappa) \big|^{-1} + \ldots + w_{N^{\text{v}}} \big| C^{\text{p}}(\kappa) \cap A_{N^{\text{v}}}(\kappa) \big|^{-1} + \tag{2.13b}$$

$$w_{N^{\text{v}}+1} \big| C^{\text{p}}(\kappa) \cap (A_1(\kappa) \cup \ldots \cup A_{N^{\text{v}}}(\kappa)) \big|^{-1} \Big) + \tag{2.13c}$$

$$w_{N^{\text{v}}+2} \Big\| \Big[x^{\text{rob}}(k^{\text{c}}+N^{\text{p}}),\ y^{\text{rob}}(k^{\text{c}}+N^{\text{p}}) \Big]^{\top} - \big[x^{\text{exit}}, y^{\text{exit}} \big]^{\top} \Big\| \tag{2.13d}$$

$$J^{\text{nom,coverage}}\Big(k^{\text{c}}, \tilde{\boldsymbol{r}}^{\text{rob}}(k^{\text{c}}), \tilde{\boldsymbol{v}}(k^{\text{c}})\Big) = \sum_{\kappa=k^{\text{c}}+1}^{k^{\text{c}}+N^{\text{p}}} \sum_{n=k^{\text{c}}}^{\kappa-1} w_{N^{\text{v}}+3} |C^{\text{p}}(\kappa) \cap C^{\text{p}}(n)| \tag{2.14}$$

In (2.11), k^{c} is the control time step, N^{p} is the MPC prediction horizon, and tilde is used for a variable to indicate the sequence of that variable across the MPC prediction horizon, e.g., $\tilde{\boldsymbol{r}}^{\text{rob}}(k^{\text{c}})$ is the sequence of the predicted nominal state variables (i.e., coordinates $x^{\text{rob}}(\kappa)$ and $y^{\text{rob}}(\kappa)$) of the robot within the prediction horizon, i.e., for $\kappa \in \{k^{\text{c}}, \ldots, k^{\text{c}}+N^{\text{p}}-1\}$. Thus, we have $\tilde{\boldsymbol{r}}^{\text{rob}}(k^{\text{c}}) = \{\boldsymbol{r}^{\text{rob}}(k^{\text{c}}+1), \ldots, \boldsymbol{r}^{\text{rob}}(k^{\text{c}}+N^{\text{p}})\}$. The optimization variable $\tilde{\boldsymbol{v}}(k^{\text{c}})$ includes the sequence of the nominal control input $\boldsymbol{v}(\kappa)$, which is the vector of the linear velocity $v^{\text{rob}}(\kappa)$ and the angular position $\theta^{\text{rob}}(\kappa)$ of the robot, estimated across the prediction window $\kappa \in \{k^{\text{c}}, \ldots, k^{\text{c}}+N^{\text{p}}-1\}$, i.e., we have $\tilde{\boldsymbol{v}}(k^{\text{c}}) = \{[v^{\text{rob}}(k^{\text{c}}), \theta^{\text{rob}}(k^{\text{c}})], \ldots, [v^{\text{rob}}(k^{\text{c}}+N^{\text{p}}-1), \theta^{\text{rob}}(k^{\text{c}}+N^{\text{p}}-1)]\}$.

Constraint (2.11b) predicts the evolution and transition of approximate areas A_i according to the mapping $\mathscr{M}^{\text{Evac}}$, which corresponds to the crowd evacuation model explained in Section 2.2.2. Constraint (2.11c) assesses per control time step whether or not the surface of the intersection area for C^{p} and A_i has reached a specific threshold (given as a percentage of A_i), with $|\cdot|$ giving the surface of a continuous 2-dimensional area and π^{c} a design/tuning parameter; whenever this condition is satisfied, the control system assumes that target victim i has been detected (or more precisely the robot should not search for this victim any more) and thus A_i is considered as a null set afterwards. Constraint (2.11d) describes the time evolution of the coordinates of the perception field $C^{\text{p}}(k)$ of the robot (i.e., the area that the sensors of the robot can cover, no matter whether or not parts of the area are obstructed by obstacles). Since the shape of the perception field always remains the same (i.e., circular), $\mathscr{T}(\cdot, \boldsymbol{\lambda})$ is a linear mapping that transitions a 2-dimensional area (or a set of coordinates) along the given 2-dimensional vector $\boldsymbol{\lambda}$. In our case, this vector is the linear velocity of the robot. Constraints (2.11e), (2.11f) and (2.11g) account for the physical constraints of the robot, with $\omega^{\text{rob,max}}$ the maximum angular velocity of the robot, $\Delta v^{\text{rob}}(k^{\text{c}}) = v^{\text{rob}}(k^{\text{c}}) - v^{\text{rob}}(k^{\text{c}}-1)$, and T^{c} the control sampling time (which, in general, is different from the simulation sampling time T^{s}), and τ a fixed constant. For the nominal controller, these constraints are tightened compared to those of the ancillary controller (see [217] for further information). Constraint (2.11h) describes the evolution of the states of the robot according to the nominal version of the model given by (2.1), i.e., when $\delta_x(\kappa)$ and $\delta_y(\kappa)$ are excluded. Constraint

(2.11i) formulates that the vector of nominal control input, i.e., the optimization variables of the nominal MPC, includes the linear velocity and the angular position of the robot. Finally, constraint (2.11j) allows the robot to move in the free and safe part of the SaR environment (i.e., avoids collision with obstacles). Note that $\mathscr{O}^{\mathrm{s}}$ and $\mathscr{O}^{\mathrm{d}}(k)$ include safety margins between the robot and obstacles and victims. If needed, constraint (2.11j) may be softened (but never relaxed) by adjusting these safety margins.

The problem is guaranteed to be recursively feasible as explained next. There are hard constraints on the states of the robot via (2.11j) and on the control inputs via (2.11f) and (2.11g). However, the initial configuration, based on Assumption A3, provides initial feasibility for the control problem, which implies recursive feasibility due to the following reasons: According to (2.11b), targets continuously move according to a physics-based state-space model. Thus, even if their position temporarily obstructs the exit of the robot, the exit will become accessible again after they move. Moreover, upper bound limits on the control inputs (i.e., speed of the robot) only increase the time of reaching the exit, but do not make the problem infeasible.

The objective function in (2.11a) is given by (2.12), that is composed by (2.13) and (2.14).

The objective function is composed of a stage cost and a terminal cost, where w_j for $j = 0,\dots,N^{\mathrm{v}}+3$ are weighting factors. In (2.12) the nominal economic cost has been formulated, which includes a target-oriented SaR term $J^{\mathrm{nom,target}}(\cdot)$ and a coverage-oriented SaR term $J^{\mathrm{nom,coverage}}(\cdot)$. The target-oriented objective function $J^{\mathrm{nom,target}}(\cdot)$ is expanded in (2.13), where (2.13a)-(2.13c) represent the target-oriented stage costs, and (2.13d) corresponds to the target-oriented terminal cost. Moreover, (2.14) expands the formulation of the coverage-oriented objective function. Since coverage is a stage property, this objective function is only composed of stage terms. In (2.13), the first term (2.13a) reduces the travel time of the robot from its initial position at control time step k^{c} to its target position at the end of the prediction horizon, i.e., at control time step $k^{\mathrm{c}}+N^{\mathrm{p}}-1$. With the second term (2.13b), the stage cost maximizes the intersection of the perception field $C^{\mathrm{p}}(\kappa)$ of the robot and every area $A_j(\kappa)$ where the victims are located, for $j = 1,\dots,N^{\mathrm{v}}$. The third term of the stage cost (2.13c) maximizes the intersection between $C^{\mathrm{p}}(\kappa)$ and the union of the areas $A_1(\kappa),\dots,A_{N^{\mathrm{v}}}(\kappa)$. Finally, the terminal cost (2.13d) - which should become highlighted by adapting $w_{N^{\mathrm{v}}+2}$ when the robot is close to the end of its battery life - assures that the distance between the robot and the exit is terminally minimized. The term in (2.14) minimizes the area of the intersection between $C^{\mathrm{p}}(\kappa+i)$ and $C^{\mathrm{p}}(\kappa),\dots,C^{\mathrm{p}}(\kappa+i-1)$, for $i = 1,\dots,N^{\mathrm{p}}$, and is used to increase the coverage of the area.

The optimization problem given by (2.11)-(2.14) is in general nonlinear and non-convex. While linearization of the problem and deploying quadratic optimizers may be considered, we instead use a well-established nonlinear tube-based approach [217], due to the following reasons: (1) Avoid introducing further model mismatches that may impact the recursive feasibility. (2) Prevent an increase in the computation time, which occurs when the linearization is not efficient. (3) Avoid significant decrease in the precision by oversimplifying the model, where imprecision negatively impacts both the performance and the recursive feasibility. Note that based on Assumption A3 the MPC problem (2.11)-(2.14) is feasible by nature.

Regarding the computation time of the approach, our focus is on using MPC for op-

timizing competing costs and handling hard constraints, and we analyze the MPC to showcase its performance, therefore avoiding to aim for faster computations. To make sure that the problem does not become intractable, we first solve the optimization and then provide the control action to the robot, that performs its motion. Since these two phases are separated, we always allocate enough time to execute the computations required by the optimizations, therefore the problem is always tractable. When this is not possible, for example in real life, the computation times may be reduced by using fast MPC methods (see, e.g., [218], [219]).

DEALING WITH UNCERTAINTIES VIA ROBUST TUBE-BASED MPC

In order to deal with uncertainties in the environment, we implement a robust tube-based version [220] of the nominal MPC proposed in Section 2.2.2. Note that tube-based MPC was originally formulated for linear systems (see [220]), but extensions to nonlinear systems exist and are established in the literature (see, e.g., [217]), where two controllers are employed, one nominal (that does not consider uncertainties) and one ancillary (that considers the uncertainties and is used to steer the nonlinear system close to the nominal trajectory).

Therefore, consider the following nonlinear system subject to bounded additive disturbances $\boldsymbol{w}(k) \in \mathbb{W}$, e.g., the displacement of the robot position considered as non-smoothness of the floor (compare with (2.1) where $[\delta_x(k^s), \delta_y(k^s)]^\top$ is the external disturbance vector):

$$\boldsymbol{\chi}(k+1) = f(\boldsymbol{\chi}(k), \boldsymbol{u}(k)) + \boldsymbol{w}(k) \tag{2.15}$$

The following ancillary controller will be formulated to obtain a feedback control law in the nonlinear case [217], thus this is the objective function:

$$J^{\text{anc}^*}(\tilde{\boldsymbol{\chi}}(k), \tilde{\boldsymbol{z}}(k)) = \min_{\tilde{\boldsymbol{u}}(k)} J^{\text{anc}}(\tilde{\boldsymbol{\chi}}(k) - \tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{v}}(k)) \tag{2.16}$$

where the constraints are as in (2.11), and this is the control input:

$$\tilde{\boldsymbol{u}}^*(k) \in \arg\min_{\tilde{\boldsymbol{u}}(k)} J^{\text{anc}}(\tilde{\boldsymbol{\chi}}(k) - \tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{v}}(k)) \tag{2.17}$$

where the symbol $\tilde{\cdot}$ is used with a variable to show the sequence of that variable within the prediction horizon and $\boldsymbol{z}(k)$ and $\boldsymbol{v}(k)$ correspond to the nominal system, i.e.:

$$\boldsymbol{z}(k+1) = f(\boldsymbol{z}(k), \boldsymbol{v}(k)) \tag{2.18}$$

The ancillary control input will be added to the nominal input $\boldsymbol{v}(k)$ to control system (2.15). In our case, the disturbances correspond to $\delta_x(k^s)$ and $\delta_y(k^s)$ in the model for the motion of the robot, i.e., in (2.1). These deviations may occur due to, e.g., the non-smooth ground on which the robot moves.

Thus, in more detail, our proposed approach is integrated into this formulation by employing the objective function (2.12) for the nominal controller, and the objective function (2.16), that optimizes the difference between the states and the control actions of the two systems, for the ancillary controller. In both controllers, the system is the nonlinear unicycle model (2.1).

Algorithm 1 illustrates how robust tube-based MPC for nonlinear systems can be implemented.

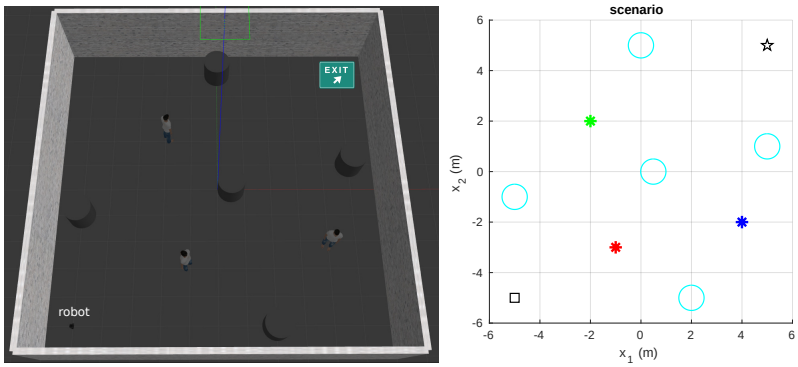
Algorithm 1 Robust tube-based MPC

N_{iter}: Number of iterations
f: nonlinear system model
w_d: additive disturbance

$z_0 \leftarrow$ robot_initial_state
$\chi_0 \leftarrow$ robot_initial_state
$z_{\text{goal}} \leftarrow 1$
$v(1:N^{\text{p}}) \leftarrow 0(1:N^{\text{p}})$
$u(1:N^{\text{p}}) \leftarrow 0(1:N^{\text{p}})$
for $i \leftarrow 1$ **to** N_{iter} **do**
    $v_{\text{new}}(1:N^{\text{p}}) = \text{nominal\_control\_problem}(N^{\text{p}}, N^{\text{v}}, z_0, v(1:N^{\text{p}}), w(1:N^{\text{v}}+3), z_{\text{goal}})$
    $u_{\text{new}}(1:N^{\text{p}}) = \text{ancillary\_control\_problem}(N^{\text{p}}, w_d, \chi_0, z_0, u(1:N^{\text{p}}), v_{\text{new}}(1:N^{\text{p}}))$
    apply_control_action($u_{\text{new}}(1)$)
    $z_0 \leftarrow$ robot_state_measurement
    $z_{\text{goal}} \leftarrow$ current_target_victim
    $v(1:N^{\text{p}}) \leftarrow v_{\text{new}}(1:N^{\text{p}})$
    $u(1:N^{\text{p}}) \leftarrow u_{\text{new}}(1:N^{\text{p}})$
end for

Function J = nominal_control_problem(N^{p}, N^{v}, z_0, $v(1:N^{\text{p}})$, $w(1:N^{\text{v}}+3)$, z_{goal})
$J \leftarrow 0$
$z(1) \leftarrow z_0$
for $k \leftarrow 1$ **to** N^{p} **do**
    $z(k+1) \leftarrow f(z(k), v(k))$
end for
area_tot_amount = compute_intersection_of_unions_amount()
for $h \leftarrow 1$ **to** N^{v} **do**
    area_amount(h) = compute_intersection_amount(h)
end for
intersection_area_amount = compute_intersection_amount()
$J \leftarrow J + w_0\text{sum}(v(1:N^{\text{p}})) + w_1\text{area\_amount}(1) + \ldots + w_N^{\text{v}}\text{area\_amount}(N^{\text{v}}) + w_{N^{\text{v}}+1}\text{area\_tot\_amount} + w_{N^{\text{v}}+2}\text{norm}(z(N^{\text{p}}) - z_{\text{goal}}) + w_{N^{\text{v}}+3}\text{intersection\_area\_amount}$

Function J = ancillary_control_problem(N^{p}, w_d, χ_0, z_0, $u(1:N^{\text{p}})$, $v_{\text{new}}(1:N^{\text{p}})$)
$J \leftarrow 0$
$z(1) \leftarrow z_0$
for $k \leftarrow 1$ **to** N^{p} **do**
    $z(k+1) \leftarrow f(z(k), v_{\text{new}}(k))$
end for
$\chi(1) \leftarrow \chi_0$
for $k \leftarrow 1$ **to** N^{p} **do**
    $\chi(k+1) \leftarrow f(\chi(k), u(k)) + w_d$
end for
for $k \leftarrow 1$ **to** N^{p} **do**
    $J \leftarrow J + (\chi(k) - z(k))^2 + (u(k) - v_{\text{new}}(k))^2$
end for



(a) (b)

Figure 2.4: Schematic view of the simulations in (a) Gazebo and in (b) MATLAB with the three victims (shown via red, green, and blue asterisks), the cylindrical static obstacle (shown via cyan circles), the start point of the robot (shown via a black square) and the exit point (shown via a black star).

2.3. CASE STUDY

Next we explain the case study, we present the results, and discuss them.

2.3.1. SETUP

The models and the algorithms in this case study have been implemented in MATLAB (R2021a version), ROS (Melodic version) and Gazebo (9.0 version). The victims movement model has been implemented and simulated in, respectively, ROS and Gazebo. The robot simulated for this case study is a TurtleBot 3 Burger [221], which has open source ROS and Gazebo models [222]. The robust tube-based MPC controller has been implemented in MATLAB, which was connected to the robot model in Gazebo in order to simulate it. The PC, on which the experiments have been simulated, had Intel Core i7 processor with 4 cores at 1.80 GHz-2.30 GHz and 16 GB RAM.

We consider different scenarios for different simulation types, i.e., (1) the main scenario where we compare our approach against other state-of-the-art approaches, (2) the scenario for showing the performance of the increased area coverage, (3) the scenario for showing the performance in highly non-convex environments. Therefore, for each of these simulation types we have 1 scenario, and 3 simulations in total. After that, we perform experiments in real life, and for that we consider 3 different scenarios.

In the first scenario, the environment of the robot is a partially collapsed building with obstacles (see Section 2.2.1 for details) that has been simulated in Gazebo. In such scenario, we have a 12 m by 12 m square room, where there are three victims and five static obstacles. While in the formulation we assume obstacles with any shape, in this case study we consider them to be cylinders, with cross sections of radius 0.5 m and their centers positioned at (0.49, 0), (-5, -1), (0, 5), (5, 1) and (2, -5) (see Figures 2.4). We suppose that the victims start at positions (-1, -3), (-2, 2) and (4, -2), while the robot only knows an estimate initial area that each victim is positioned in, where these areas are circles of radius 2 m centered around the real positions of the victims. We con-

sider a uniform distribution for the uncertainties about the positions of the victims. This choice is based on [146], which introduces the evacuation model we have adopted for the MPC formulation in this chapter. This choice may differ for other implementations, by sampling the initial area $A_i(0)$ of the victims according to another distribution (e.g., Gaussian). The ratio between the area of the environment and the perception field of the robot is 6, therefore the area coverage is substantially larger than the perception range of the robot. The robot starts its mission at coordinates $(-5,-5)$ and ends it at the exit point with coordinates $(5,5)$. The bounded uncertainties corresponding to the mismatch of the kinematics model of the robot and the reality are selected to satisfy $(\delta_x,\delta_y) \in [-0.1\text{ m}, 0.1\text{ m}] \times [-0.1\text{ m}, 0.1\text{ m}]$ in a grid of 12 by 12 cells. We consider the values of the uncertainties to remain constant, i.e., every cell corresponds to a fixed uncertainty in the given range. The resulting grid may be interpreted as the non-smoothness map of the environment (the tables are publicly available in the 4TU.ResearchData repository [223]).

The optimization problem given by (2.11)-(2.14) for the robot controller has been solved in MATLAB using `fmincon` solver with `sqp` algorithm. The constrained optimization problem of MPC in our (or various other) SaR scenarios is in general non-convex. For non-convex optimization problems, the risk of falling in local optima exists, especially when a gradient-based algorithm is used to solve the optimization problem. Moreover, while in theory global optimization methods (e.g., genetic algorithm) find the global optima, in practice due to the limited time/iterations allocated to solving an optimization problem, such methods may also fail to find global optima. To address this issue, multi-start methods may be used, where several candidate starting values are considered for the optimization variables and the problem is solved for all these starting values. This way, by a proper choice of the starting values, the potential set of solutions is more extensively explored.

In our case studies, we have used a gradient-based approach to solve the optimization problem, because in general gradient-based solvers are faster than most other methods. Multiple starting points were considered per optimization problem, and the realized values of the objective function after solving the optimization problem for all these starting points were compared. The solution that corresponded to the least realized objective value (supposing that a minimization problem is solved) was considered as the solution of MPC. This approach has been used extensively in literature (see, e.g., [224], [225]).

The prediction horizon N^{p} of the MPC controller is 10, with a sampling time of $T^{\text{s}} = 0.2$ s. We assume that the simulation time step k^{s} and the control time step k^{c} are synchronized, thus we use time step k for both. The maximum linear and angular velocities are, respectively, $v^{\text{rob,max}} = 0.7$ m/s and $\omega^{\text{rob,max}} = 2.5$ rad/s. Moreover, we have $\Delta v^{\text{rob,max}} = v^{\text{rob,max}}/2$ and $\Delta\theta^{\text{rob}} = \pi/2 + 0.1$ rad. In our case studies, the weights (which are considered to remain constant during the entire SaR mission) have been selected at the beginning using a trial-and-error approach, such that the selected values are in a meaningful range considering the SaR scenarios that will be simulated, i.e., the candidate initial positions of the victims, the estimated location of the obstacles, and the coordinates of the exit point of the robot.

The trade-off parameter π^{c} for constraint (2.11c) was tuned via trial-and-error to 0.6:

On the one hand, for smaller values of this parameter, it was observed that for various cases the robot over-trusted, based on the estimations of its prediction model, that the target victim will be detected, whereas in reality the robot was missing the victim. This was due to the limited exploration in the potential areas of the target victims by the robot due to a too small value for π^{c}. For larger values for parameter π^{c}, on the other hand, the robot, based on the estimations of its prediction model, would non-efficiently spend extra time exploring the potential areas of the target victims, whereas in reality the target victims were already detected.

The weights in (2.14) are put to zero first, because we will consider the influence of the additional term in the objective function for the area coverage separately in Section 2.3.4. The values of the other parameters used in the simulation for forces and torques of the model for the movement of the victims are taken from [146]. Moreover, in the case study we consider the following additional terminal cost in (2.12):

$$
\begin{aligned}
& w_{N^{\mathrm{v}}+4} \left\| \left[x^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}), y^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}) \right]^{\top} - \left[x^{\mathrm{cg}_1}, y^{\mathrm{cg}_1} \right]^{\top} \right\| + \\
& \quad \ldots + \\
& w_{2N^{\mathrm{v}}+4} \left\| \left[x^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}), y^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}) \right]^{\top} - \left[x^{\mathrm{cg}_{N^{\mathrm{v}}}}, y^{\mathrm{cg}_{N^{\mathrm{v}}}} \right]^{\top} \right\| + \\
& w_{2N^{\mathrm{v}}+5} \left\| \left[x^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}), y^{\mathrm{rob}}(k^{\mathrm{c}}+N^{\mathrm{p}}) \right]^{\top} - \left[x^{\mathrm{cg}_{\mathrm{tot}}}, y^{\mathrm{cg}_{\mathrm{tot}}} \right]^{\top} \right\|
\end{aligned} \tag{2.19}
$$

This additional term is used to minimize the distance between the robot and the centers of gravity $\left[x^{\mathrm{cg}_1}, y^{\mathrm{cg}_1}\right]^{\top}, \ldots, \left[x^{\mathrm{cg}_{N^{\mathrm{v}}}}, y^{\mathrm{cg}_{N^{\mathrm{v}}}}\right]^{\top}$ of the victim shapes $A_1, \ldots, A_{N^{\mathrm{v}}}$, and the average center of gravity $\left[x^{\mathrm{cg}_{\mathrm{tot}}}, y^{\mathrm{cg}_{\mathrm{tot}}}\right]^{\top}$ of all these centers of gravity. The reason for adding this term is the following: In theory, with an infinite prediction horizon, for every time step all victims will fall within the prediction horizon of the robot, thus the robust tube-based MPC will be able to compute the intersection with all areas $A_1, \ldots, A_{N^{\mathrm{v}}}$ (see the $\left|C^{\mathrm{p}}(\kappa) \cap A_i(\kappa)\right|^{-1}$ term, for $i = 1, \ldots, N^{\mathrm{v}}$, in (2.13)). In practice, however, an infinite (or large enough) prediction horizon may make the computations intractable. Thus the term given by (2.19) is used so that we can select a reasonably small value for N^{p} and reduce the computation time significantly.

The weights for various terms of the objective function were also tuned in advance and by trial and error. We have chosen manual tuning instead of systematic hyperparameter tuning to obtain interpretability, since we can better understand how each weight influences the behavior of the controller, and to allow flexibility for using them in different scenarios. Therefore, the weights were tuned as it will be explained next. For the target-oriented objective function, the parameters $w_1, \ldots, w_{N^{\mathrm{v}}}$ in (2.13b) that weigh the intersection of the area corresponding to the perception field of the robot and the area corresponding to each target victim were tuned to 10^5, whereas $w_{N^{\mathrm{v}}+1}$ in (2.13c), i.e., the weight of the term that defines the intersection of the area corresponding to the perception field of the robot and the union of all areas of the target victims was tuned to 10^8. With lower ranges for $w_1, \ldots, w_{N^{\mathrm{v}}}$, the robot was not attracted sufficiently by the individual areas corresponding to the target victims and was mainly moving towards the center of the union area without detecting any individual victims. On the contrary, for much larger ranges for $w_1, \ldots, w_{N^{\mathrm{v}}}$, the robot was purely concentrating on individual target

victims, losing the global picture of other target victims. This could result in solutions that were globally far from optimum.

In addition, w_0 is put to 10^{-2}, in order to give priority to finding the victims rather than reducing the mission time, that for the case study is already quite low. Next, w_{N^v+2} is put to 10^3, so that the robot has a reduced priority for going to the exit while it is searching for victims, i.e., compared to $w_1,\dots,w_{N^v}$ and w_{N^v+1}, and higher priority after these terms are no longer relevant because the victims are already found. Finally, for the area coverage, the weight w_{N^v+3} is put to 10^3. Even if lower compared to $w_1,\dots,w_{N^v}$ and w_{N^v+1}, this shows a sufficiently good exploration performance when the term is activated.

The weights of the variation of the terminal state of the robot with respect to the individual centers of gravity in (2.19), i.e., $w_{N^v+4},\dots,w_{2N^v+4}$, are set to 10^7 and the weight of the variation of the terminal state of the robot with respect to the center of gravity of the union of the shapes, i.e., w_{2N^v+5}, is set to 10^4. These values showed, within the range of the parameters and dynamics of the case studies, that a balanced trade-off will be achieved by the robot, to target the individual victims, while considering not to get too far from the rest of the victims.

2.3.2. COMPARISON

The proposed robust tube-based approach has been compared in simulation with four state-of-the-art methods, from which the first (Farrokhsiar tube-based MPC) and second (A* MPC) are target-oriented and the third (randomized MPC) and the fourth (boustrophedon motion A*) are coverage-oriented.

The first approach that we compare our results with is the one given in [192]. There, a robust tube-based MPC controller is used to determine a path for the robot that leads to a specific target point. This controller, as in the approach that we propose, is composed of a nominal and an ancillary control, and the uncertainty in the kinematics model of the robot is also the same. However, unlike our robust tube-based MPC controller, the controller in [192] does not have the victims movement model in its constraints, thus the objective function includes a term that minimizes the distance of the robot to the initial positions of the target victims. Moreover, there is no coverage-oriented term in the objective function. This approach is representative of robust methods in literature based on MPC for target-oriented path planning.

The second approach is the one in [226], which uses the A* algorithm to determine a shortest path to a specific target point. An MPC controller is then used to track this path. In order to account for the partial information that is available from the environment for the robot, similarly to [193] at every time step we define an intermediate target point on the border of the perception field of the robot that is the closest to the current target (i.e., a victim or the exit) of the robot. Whenever the robot reaches this intermediate target, the A* algorithm determines a new path, and this procedure continues until the target is reached. This approach is representative of A*-based methods that are commonly found among standard target-oriented path planning.

The third approach is the one in [227], in which a randomized MPC algorithm is used for path planning. In particular, a number of random samples in the 2D space are generated, and then an algorithm similar to a rapidly exploring random trees (RRT)

[228] is used to determine a shortest path that extends through these random points. In this algorithm, the targets are the random points and then the exit. If a random point falls within the forbidden areas, it is discarded since we constrain the algorithm from the beginning. The MPC is used in the planner to generate a collision-free control trajectory. Due to the random selection of the points in the environment, the approach increases the coverage of the area with respect to purely target-oriented approaches. This approach is representative of random-search-based methods, often used as baseline for coverage-oriented path planning. In addition, as a random search strategy, this approach is used as the lower bound benchmark in our experiments.

The fourth approach is the method in [229]. This paper uses the boustrophedon motion algorithm, which mimics the back-and-forth motion of an ox when plowing a field [230], to explore an area as in coverage path planning; when the robot is close to obstacles and cannot continue its current motion, the algorithm determines backtracking points in the visited path and the robot returns to them. Finally, an A* algorithm is used to determine a shortest path to the nearest backtracking point with respect to the current position of the robot; there, the robot starts a new boustrophedon motion, and this procedure is repeated until there are no more unvisited backtracking points. This approach is used as a representative of the heuristic methods that are used in standard coverage path planning.

Finally, we also consider an upper bound benchmark, i.e., an ideal scenario where the robot has perfect knowledge of the environment and the positions of the victims. We implement this benchmark by providing our proposed MPC-based approach with the exact locations of the victims as they appear in the Gazebo simulator.

In all the cases given above, MPC has been used to steer the robot to track the reference path. In addition, for all cases the simulation ends when the robot reaches the exit point.

2.3.3. RESULTS

In this section, we present the results of the simulations. Figure 2.5 shows the trajectories of the robot and the 3 victims in the environment, for a case study. In these figures, the trajectory of the robot is shown in black and for the 3 victims the trajectories are shown in red, green, and blue. The following symbols with their corresponding meanings have been used in these figures: Small square, showing the starting positions of the victims and the robot; Star, showing the final position of the victims and the exit point for the robot; Asterisk, showing the position of a victim at the time of being detected by the robot. Whenever the robot detects a victim, the perception field of the robot has also been illustrated. The corresponding videos are publicly available in the 4TU.ResearchData repository [223].

Figure 2.6 shows the number of victims detected by the robot in time for all the control approaches. Figure 2.7 presents the percentage of the area that is covered by the perception field of the robot with respect to time during the entire simulation. Figure 2.8 illustrates via a heat map the number of time steps for which the robot visits the same point of the environment.

Finally, in Table 2.2 we show the computational cost of the algorithm iterations for all the six approaches of the comparison.

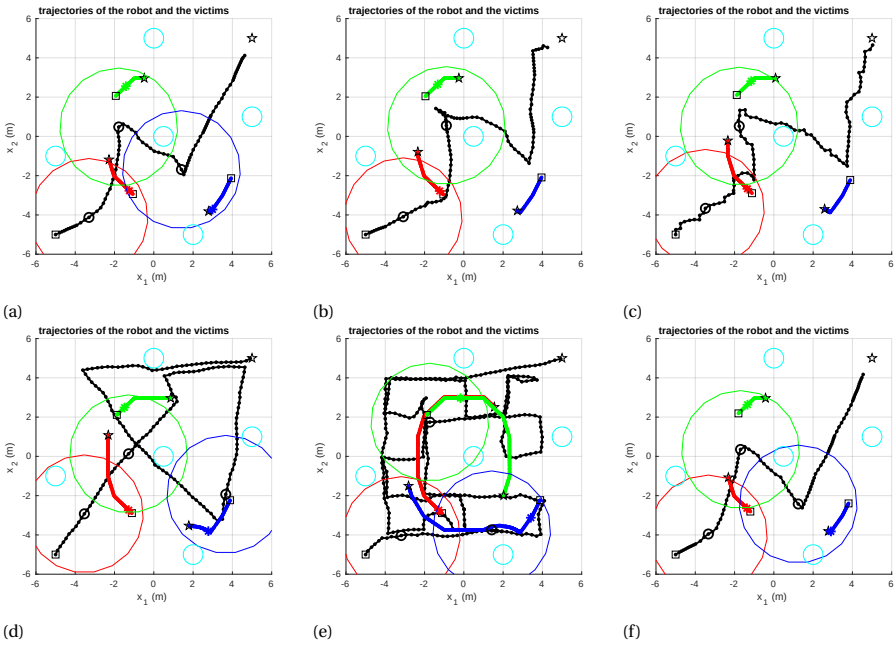


Figure 2.5: Plot of the trajectories of the robot (in black) and the three victims (in red, green, and blue) in the environment, with the obstacles shown by small circles (in cyan): Whenever the robot detects a victim, this is illustrated with an asterisk on the trajectory of that victim, as well as by showing the circular perception field of the robot (in the color of the corresponding victim) at the time of detecting the victim. The black squares indicate the starting positions of the victims and the robot, while the black stars illustrate the final position of the victims and the exit point for the robot. Algorithms: (a) Our approach, (b) [192], (c) [226], (d) [227], (e) [229], (f) our approach with perfect information.

2.3.4. DISCUSSION

Next, in Section 2.3.4 we discuss and compare the outcome of the different control approaches based on the results given in Section 2.3.3. After that, in Section 2.3.4 we assess and discuss the performance of the proposed robust tube-based MPC controller, when the objective function includes the area coverage term given in (2.14), for a special scenario that properly showcases the importance of the area coverage in SaR missions. Moreover, in Section 2.3.4 we describe and discuss a simulation realized in a special scenario that highlights the non-convexity of the space, in order to show that our approach can be also applied to such non-convex scenarios, despite the risk for the MPC optimization to fall in local minima. Finally, in Section 2.3.4 we present and discuss the results of real-life experiments performed using real robots, comparing our approach to [226] and [227], and we include a discussion on the performance of our tube-based MPC approach compared to two decoupled path planning and trajectory tracking controllers.

MAIN SCENARIO

From Figure 2.5, our proposed robust tube-based MPC approach guarantees a safe margin in avoiding the obstacles, whereas other control methods (except for [192] which

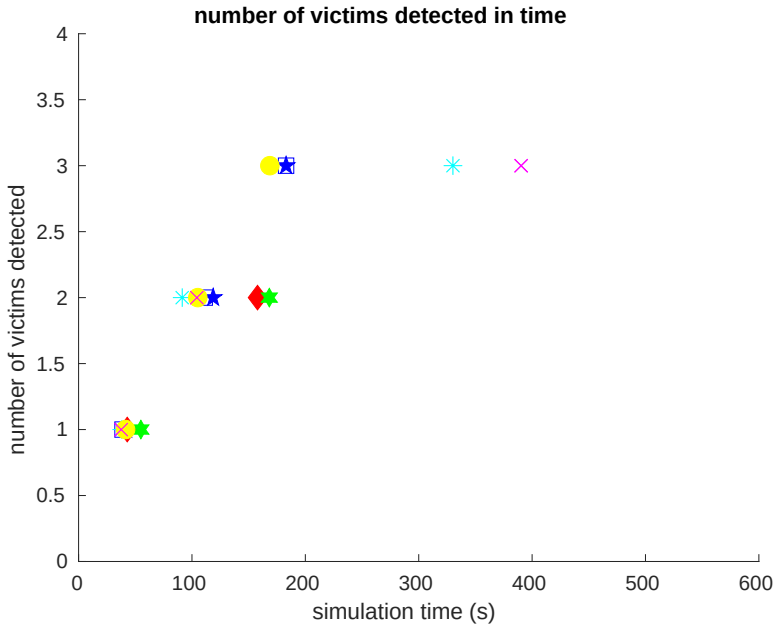


Figure 2.6: The number of victims found by the robot: our approach (blue five-pointed star), [192] (red diamond), [226] (green six-pointed star), [227] (cyan asterisk), [229] (magenta cross), and our approach with perfect information (yellow circle). These markers indicate that the victim is inside the perception field (in Gazebo, the robot has detected the real location of the victim); for our approach, the blue square indicates that the victim is detected by the robot (the controller in MATLAB, through the intersection of the areas).

also uses a robust MPC method) result in trajectories that are prone to the risk of collision with the obstacles. In fact, although avoiding the obstacles has been formulated as a constraint for all MPC controllers, due to the absence of a systematic robustness (which is provided by the robust tube-based approach) the robot may crash into an obstacle because of the non-smoothness uncertainty (this occurs, e.g., in Figure 2.5 (c) and (d)).

Based on Figure 2.5, we observe that there is a trade-off between the time to complete the mission and the number of the victims detected. In fact, the robot is able to find more victims with coverage-oriented approaches (see cases (d) and (e)), compared to target-oriented ones (see cases (b) and (c)). This is because the robot normally explores a larger area with coverage-oriented methods in a given time, while with target-oriented meth-

Approach	1	2	3	4	5	6
Computational cost (mean)	81.70	65.43	131.13	331.51	220.29	58.51
Computational cost (std)	23.13	14.38	39.67	288.37	112.60	19.09

Table 2.2: Computational cost in seconds, for the average algorithm iteration, and corresponding standard deviation, of the approaches of the comparison: (1) Our tube-based MPC, (2) Farrokhsiar tube-based MPC [192], (3) A* MPC [226], (4) Randomized MPC [227], (5) Boustrophedon motion A* [229], and (6) our approach with perfect information.

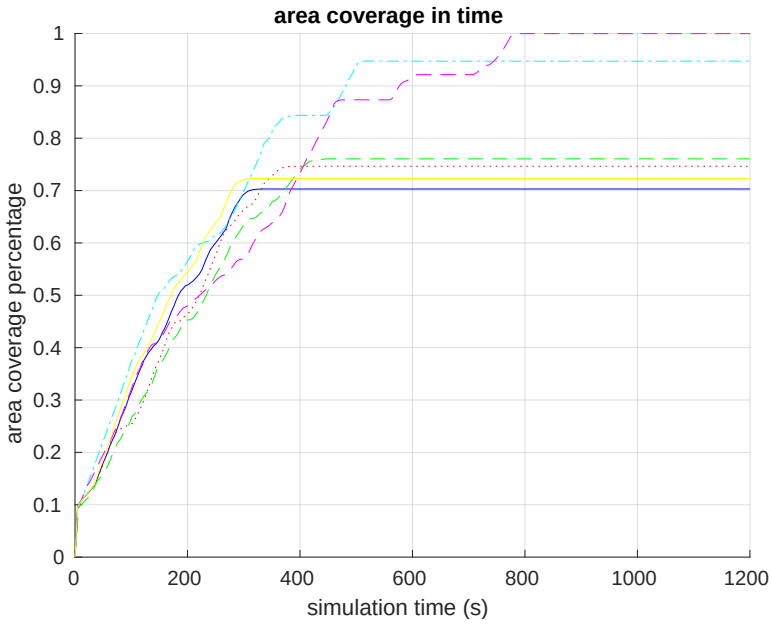


Figure 2.7: The percentage of environment area covered in time by our approach (blue solid line), [192] (red dotted line), [226] (green dashed line), [227] (cyan dash-dotted line), [229] (magenta dashed line), and our approach with perfect information (yellow solid line).

ods the mission is accomplished more quickly, after the specific targets are approached. In addition, in the coverage-oriented approaches the trajectories of the robot are more complex, because the robot should avoid the obstacles and the moving victims more times during the mission. Our proposed robust tube-based MPC method (see case (a) in Figure 2.5) performs as well as the coverage-oriented methods in detecting the victims, while resulting in a simpler trajectory and finishing the mission faster. These results are confirmed also via Figure 2.6. In general, with the target-oriented approaches (see the red and green symbols) the robot is not able to find and reach all the victims. This is because the robot encounters victims by chance, while moving towards their initial positions (note that in these cases the robot does not track the victims via their movement model). As expected, with the coverage-oriented approaches (see the blue, cyan and magenta symbols), the robot visits a larger percentage of the environment, and therefore it is likely to find more victims than with the target-oriented methods. Moreover, with our proposed method the robot outperforms the target-oriented approaches in victim detection (see the red, green and blue symbols), and all target-oriented and coverage-oriented methods in speed for finding the victims (i.e., using our approach all victims are detected by time 183.0 s). Only the upper-bound benchmark approach (see the yellow symbol in Figure 2.6) is faster than our method, since it has perfect knowledge of the locations of the victims and therefore, can reach them more quickly. Furthermore, in our approach the prediction of the robust tube-based MPC about detecting the victims is closely aligned with real detection of the victims (compare the blue square and blue

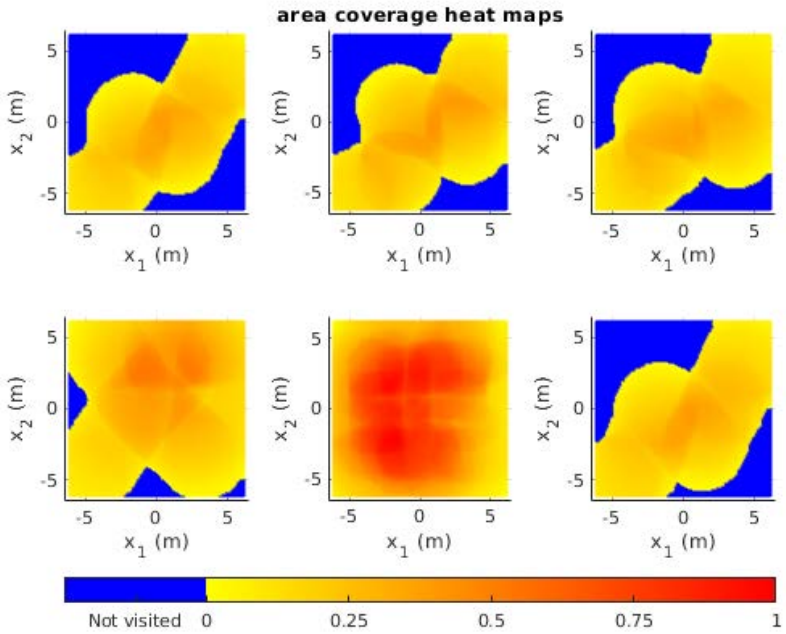


Figure 2.8: Heat maps illustrating the intensity of the area coverage, i.e., whenever the robot visits a point for more time steps, the intensity of the heat map varies from 0 to its maximum 1. From top left to bottom right: Our approach, [192], [226], [227], [229], and our approach with perfect information.

star in Figure 2.6). This implies that the trade-off parameter π^c has properly been tuned.

From Figure 2.7 we can compare the area coverage performance of different control approaches. At the beginning of the simulation, all the methods perform similarly. As it is expected, the coverage-oriented approaches (see the cyan and magenta curves) outperform the target-oriented approaches (see the red and green curves) in the long run: In fact, the target-oriented approaches reach a maximum of around 80% area coverage, whereas both coverage-oriented methods reach almost 100% coverage. In the first 400 time steps, our method (see the blue curve) is moderately faster in covering the area compared to the target-oriented approaches. The robot immediately moves towards the exits after detecting all the target victims, thus the area coverage does not increase afterwards. By including the additional coverage-oriented term (see (2.14)) in the objective function, the area coverage will continue to increase also after detecting the target victims. From the two coverage-oriented approaches, the randomized MPC method [227] outperforms the boustrophedon motion A* method [229] regarding the mission time. This is because the boustrophedon motion coverage algorithm requires the robot to visit the area little by little, whereas random points have a higher chance of falling within the unexplored areas of the environment. In fact, the boustrophedon motion A* approach [229] is the only one that reaches 100% area coverage, thanks to its algorithm that is particularly designed for systematic coverage of the area.

	Our approach without coverage term	Our approach with coverage term
Computational cost (mean)	6.01	32.21
Computational cost (std)	10.91	87.02

Table 2.3: Computational cost in seconds, for the average MPC minimization, and corresponding standard deviation, of our proposed approach with and without the additional coverage term.

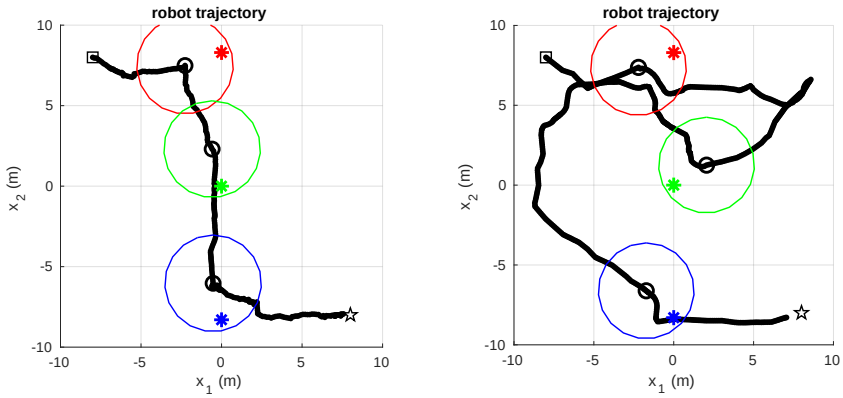
From Figure 2.8, via coverage-oriented approaches the robot keeps revisiting some areas of the environment (see the two heat maps at the bottom of the figure), while in our approach and in the target-oriented approaches the robot visits the area once (see the three heat maps at the top of the figure). On the one hand, the behaviour of the coverage-oriented approaches can be preferable in presence of dynamic obstacles or dynamic targets. On the other hand, a trade-off between the time efficiency and the additional information gained by revisiting the areas should be considered.

Comparing the various approaches with the performance of the upper bound benchmark, we show that our approach is the closest to it since it is the fastest in covering new areas (see Figure 2.7), due to the combination of target and coverage-oriented objectives, and in detecting the victims (see Figure 2.6), since we include an accurate prediction model for the movement of the victims.

For the computation times, Table 2.2 shows that our approach outperforms A* method [226] and both coverage-oriented approaches. Only Farrokhsiar tube-based MPC [192] and the upper-bound benchmark method are faster than our approach, because unlike our MPC controller, they do not consider the evolution of the 2D areas assigned to each victim in the optimization loop. Since our main focus is on using MPC for optimizing competing costs and handling hard constraints, we analyze the MPC in comparison with the other approaches to showcase its performance, and therefore avoiding to aim for faster computations. The reported results are impacted by the usage of laptops with limited resources that run micro-simulations. In real life, instead, more advanced hardware can be used and micro-simulators are not needed. In addition, using fast MPC methods (see, e.g., [218], [219] about fast MPC alternative methods), the computation time could be significantly improved. There already exists vast research that discusses improvement of computational efficiency (see, e.g., [231], [232]) and analysis of feasibility of MPC (see, e.g., [233], [234]). Such methods may be used for the MPC formulation that is proposed in this chapter, in real-life applications to combined coverage and target-oriented SaR, where fast computation and feasibility should be guaranteed.

SPECIAL SCENARIO FOR AREA COVERAGE TERM

In order to assess the effect of adding the additional area coverage term (i.e., putting the weights of (2.14) to non-zero) in the objective function (2.12), we have simulated an additional specific scenario, in which the robot has to visit three static victims before going to the exit (see Figure 2.9 where the position of the victims are shown by colored asterisks). In this scenario, we have a 20 m by 20 m square room, where there are three static victims with initial positions $(0, 8.3)$, $(0, 0)$ and $(0, -8.3)$ and no obstacles. The robot starts its mission at coordinates $(-8, 8)$ and ends at the exit point at coordinates $(8, -8)$. This configuration has been selected because after reaching a certain victim, the robot



(a) (b)

Figure 2.9: The robot trajectory (in black) with the position of the three victims (red, green, and blue asterisks); whenever the robot detects a victim, the circular perception field of the robot is illustrated in the color of the detected victim. (a) Proposed robust tube-based MPC approach without the area coverage term. (b) Proposed robust tube-based MPC approach with the area coverage term.

Iteration number	10	20	35
Cost function values	**0.0013**, 0.77, 0.24, 1.15, 0.29	**-0.0037**, 0.98, 0.36, 1.15, 0.32	**-0.0066**, 0.28, 0.097, 0.58, 0.28

Table 2.4: Values of the objective function of the MPC in the nonconvex scenario for three indicated iterations, and the points that are selected as minima are shown in bold.

has additional space, especially at the left and at the right of the victims, that it can explore. In fact, with the additional coverage term, the robot continues on its direction after one victim is detected, to explore more area, until the border of the scenario is reached, where the coverage term is deactivated and therefore the robot goes to the following victim. In particular, from Figure 2.10 we can show that the additional term gives a gain in coverage percentage of 29.3%. Finally, in Table 2.3 we show the computational cost of the MPC minimizations for our approach with and without the additional area coverage term: Without the coverage term, the MPC is more then five times faster, and therefore there is a trade-off in selecting computational performance and area coverage amount. Note that the computation times are not comparable with respect to those in Table 2.2 because we do not employ multi-start in this optimizations.

SPECIAL SCENARIO FOR ADDRESSING THE NON-CONVEXITY

We have designed a scenario, where due to the configuration of the obstacles (as it is shown in Figure 2.11), the resulting constrained optimization problem is obviously non-convex. This scenario considers a $6\times6\,\mathrm{m}^2$ room, where two victims are initially located at $(-1,1)$ and $(1.5,-0.5)$, and various obstacles create non-convex shapes. The robot starts its mission from position $(-2.5,-2.5)$ and ends the mission at the exit point located at $(2.5,2.5)$.

The optimization problem was run for 5 different starting points per iteration, where one starting point is the previous solution, while the other four starting points are se-

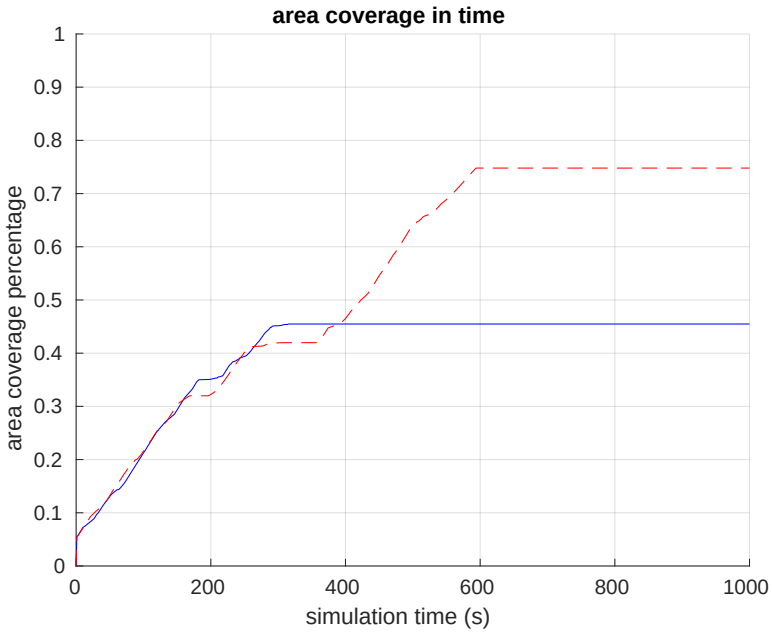


Figure 2.10: Percentage of the area covered in the special scenario: Proposed robust tube-based MPC approach without the area coverage term (blue solid line) and proposed robust tube-based MPC approach with the area coverage term (red dashed line).

lected randomly. As Figure 2.11 shows, despite a highly non-convex space, the robot executes its mission by successfully detecting both victims, and by reaching the exit afterwards, while avoiding the obstacle traps. Moreover, Table 2.4 shows the realized values of the objective function for these 5 optimization starting points, for 3 randomly selected optimization iterations. The solution corresponding to the value in bold fonts (i.e., the least of the 5 values) is selected per iteration.

REAL-LIFE EXPERIMENTS

In order to show the applicability of our approach beyond simulations, especially when real-life uncertainties, disturbances, and measurement noise exists, we designed and performed experiments in real life, using the same software (MATLAB, ROS) and laptop, as described in Section 2.3.1. We used a TurtleBot 3 Burger [221] to represent a SaR robot, and additional robots, i.e., iRobot Create 3 [235], to move according to the victims motion model and to represent the victims. The movements of the iRobot Create are aligned with the movements of the victims. More specifically, the iRobot Create first receives commands on the rotation angle, and then on the linear velocity, where these values are corresponding to those computed with the victims motion model. The environment and the robots are shown in Figure 2.12.

Three scenarios were simulated in a room of size 3.5×5.2 m^2, with three obstacles at locations (0, 0), (0.5, 2) and (-0.5, -2), and a victim with a different initial position per scenario, i.e., (1.5, -0.5) for scenario 1, (-1, 1) for scenario 2, and (1.25, 0.75) for scenario 3,

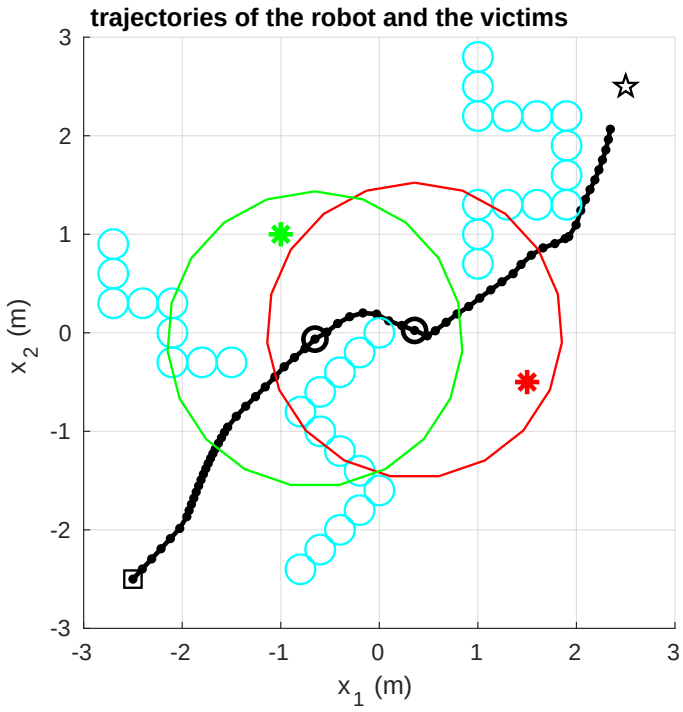


Figure 2.11: A highly non-convex SaR scenario: The robot trajectory is shown in black and the real positions of the two victims that are determined via the micro-simulator are shown with a red and a green asterisk. Whenever the robot detects a victim, the circular perception field of the robot has been illustrated in the same color used for illustrating the position of the detected victim.

with a circular uncertain positioning area of radius 0.5 m. The robot started its mission at coordinates $(-1.25, -2.1)$ and ended the mission at the exit with coordinates $(1.25, 2.1)$. Moreover, the perception field of the robot had a radius of 1.5 m.

With regard to the source of uncertainties in the real-life experiments, the drift of the odometry measurements based on the motion of the wheels generates disturbances in computing the position of the robot. In addition, another source of uncertainties is due to the misalignment of the map of the environment generated by the robot and the real configuration of the scenario. These uncertainties also affect the localization of the robot. Finally, there may be mismatches between the model of the TurtleBot that is used in ROS for sending the motion commands and the actual behavior of the robot.

We have presented the results of the experiments in Tables 2.5, 2.6 and 2.7 (for scenarios 1, 2 and 3, respectively), where the time of detecting the victim and the average computation time per iteration are given. The results are presented and compared for our approach, and for the target-oriented and coverage-oriented approaches given in, respectively, [226] and [227]. For all the three scenarios, our approach detects the victim (up to 50.0% with respect to the slowest method) faster. In all the 3 scenarios our approach succeeds in detecting the victim, whereas each of the other 2 approaches fails

Figure 2.12: A view of the scenario of the experiments, with the robots (the TurtleBot used as the SaR robot is in the middle and the iCreate3 robot used as the victim is at the side) and the obstacles (white buckets).

	Our tube-based MPC	A* MPC [226]	Randomized MPC [227]
Time step of victim detection	10	12	-
Computational cost (mean)	12.91	43.98	45.68
Computational cost (std)	9.50	24.41	17.64

Table 2.5: Time step when the victim is detected, as well as the average computation time in seconds per optimization iteration, and corresponding standard deviation, for scenario 1: The results are shown for our approach, for target-oriented A* MPC [226] and for coverage-oriented randomized MPC [227].

once. This is due to the fact that our approach has a victim movement model and incorporates the uncertainties in the trajectories of these victims, which together raise the chances of detecting the victims, in a more time-efficient way. About the average computation time, our approach is up to 76.1% faster than the two other approaches. This is in accordance with Table 2.2 for the simulations results. Corresponding videos of the experiments are publicly available via the 4TU.ResearchData repository [223].

We have also performed an experiment to compare the area coverage percentage of the three approaches, when the coverage term (i.e., (2.14)) has been activated for our approach. In this case, a static victim is positioned at $(-1,0)$, and the obstacles are removed. The starting and exit coordinates of the robot are, respectively, $(-1.25,-2.1)$ and $(1.25,-2.1)$. The results for scenario 4 are shown in Table 2.8, where our approach covers, respectively, 24.7% and 13.6% more than the approaches in [226] and [227].

We have analyzed the computation times of the approach using both nominal and ancillary controllers (therefore, the complete tube-based MPC approach), and only nominal controller (therefore decoupled path planning and trajectory tracking approach). For each of the three scenarios, considering the mean computation time of all the MPC optimization iterations, we have:

- Scenario 1: 2.5086 s (both controllers) and 2.1926 s (only nominal controller).

	Our tube-based MPC	A* MPC [226]	Randomized MPC [227]
Time step of victim detection	15	-	17
Computational cost (mean)	19.88	53.77	37.39
Computational cost (std)	15.23	8.36	20.75

Table 2.6: Time step when the victim is detected, as well as the average computation time in seconds per optimization iteration, and corresponding standard deviation, for scenario 2: The results are shown for our approach, for target-oriented A* MPC [226] and for coverage-oriented randomized MPC [227].

	Our tube-based MPC	A* MPC [226]	Randomized MPC [227]
Time step of victim detection	13	17	26
Computational cost (mean)	14.50	60.60	30.01
Computational cost (std)	8.42	21.74	15.92

Table 2.7: Time step when the victim is detected, as well as the average computation time in seconds per optimization iteration, and corresponding standard deviation, for scenario 3: The results are shown for our approach, for target-oriented A* MPC [226] and for coverage-oriented randomized MPC [227].

- Scenario 2: 1.4663 s (both controllers) and 1.3298 s (only nominal controller).
- Scenario 3: 2.6402 s (both controllers) and 2.7304 s (only nominal controller).

Therefore, our combined approach is only 12,60% slower than the decoupled approach in the first scenario and only 9,31% slower in the second scenario, while it is 3.30% faster in the third. This is due to the fact that the outputs of the optimizations are different in the two cases, so a trajectory may be longer in a scenario with a certain controller, and shorter in another one. Hence, there can be a benefit of using a combined approach rather than a decoupled approach, depending on the specific scenario.

2.4. CONCLUSIONS AND FUTURE TOPICS

This chapter presented a combined target-oriented and coverage-oriented path planning approach based on robust tube-based implementation of a novel formulation of MPC for a ground SaR robot in a scenario with dynamic targets. The controller determines an optimal path to the moving victims, optimizing the mission time and the area coverage, while dealing with uncertainties. In our case study, we have shown that our method outperforms two target-oriented and two coverage-oriented state-of-the-art methods in victim detection efficiency, area coverage, and mission completion time, while being robust to uncertainties.

In the future, in order to address more indoor scenarios, we propose to include fire in the simulations and a fire propagation model as prediction model for MPC: In particular, a fire model based on cellular automata will be incorporated and we will study how our approach performs under different conditions of fire propagation based on wind speed and direction, structure and materials of the buildings, and different propagation times from ignition points. Moreover, this work can be expanded to a multi-robot system: We will analyze how the proposed MPC formulation can be used in a distributed or a decentralized architecture, where considering the exchange of information among all the

	Our tube-based MPC	A* MPC [226]	Randomized MPC [227]
area coverage percentage	74.6 %	49.9 %	61.0 %

Table 2.8: Area coverage percentage for scenario 4: The results are shown for our approach, A* MPC [226] and Randomized MPC [227].

robots, among clusters of them, or no communication; in this case, how the method will scale with the number of robots will be studied and clarified. Furthermore, running simulations in photorealistic environments, for example by using Gazebo environments provided by Amazon Web Services, is recommended. We also propose to extend the case study benchmarks by considering a state-of-the-art learning-based approach to compare against. Systematic hyperparameter search could also be used in the future to tune the weights of the terms of the MPC objective function, to allow better scalability when the number of parameters increases, and to reduce time investment in the parameters design. Another topic for future research is to implement a simplified formulation of the given MPC problem (e.g., linearized) and compare the performance and computations time with when nonlinear MPC is solved directly. Stability analysis is another topic for future work. Assumption A4 (i.e., perfect knowledge of the states of the robot) may be relaxed in the future by integrating of a state estimator within our MPC approach. In addition, in view of the promising results from the computer-based simulations and experiments with real robots, in order to tackle the computation time for more complex SaR cases, fast MPC methods may be adopted in the future for the proposed approaches of this chapter. Finally, we can apply one of the techniques that are available in literature in order to deal with the fact that the decision step of the robot takes several seconds while the victims are moving according to their model, so that we can synchronize them.

DATA AVAILABILITY

We provide two tables with the values of the non-smoothness map, for each coordinate x and y, and videos corresponding to plots in Figures 2.5 in the 4TU.ResearchData repository [223].

3

Enabling Robots to Autonomously Search Dynamic Cluttered Post-Disaster Environments

Robots will bring Search-and-Rescue (SaR) in disaster response to another level, in case they can autonomously take over dangerous SaR tasks from humans. A main challenge for autonomous SaR robots is to safely navigate in cluttered environments with uncertainties, while avoiding static and moving obstacles. We propose an integrated control framework for SaR robots in dynamic, uncertain environments, including a computationally efficient heuristic motion planning system that provides a nominal (assuming there are no uncertainties) collision-free trajectory for SaR robots and a robust motion tracking system that steers the robot to track this reference trajectory, taking into account the impact of uncertainties. The control architecture guarantees a balanced trade-off among various SaR objectives, while handling the hard constraints, including safety. The results of various computer-based simulations, presented in this chapter, showed significant outperformance (of up to 42.3%) of the proposed integrated control architecture compared to two commonly used state-of-the-art methods (Rapidly-exploring Random Tree and Artificial Potential Function) in reaching targets (e.g., trapped victims in SaR) safely, collision-free, and in the shortest possible time.

Parts of this chapter have been published in Scientific Reports **15**, 34778 (2025) [2].
M. Baglioni contributed to the methodology, assessed the results, supervised the MSc student involved, and reviewed the final draft of the chapter. K. Rado performed the conceptualization, performed the simulations, contributed to methodology and results, and wrote the first draft. A. Jamshidnejad contributed to methodology, assessed the results, edited the final draft, and performed supervision.

3.1. INTRODUCTION

AUTONOMOUS Search-and-Rescue (SaR) robots are emerging in post-disaster response, in order to reduce the exposure of human SaR staff to life-threatening risks. Structural damages and collapsed buildings reshape an environment post disaster. Thus, reliable and effective control methods for autonomous, safe navigation in dynamic and (partially) unknown environments are crucial for SaR robots [19], [24], [25], [134], [236], [237]. Another main challenge for autonomous navigation of SaR robots is avoiding the obstacles that move according to generally nonlinear trajectories, without (significantly) compromising the mission criteria. State-of-the-art solutions either rely on assuming static obstacles only, or provide ad-hoc solutions that cannot be generalized or provide guarantees on the performance of SaR missions via robots [154]–[156].

To tackle these challenges, we propose a novel control architecture for autonomous SaR robots that should reach known target positions in dynamic cluttered post-disaster environments: We first divide the mission planning of SaR robots into two stages, motion planning and optimal motion tracking. We extend the greedy heuristic path planning method [193] that has been introduced to generate a nominal obstacle-free path towards the targets of the robot, in order to account for dynamic obstacles that may appear on the way of the robot. Note that by nominal, we are referring to a trajectory that is determined assuming there are no uncertainties or uncontrollable disturbances. We then integrate this path planning method with an optimal motion tracking system that closely follows the nominal trajectory estimated by the heuristic motion planning system while guaranteeing robustness to bounded uncontrollable disturbances.

The optimal motion tracking system is based on a robust version of Model Predictive Control (MPC), called Robust Tube-based Model Predictive Control (robust TMPC). Robust TMPC has proven very effective in steering robots for post-disaster SaR [41], due to its unique capability in providing robustness to bounded uncertainties, balanced trade-offs among competing objectives of SaR (e.g., maximizing the area coverage and minimizing the mission time [25]), systematic handling of the states and inputs constraints, and on-the-go stability guarantees [42].

3.2. MAIN CONTRIBUTIONS AND ROAD MAP OF THE CHAPTER

This chapter introduces a novel planning and control framework for autonomous navigation in dynamic, cluttered environments with the following main contributions:

- **Unified planning and control:** We integrate a greedy heuristic path planning system with a robust Robust Tube-based Model Predictive Control (robust TMPC) system within a single-layer architecture. This design synergizes the responsiveness of heuristic reasoning with the constraint-handling guarantees of Model Predictive Control (MPC)-based systems, leading to enhanced computational efficiency and safety compared to traditional bi-level planning-control schemes.

- **Dynamic obstacle avoidance via obstacle belts:** We extend the path planning system to handle moving obstacles by predicting their trajectories and aggregating them into time-indexed constraint regions, called *obstacle belts*, to enable anticipatory collision avoidance.

- **Uncertainty-aware robust TMPC reformulation:** We reformulate robust TMPC by replacing its nominal controller with the heuristic planning mechanism, while retaining the ancillary controller for robust trajectory tracking. Time-varying constraints and dynamic tightening account for both external disturbances and perception uncertainty.

- **Stateless, perception-driven control:** The framework operates without memory of past states, by relying solely on real-time perception. While this condition can be relaxed in the presence of advanced sensing and computational resources, this non-restrictive design reduces reliance on such infrastructure and enables deployment in partially observable environments typical of Search-and-Rescue (SaR) missions.

Extensive simulations demonstrate the superior performance of the proposed method compared to state-of-the-art planners, particularly Horizon-based Lazy Rapidly-exploring Random Tree (HL-RRT*), which embodies heuristic reasoning similar to our planning system, and COLREGS Artificial Potential Function (APF), which emphasizes systematic obstacle avoidance and target convergence, as does the robust TMPC component of our proposed framework. The proposed approach outperforms both, especially in scenarios involving uncertainty and dynamic constraints.

In the rest of this chapter we provide a background discussion, the problem statement and assumptions, our proposed methods, the results of the case study with discussions, conclusions, and topics for future work. Moreover, Table 3.1 gives the mathematical notation that is frequently used in the chapter.

Table 3.1: Frequently used mathematical notations

Notation	Explanation	Notation	Explanation
κ	Discrete time step	$x^{\text{static,obs}}(o)$	The x position of static obstacle o
c	Control sampling time	$y^{\text{static,obs}}(o)$	The y position of static obstacle o
N^{p}	Prediction horizon of MPC	$x_{\kappa}^{\text{dyn,obs}}(o)$	The x position of dynamic obstacle o at time step κ
x_{κ}^{rob}	The x position of the robot at time step κ	$y_{\kappa}^{\text{dyn,obs}}(o)$	The y position of dynamic obstacle o at time step κ
y_{κ}^{rob}	The y position of the robot at time step κ	$v_{x,\kappa}^{\text{dyn,obs}}(o)$	Horizontal velocity of dynamic obstacle o at time step κ
$\theta_{\kappa}^{\text{rob}}$	Heading angle of the robot at time step κ	$v_{y,\kappa}^{\text{dyn,obs}}(o)$	Vertical velocity of dynamic obstacle o at time step κ
v_{κ}^{rob}	Linear velocity of the robot at time step κ	ρ^{obs}	Radius of the obstacle or of the smallest circular area encountering it
$\omega_{\kappa}^{\text{rob}}$	Angular velocity of the robot at time step κ	$\boldsymbol{x}_{\kappa}^{\text{ref}}$	Vector of robot's reference states for time step κ
ρ^{rob}	Radius of the robots (assuming circular shapes)	$\boldsymbol{u}_{\kappa}^{\text{ref}}$	Vector of robot's reference control inputs for time step κ

3.3. BACKGROUND DISCUSSION

Planning the disaster response via robots depends on the disaster's environment. Three operational environments are identified for SaR [14], urban (involving constrained environments [238], e.g., inside a collapsed building), wilderness (involving open-ended environments [86], e.g., a forest), and air-sea (involving waters [239], e.g., a sea where vessels accidents or water landing has occurred). Ground robots [97], flying robots [240], and underwater robots [102] may be used in SaR, while for various indoor constrained environments ground robots are preferred [96]. Our focus is on urban SaR via ground robots. We address the problem of autonomous, effective mission planning and safe navigation of these robots. This yields to a generally nonlinear constrained optimization-based problem with multiple competing objectives, e.g., reducing the mission time while increasing the area coverage.

For SaR robots, it is common to use heuristic methods, especially those based on shortest path planners and artificial intelligence, e.g., reaction-based swarming [241], fuzzy logic control [242], and ant colony optimization [243]. The main motivation for using these methods is their computational efficiency, making them suitable for on-board deployment in SaR robotics. The main shortcomings of heuristic approaches, however, are their ad-hoc case-specific nature and the lack of performance guarantees. With advances in computational power and robotics technology [13], [104], incorporating guaranteed mathematical approaches or novel integrated versions of them that provide balanced trade-offs between a high performance and computational efficiency has been emerging [1], [25], [30], [133], [193], [244].

Model Predictive Control (MPC) is a mathematics-based, systematic control approach that determines a sequence of control inputs by optimizing an objective function within a given prediction window, satisfying the state and input constraints. MPC is implemented in a rolling horizon fashion, i.e., after determining the control input sequence, only the first one is injected into the controlled system and the MPC problem is solved for the shifted prediction window at the next time step. MPC has proven very effective for addressing constrained optimization-based problems. MPC has often been used in static SaR environments for tracking a reference trajectory that is assumed to be provided by another path planning approach [96], [133], [191], [192].

In connection to path planning for robots, exploration and coverage of the SaR environment are both crucial [165], [166]. Various approaches for area coverage have been proposed based on random search [227] or artificial intelligence (especially deep learning, reinforcement learning, fuzzy logic control [140], [183], [242]). In this regard, MPC has been used for SaR robots to determine control inputs that maximize a reward for visiting new parts of the environment [30], [176], [177]. Another novel application of MPC in multi-objective SaR via robots is through a bi-level architecture [25], where a supervisory MPC level enhances the area coverage by re-distributing SaR robots when they locally decide to visit the same or neighboring parts of the environment. This division of local and supervisory decision making provides a balanced trade-off between optimizing the global objectives of the SaR mission and meeting the computational requirements.

Robust versions of MPC [31], in particular Robust Tube-based Model Predictive Control (robust TMPC), [220] deal with bounded uncertainties, e.g., bounded disturbances

and perception errors that often occur for SaR robots. In robust TMPC, first the nominal version of the MPC problem is solved where no uncertainties are considered and the constraints have been tightened compared to the original problem [245]. Constraint tightening implies adjusting the bounds on the states and/or control inputs (e.g., by shrinking their admissible sets) to ensure that the controlled system always operates within its safe operational limits [42], although the operational conditions may be affected by larger disturbances than those considered in the decision making procedure. During the online implementation of robust TMPC, an ancillary control input minimizes the error between the nominal and actual states [246]. While the actual state trajectory may deviate from the nominal one, it always remains within a safe bounded region, called the *tube*, where the constraints are guaranteed to always be satisfied. The diameter of the cross section of the tube is determined according to the maximum difference between the nominal and actual states, considering the worst-case uncertainty scenario.

Within the context of robust TMPC, [247] introduces a robust MPC framework for real-time robot navigation. This method incorporates time-varying constraints and improves tracking performance in the presence of uncertainty. Based on the results for lab-based simulations applied to robot motion control, their proposed framework improves trajectory tracking and increases navigation success rates and robustness compared to methods such as conventional MPC and other robust MPC. However, due to the deviation from the nominal values in closed loop, this framework tends to generate large tubes, which keeps conservatism relatively high. In [248], a hybrid control architecture is proposed for multi-Unmanned Aerial Vehicle (UAV) systems that combines explicit MPC, leveraging offline computations, with robust TMPC. By exploiting the benefits of both approaches, their proposed hybrid architecture obtains lower computational complexity and enhanced robustness to disturbances, such as wind and sensor noise. However, the presented method is applied on a linearized model of the UAVs, therefore stability guarantees are provided for linear systems only, and thus the applicability to highly nonlinear systems might be limited. Surma and Jamshidnejad [41] propose a robust TMPC framework that models or learns disturbance dynamics and incorporates this information into the decision making process, thereby reducing conservatism and the risk of infeasibility. While this framework leads to enhanced performance through improved trajectory planning, computational efficiency remains a concern due to the use of polytopes to describe the error sets, making the computational complexity grow exponentially with the prediction horizon.

Navigating SaR robots is often target-driven [209], i.e., it involves steering the robot via reference points towards a given target. In this context, learning-based methods, such as reinforcement learning, are widely adopted [206], [249]. Recent hybrid approaches combine high-level learning-based decision policies with classical planners to achieve more adaptive navigation. For example, RLoPlanner [250] integrates reinforcement learning with low-level motion planning, while hierarchical navigation algorithms, e.g.,[251], employ model-free strategies to guide flying robots in structured environments.

However, many of these methods focus on static or slowly changing environments and lack explicit mechanisms for handling time-varying constraints introduced by moving obstacles. They often require extensive training data and do not provide formal safety guarantees.

In contrast, this chapter proposes a novel control architecture specifically tailored for dynamic SaR scenarios. Our framework integrates a heuristic steering approach with robust, constraint-aware MPC. This allows for real-time adaptation to moving obstacles and bounded disturbances [199], without relying on learned policies or hierarchical coordination, and results in a lightweight, yet reliable, architecture that maintains safety and computational efficiency under high environmental uncertainty.

3.4. PROBLEM STATEMENT AND ASSUMPTIONS

We consider the control problem of a SaR ground robot that should autonomously navigate a dynamic, cluttered, and (partially) unknown environment to reach a known target point. The control problem is formulated within a 2D continuous-space framework in discrete time, with time step variable κ. In the mathematical derivations, we consider a differential drive [252] ground robot with a circular shape of radius ρ^{rob}, but the proposed methods are adoptable for different types and shapes of robots and post-disaster environments. For the sake of simplicity of the mathematical formulations, the static and dynamic obstacles all have a circular shape with a fixed radius ρ^{obs}. This is equivalent to considering the smallest circular area that encounters an obstacle an area for the robot to avoid. The robot has a camera that provides visual information within a circular perception field centered around the robot and perceives the shape, position, and velocity of the obstacles. In addition to static obstacles (e.g., rubble or stones) there are moving obstacles (e.g., humans or falling debris) in the environment. The control system should determine per control time step the linear and angular velocities that steer the robot towards its next desired states.

We consider the following assumptions:

A1 Per time step, the robot has perfect information of its own states and scans and gathers information about the part of its environment that falls within the perception field of the robot's camera. The robot keeps no memory of the past.

A2 Environmental unmodeled disturbances (e.g., unevenness or non-smoothness of terrain) affect the states of the robot. In other words, the position, given for the center of the robot, and the velocity may diverge from their nominal values due to terrain-induced disturbances. Such effects are treated as bounded external disturbances.

A3 The perception of the robot about the position of the static obstacles is perfect, but for dynamic obstacles this perception is prone to a bounded error (which may be due to, e.g., the motion blur or the obstacle moving faster than the camera's update rate).

3.5. BI-LEVEL CONTROL ARCHITECTURE FOR AUTONOMOUS SAFE SEARCHING OF DYNAMIC CLUTTERED AREAS

The control architecture that is proposed for steering a robot in dynamic cluttered SaR environments is illustrated in Figure 3.1: In order to improve the computational effi-

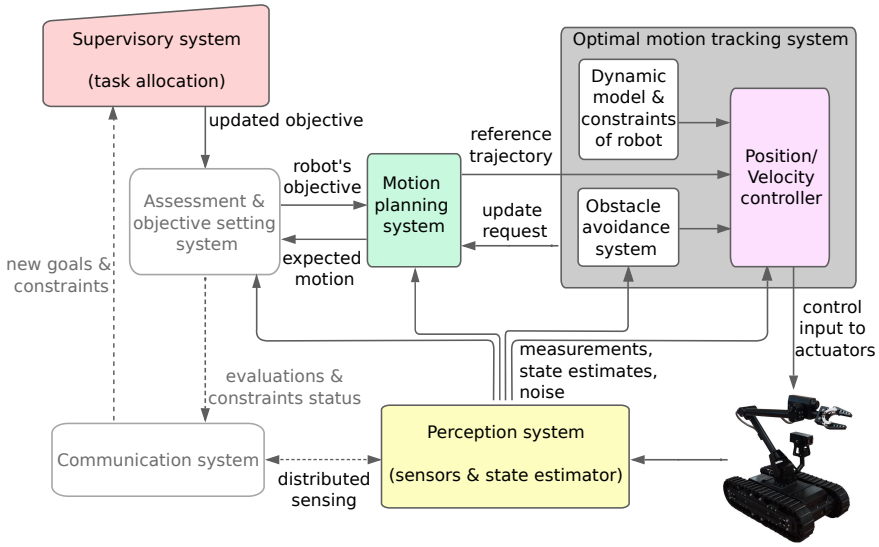


Figure 3.1: General control architecture proposed for autonomous steering of robots in cluttered dynamic SaR environments.

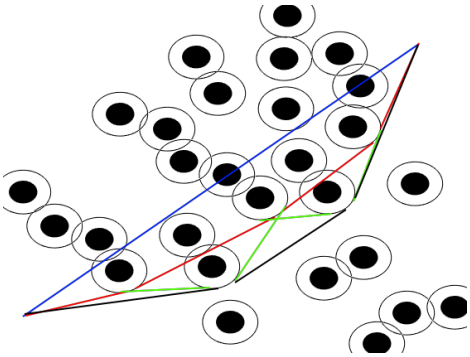
ciency and the responsiveness of the steering system of the robot, both highly crucial for SaR missions [253], [254], we opt to separate the steering system into a motion planning system and an optimal motion tracking system, which must follow the trajectory that is generated by the motion planning system as closely as possible. The output of the motion planning system (shown via a green box in Figure 3.1) is injected, as the reference trajectory, into the optimal motion tracking system (shown via a gray box in Figure 3.1), which will request the motion planning system to re-plan the trajectory whenever needed.

3.5.1. MOTION PLANNING SYSTEM: HEURISTIC PLANNING

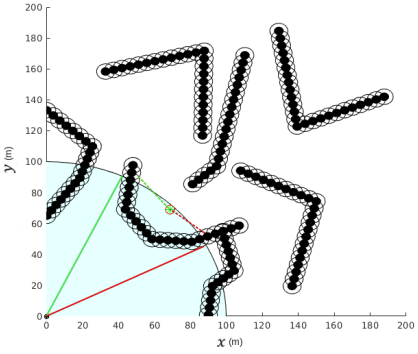
Due to its ease of implementation and computational efficiency, the obstacle-avoiding shortest path approach introduced by Jamshidnejad and Frazzoli [193] was chosen as the basis for the motion planning system. There are, however, two main challenges regarding the adoption of this approach for a dynamic, cluttered SaR environment that should be addressed: First, the approach only includes static obstacles and does not incorporate dynamic ones. Second, since this approach relies on local knowledge of the robot from its environment, its feasibility may be at risk.

MODIFIED OBSTACLE-AVOIDING SHORTEST PATH APPROACH FOR DYNAMIC ENVIRONMENTS

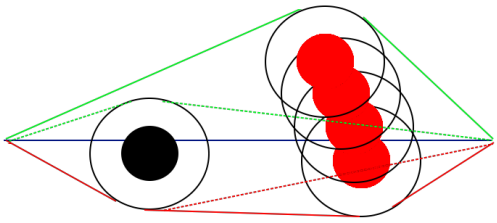
In the original algorithm [193], a greedy heuristic approach has been adopted to avoid obstacles that are composed of any arbitrary union of circular shapes, by generating the path across the tangential arcs to these circular shapes (see Figure 3.2a). This way various arbitrarily-shaped obstacles can be handled by dividing them into smaller pieces and by approximating each piece by the smallest circle that encounters the piece. Due to its



(a) Path smoothening occurs in case of self-intersection: The plot shows the connecting line in blue, the first and final iterations in red and green, and the smoothened path in black.



(b) Current candidate paths (solid+dashed red and green), with a target that is unreachable according to current perception, are replaced by paths with reachable targets (solid green and red).



(c) Solid curves are paths that consider all positions of the dynamic obstacle (red circle) within the prediction window. Dashed curves are original paths assuming the obstacle remains static (black circle).

Figure 3.2: Main idea behind the heuristic path planning approach [193]. Black hollow circles surrounding static (black) and dynamic (red) obstacles show forbidden areas for the robot. In plot 3.2b the robot is at the origin and its current perception field is the light blue area. In plot 3.2c the robot is at the left-hand side end of the blue piece of line and its current target is at the right-hand side end of it. This blue line represents the straight path that, regardless of the obstacles, connects the current position of the robot and the position of its current temporary target.

limited perception field, the robot determines a temporary intermediate target, which it intends to approach, per control time step. The motion planning system evaluates the shortest paths at both sides of the straight path that, regardless of the obstacles, connects the current position of the robot and the position of its current temporary target. In case the candidate paths are of the same length, one is chosen randomly. Equidistant points on the shortest path are injected as reference points into the local motion tracking system of the robot. In case the resulting shortest path is non-smooth or self-intersecting (see the green curves in Figure 3.2a), a tangent line is drawn across the outer obstacles to smoothen the path (see the black curve in Figure 3.2a).

We made the following modifications to the heuristic path planning approach, in order to make it suited for practical cases where the robot should avoid moving obstacles as well and does not store detailed past information for efficiency of the on-board computations and the memory and energy consumption (see Assumption **A1**): Due to the limited awareness of the robot about its environment (limited to the perception field of its camera), it may face situations where the temporary target that connects its current position and the final target is unreachable according to the robot's environmental awareness (Figure 3.2b). We address this by replacing the original candidate paths with traversable paths within the detection zone of the robot that have an endpoint as close as possible to the unreachable temporary target.

Moreover, in order to incorporate the dynamic obstacles into the heuristic motion planning approach, the following steps are taken, assuming perfect knowledge about the dynamic equations of the obstacle:

Step 1 The obstacle avoiding path planning approach is implemented to generate traversable, safe paths for the robot considering only the obstacles perceived as static.

Step 2 Using the predicted dynamics of the dynamic obstacles within a given prediction window, all relevant dynamic obstacles are modeled, each as a set of static obstacles located at these predicted positions.

Step 3 The reference points of the robot for all the time steps within this prediction window are specified on the shortest traversable and safe path obtained via Step 1, with all positions of the dynamic obstacle for the time steps within the prediction window included in the picture of the robot as static obstacles.

Step 4 In case, by comparing the position of the robot and the static obstacles that represent the position of the dynamic obstacle any risks of collision are detected, the illustrated static obstacles for the time steps with a risk of collision will be united to form an obstacle belt. A new collision-free path is then generated.

Step 5 Repeat Step 1–Step 4 until the shortest traversable and safe path is determined.

Figure 3.2c illustrates how the shortest traversable and safe path is determined, by first considering only the static obstacle (shown via the black circle) and then by modifying the resulting path in order to avoid collisions with the static obstacles that represent the dynamic obstacle for all the time steps within the given prediction window. We assume

that the robot moves across this path with linear and angular velocities that ensure a trade-off between performance and safety. This, for instance, is obtained by selecting the higher threshold between half of the maximum velocity of the robot and its midpoint velocity.

Remark 4. ***Obstacle Representation and Generalization:*** *To enable safe navigation in environments with arbitrarily-shaped or moving obstacles, we adopt the concept of a* forbidden belt, *originally introduced in [193]. This construct allows multiple circular (primitive) obstacle regions, each representing predicted positions of dynamic objects at discrete time steps, to be unified into a deformable constraint region. The resulting belt approximates the occupancy of arbitrarily shaped or time-evolving obstacles over a prediction horizon. This abstraction supports the generalization of our method to real-world SaR environments, where obstacle boundaries are often irregular and unknown a priori. Crucially, the method does not require explicit modeling of obstacle geometry, but instead incorporates their aggregated occupancy.*

Remark 5. ***Compatibility with Memory-augmented Settings:*** *While the no-memory assumption enables a lightweight and reactive implementation, the proposed framework remains fully compatible with memory-augmented modules (e.g., SLAM or local map tracking), which will potentially further improve robustness in partially observable environments.*

Next, we expand the discussions for situations where the dynamics of the moving obstacles is not perfectly known by the robot, thus the robot's predictions about the future positions of the obstacle are prone to errors.

INCLUDING DYNAMIC OBSTACLES WHEN PERCEPTION ERRORS MAY EXIST IN THE MOTION PLANNING SYSTEM

In practice, especially in cluttered SaR environments that are prone to various uncontrolled stimuli, the dynamics of the obstacles cannot be perfectly captured via mathematical models. Additionally, the perceived position of dynamic obstacles based on the images of the camera is prone to errors (Assumption **A3**). Therefore, the path planning approach must be made robust to these uncertainties. This is guaranteed if the robot safely navigates in the environment even when maximum uncertainties are realized, i.e., when the forbidden areas (hollow circles around each dynamic obstacle in Figure 3.2) are expanded considering the maximum modeling or perception error. Moreover, the cumulative errors enhance the uncertainty of the predictions. This effect is incorporated by increasing the upper value of the errors according to the proximity of the prediction to the current time. In fact, a larger prediction window allows to incorporate and assess longer-term impacts of current control inputs, which increases the chances of a safe and optimal mission and decreases the risk of recursive infeasibility. However, in addition to heavier online computations, prediction in larger windows leads to larger cumulative errors and thus risks to safety and degrading the performance. This further motivates the introduction of a bi-level control architecture that generates a reference path that approximately provides safety and desirable performance, and that delegates the obstacle avoiding task to a reference tracking control system (see Figure 3.1). Whenever the

local reference tracking control problem becomes infeasible or the performance criteria falls under desirable thresholds, the motion planning system updates the reference path, based on the updated information. Meanwhile, the optimal motion tracking system keeps the motions of the robot safe and crash-free.

3.5.2. OPTIMAL MOTION TRACKING SYSTEM: ROBUST TUBE-BASED MPC

Based on assumptions **A2** and **A3**, two sources of uncertainties affect the performance of the robot: The unmodeled disturbances that deviate the states of the robot from the planned states and the errors in perceiving the position of dynamic obstacles. Therefore, we use robust TMPC in motion tracking in order to optimally follow the reference trajectory that is determined by the motion planning system, while systematically incorporating all the constraints into the decision making procedure. Robust TMPC uses dynamic mathematical models to predict the states of the robot and of the dynamic obstacles in a given prediction window, and uses these predictions to optimize the control inputs. The state vector of the robot, which encapsulates all the necessary past information for time step κ to predict the future states, is given by $\boldsymbol{x}_\kappa^{\text{rob}} = [x_\kappa^{\text{rob}}, y_\kappa^{\text{rob}}, \theta_\kappa^{\text{rob}}]^\top$, which includes, respectively, the 2D position of the center of the robot and the robot's orientation with respect to the horizontal axis. The control input vector that steers the motion of the robot for time step κ is given by $\boldsymbol{u}_\kappa^{\text{rob}} = [v_\kappa^{\text{rob}}, \omega_\kappa^{\text{rob}}]^\top$, which includes, respectively, the linear and the angular velocities of the robot. The obstacles are identified by their position vector $\boldsymbol{r}^{\text{static,obs}}(o) = [x^{\text{static,obs}}(o), y^{\text{static,obs}}(o)]^\top$ for static obstacle o and by their state vector $\boldsymbol{x}_\kappa^{\text{dyn,obs}} = [x_\kappa^{\text{dyn,obs}}(o), y_\kappa^{\text{dyn,obs}}(o), v_{x,\kappa}^{\text{dyn,obs}}(o), v_{y,\kappa}^{\text{dyn,obs}}(o)]^\top$ per time step κ for dynamic obstacle o, including the 2D position $\boldsymbol{r}_\kappa^{\text{dyn,obs}}(o)$ and the velocity elements of the obstacle, respectively (note that due to the circular shape of obstacles, instead of their orientation or angular velocity, the robot perceives the components of the linear velocity). The state evolves due to the accelerations $a_{x,\kappa}^{\text{obs}}(o)$ and $a_{y,\kappa}^{\text{obs}}(o)$ by the driving forces of the obstacle. Next, we explain how the motion tracking system predicts the states of the robot and of the dynamic obstacles within a given prediction window.

DYNAMIC PREDICTION MODELS FOR THE MOTION OF THE ROBOT AND OF THE DYNAMIC OBSTACLES

The kinematics equations for translational motion of the centroid of a differential drive mobile SaR robot, as well as the orientation of the robot, both used by the motion tracking robust TMPC system are given by:

$$x_\kappa^{\text{rob}} = x_{\kappa-1}^{\text{rob}} + c\left(v_\kappa^{\text{rob}} \cos\left(\theta_{\kappa-1}^{\text{rob}}\right) - c\omega_\kappa^{\text{rob}} v_\kappa^{\text{rob}} \sin\left(\theta_{\kappa-1}^{\text{rob}}\right)\right) \tag{3.1a}$$

$$y_\kappa^{\text{rob}} = y_{\kappa-1}^{\text{rob}} + c\left(v_\kappa^{\text{rob}} \sin\left(\theta_{\kappa-1}^{\text{rob}}\right) + c\omega_\kappa^{\text{rob}} v_\kappa^{\text{rob}} \cos\left(\theta_{\kappa-1}^{\text{rob}}\right)\right) \tag{3.1b}$$

$$\theta_\kappa^{\text{rob}} = \theta_{\kappa-1}^{\text{rob}} + c\omega_\kappa^{\text{rob}} \tag{3.1c}$$

The state update equations (3.1a)-(3.1c) have been discretized in time using sampling time c, i.e., the states are updated every c time units, while during this interval their most recently updated values are used.

In the proposed architecture, both the motion planning system (in Step 3) and the optimal motion tracking system (as a prediction model embedded in MPC) need to model

the dynamics of the moving obstacles. In case any knowledge exists or is deducible via filters [3] about the pattern of movement of the obstacles, the corresponding kinematics equations may be obtained.

Otherwise, since the proposed framework does not rely on prior knowledge of precise obstacle trajectories, it should predict obstacle motions based on rational assumptions, e.g., obstacles follow motion patterns similar to the behavior of the robot itself.

Since based on assumption **A2**, robots deploy perception pipelines that estimate obstacle motion under bounded uncertainty, predicted uncertainty envelopes can be incorporated into the obstacle belt representation to preserve safety guarantees, even in the presence of unpredictable or partially observed obstacle motions [1].

Regardless of the prediction method, the robust TMPC system, as is detailed in the next section, incorporates time-varying constraint tightening, which accounts for uncertainty in obstacle motion. This ensures that safety constraints remain satisfied, even under bounded deviations. The robot continuously monitors the motion of objects within its perception field, and if significant changes are detected, the framework triggers real-time re-planning, supported by the rolling-horizon strategy of robust TMPC. This combination of conservative planning and adaptive response enables safe navigation in environments with limited or noisy motion information. In addition, note that in the motion tracking system based on robust TMPC, the model of the robot that is employed is a linearized version of (3.1).

FORMULATING THE TMPC PROBLEM OF THE MOTION TRACKING SYSTEM INCORPORATING THE IMPACT OF UNCERTAINTIES

The objective function of the MPC problem for motion tracking per time step κ is composed of two terms: The first term includes the offset of the state vector trajectory of the robot from the reference trajectory $\{\boldsymbol{x}^{\text{ref}}_{\kappa+1},\ldots,\boldsymbol{x}^{\text{ref}}_{\kappa+N^{\text{p}}}\}$ within the prediction window $\mathbb{P}_\kappa = \{\kappa+1,\ldots,\kappa+N^{\text{p}}\}$ that is generated by the heuristic motion planning system. The second term of the objective function represents the kinetic energy of the robot and incorporates the impact of the velocity vector of the robot to improve the energy efficiency for its motion and to assist with obstacle avoidance. This term serves dual purposes: Reducing energy consumption and moderating velocity near dynamic obstacles, which enhances safety in cluttered or uncertain environments. Note that replacing this term with a time-minimization one encourages maximum velocity, which may compromise energy efficiency and increase the risk of unsafe behavior near obstacles. Alternatively, adding a time-related term alongside the existing objectives introduces a competing priority into the optimization problem that potentially undermines the real-time tractability and responsiveness required in safety-critical scenarios.

These terms are weighed using positive parameters $w_1 < 1$ and w_2, where the impact of the predictions that correspond to farther times in the future is discounted (due to being prone to larger estimation errors) by multiplying them by w_1^k. In other words, when k is a larger time step within the prediction horizon, the weight of the corresponding term is smaller, because $w_1 < 1$. The robust TMPC problem for time step κ, with $\boldsymbol{x}^{\text{rob}}_\kappa$ and $\boldsymbol{u}^{\text{rob}}_{\kappa-1}$ given, is formulated within the prediction window $\mathbb{P}_\kappa$ via the following

minimization problem:

$$\min_{\tilde{\boldsymbol{u}}_\kappa^{\text{rob}}(N^{\text{c}},N^{\text{p}}),\,\tilde{\boldsymbol{x}}_\kappa^{\text{rob}}(N^{\text{p}})} \left(\sum_{k=\kappa+1}^{\kappa+N^{\text{p}}} w_1^{k/(\kappa+1)} \left\| \boldsymbol{x}_k^{\text{rob}} - \boldsymbol{x}_k^{\text{ref}} \right\| + w_2 \sum_{k=\kappa}^{\kappa+N^{\text{p}}-1} \left({\boldsymbol{u}_k^{\text{rob}}}^{\top} \cdot \boldsymbol{u}_k^{\text{rob}} \right) \right) \tag{3.2}$$

subject to the following constraints, within the given prediction window $\mathbb{P}_\kappa$:

(3.1) holds for updating the states of the robot, after linearization (3.3a)

(3.1) or a dynamic equation deduced from a Kalman filter holds for moving obstacles (3.3b)

$$\boldsymbol{x}^{\min} \leq \boldsymbol{x}_k^{\text{rob}} \leq \boldsymbol{x}^{\max} \tag{3.3c}$$

$$\boldsymbol{u}^{\min} \leq \boldsymbol{u}_{k-1}^{\text{rob}} \leq \boldsymbol{u}^{\max} \tag{3.3d}$$

$$\left\| \boldsymbol{u}_{k-1}^{\text{rob}} - \boldsymbol{u}_{k-2}^{\text{rob}} \right\| \leq u^{\text{smooth}} \tag{3.3e}$$

$$\left\| \boldsymbol{r}_k^{\text{rob}} - \boldsymbol{r}_\kappa^{\text{rob}} \right\| \leq \rho_\kappa^{\text{plan}} - \rho^{\text{safe}} \tag{3.3f}$$

$$\left\| \boldsymbol{r}_k^{\text{rob}} - \boldsymbol{r}^{\text{static,obs}}(o_1) \right\| \geq \rho^{\text{safe}} + w_k^{\text{rob}}, \qquad \text{for } o_1 \in \mathscr{I}_k^{\text{static,obs}} \tag{3.3g}$$

$$\left\| \boldsymbol{r}_k^{\text{rob}} - \boldsymbol{r}_{k-1}^{\text{dyn,obs}}(o_2) \right\| \geq \rho^{\text{safe}} + w_k^{\text{rob}} + w_{k-1}^{\text{dyn,obs}}, \qquad \text{for } o_2 \in \mathscr{I}_k^{\text{dyn,obs}} \text{ and for } k > \kappa+1 \tag{3.3h}$$

$$\left\| \boldsymbol{r}_k^{\text{rob}} - \boldsymbol{r}_k^{\text{dyn,obs}}(o_2) \right\| \geq \rho^{\text{safe}} + w_k^{\text{rob}} + w_k^{\text{dyn,obs}}, \qquad \text{for } o_2 \in \mathscr{I}_k^{\text{dyn,obs}} \tag{3.3i}$$

$$\left\| \boldsymbol{r}_k^{\text{rob}} - \boldsymbol{r}_{k+1}^{\text{dyn,obs}}(o_2) \right\| \geq \rho^{\text{safe}} + w_k^{\text{rob}} + w_{k+1}^{\text{dyn,obs}}, \qquad \text{for } o_2 \in \mathscr{I}_k^{\text{dyn,obs}} \text{ and for } k < \kappa+N^{\text{p}} \tag{3.3j}$$

$$\boldsymbol{u}_{k-1}^{\text{rob}} = \boldsymbol{u}_{k-1}^{\text{ref}} + K_{k-1}\left(\boldsymbol{x}_{k-1}^{\text{rob}} - \boldsymbol{x}_{k-1}^{\text{ref}} \right) \tag{3.3k}$$

where $\tilde{\boldsymbol{x}}_\kappa^{\text{rob}}(N^{\text{p}}) = \{\boldsymbol{x}_{\kappa+1}^{\text{rob}},\ldots,\boldsymbol{x}_{\kappa+N^{\text{p}}}^{\text{rob}}\}$ and $\tilde{\boldsymbol{u}}_\kappa^{\text{rob}}(N^{\text{c}},N^{\text{p}}) = \{\boldsymbol{u}_\kappa^{\text{rob}},\ldots,\boldsymbol{u}_{\kappa+N^{\text{c}}-1}^{\text{rob}},\ldots,\boldsymbol{u}_{\kappa+N^{\text{p}}-1}^{\text{rob}}\}$, with $\boldsymbol{u}_{\kappa+N^{\text{c}}-1}^{\text{rob}} = \ldots = \boldsymbol{u}_{\kappa+N^{\text{p}}-1}^{\text{rob}}$. Constraints (3.3c) and (3.3d) impose lower and upper limits on, respectively, the states and control inputs of the robot, where the symbol $\leq$ for vectors is executed element-wise on those vectors. Constraint (3.3e) limits the rate of the changes in the consecutive control inputs that are injected into the actuators of the robot, and guarantees smooth mechanical movements or dynamic variations for the actuators of the robot. Constraint (3.3f) ensures that the position of the center of the robot for the entire prediction window remains within a bounded zone with safe borders. This zone should embed the trajectory that is planned via the heuristic motion planning system of the robot as much as safety considerations allow for it. In other words, this constraint keeps the position of the center of the robot within a circle that is centered around the current position $\boldsymbol{r}_\kappa^{\text{rob}}$ of the robot with a radius $\rho_\kappa^{\text{plan}} - \rho^{\text{safe}}$, where $\rho_\kappa^{\text{plan}}$ is the largest distance of $\boldsymbol{r}_\kappa^{\text{rob}}$ from the planned path and $\rho^{\text{safe}} \geq \rho^{\text{rob}} + \rho^{\text{obs}}$ is a parameter that is tuned based on how conservatively the safety guarantees are defined. Subtracting ρ^{safe} from the planned radius ensures that when the center of the robot is positioned on the borders of the safe zone, the robot will not crash into potential obstacles outside the zone. Figure 3.3 illustrates the essence of including constraint (3.3f). The variables $\mathscr{I}_\kappa^{\text{static,obs}}$

and $\mathscr{I}_\kappa^{\text{dyn,obs}}$ in (3.3g)–(3.3j) are, respectively, the sets of all the static and dynamic obstacles that fall within the perception field of the robot at time step $k \in \mathbb{P}_\kappa$. Constraint (3.3g) keeps the robot away from the detected static obstacles for all time steps $k \in \mathbb{P}_\kappa$ taking into account the impact of the disturbances on the position of the robot, by including the upper bound w_k^{rob} for the disturbances. In other words, terrain-induced disturbances, as described in Assumption **A2**, are modeled as bounded external disturbances within the motion tracking system and are compensated for through the robust TMPC formulation via the corresponding constraint. Constraints (3.3h)–(3.3j) prevent the robot from crashing into the dynamic obstacles that have been detected within the perception field of the robot at time step κ, by incorporating the impact of both the external disturbances that affect the position of the robot, with the upper bound w_k^{rob} (for $k \in \mathbb{P}_\kappa$), and the error in the perception of the estimated states of the dynamic obstacles, with the upper bounds $w_{k-1}^{\text{dyn,obs}}$, $w_k^{\text{dyn,obs}}$, and $w_{k+1}^{\text{dyn,obs}}$ for time steps $k-1$, k, and $k+1$, respectively, where $k \in \mathbb{P}_\kappa \backslash \{\kappa, \kappa + N^{\text{p}}\}$. Note that since a discrete-time problem is solved, per time step $k \in \mathbb{P}_\kappa \backslash \{\kappa, \kappa + N^{\text{p}}\}$ the collision avoidance of the robot and the obstacle is enforced by providing the safe distance between their centers for the current time step and its immediate previous and next time steps, in order to reduce the risk of infeasibility. The reason for excluding time steps κ and $\kappa + N^{\text{p}}$ from these constraints is the following: Since the robot does not hold any memories of the previous time steps (Assumption **A1**), at the beginning of the prediction horizon, i.e., at time step $k = \kappa$, it does not have access to the information of the dynamic obstacle for time step $k-1 = \kappa - 1$. Therefore, it cannot estimate (3.3h). Moreover, since the upper limit of the prediction horizon starting at time step $k = \kappa$ is time step $\kappa + N^{\text{p}}$, thus at time step $k = \kappa + N^{\text{p}}$ the robot does not have any information about the dynamic obstacle for time step $k+1 = \kappa + N^{\text{p}} + 1$ and hence, cannot estimate (3.3j). The upper bounds are time-varying to account for the errors accumulated within the prediction horizon, while preventing too much conservatism for robust TMPC [31].

Robust TMPC estimates or deploys nominal trajectories for the state and control inputs, and adjusts the control inputs online in order to ensure that, despite the external disturbances and perception errors, the realized state trajectory of the controlled system remains within a safe, feasible region, called the tube. In our proposed architecture, the nominal state and control input trajectories for (3.2), subject to constraints (3.3a)–(3.3k), are instead those that have been injected into the motion tracking system via the motion planning system, i.e., $\boldsymbol{x}_k^{\text{ref}}$ and $\boldsymbol{u}_k^{\text{ref}}$. The control input adjusted online via robust TMPC is then determined via (3.3k), where the reference control input is adjusted using a control increment term that is determined based on a feedback from the system, i.e., the error between the realized and reference state trajectories. Note that the control law (3.3k) is linear, since we employ a linearized version of the robot model (3.1). In fact, the gain K_k used for $k+1 \in \mathbb{P}_\kappa$ is determined per time step, as is common in robust TMPC literature, such that it stabilizes the error dynamics (which is usually done by linearizing (3.1) per time step k to obtain the dynamic and input matrices A_k and B_k, respectively, and by enforcing the condition $f^{\text{SR}}(A_k + B_k K_k) < 1$ where $f^{\text{SR}}(\cdot)$ is the spectral radius function, i.e., a function that determines the largest eigenvalue of the input matrix, in this case $A_k + B_k K_k$, which can be used to determine the tube, see also, e.g., [220]). In particular, (3.3h)–(3.3j) ensure that the realized states remain within a safe tube that is

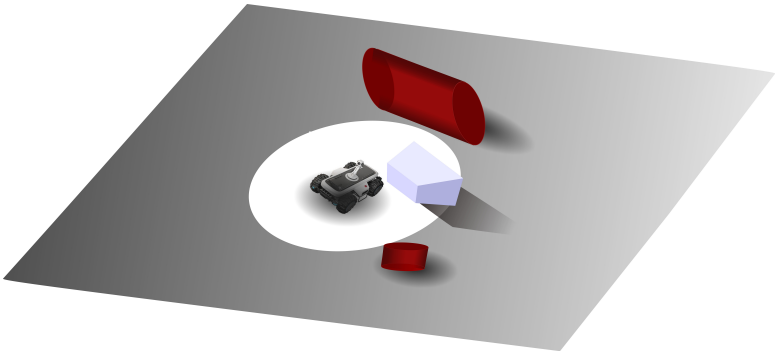


Figure 3.3: The white area around the robot shows its perception field. Any (parts of the) obstacles that fall within this perception field are known to the robot, while the robot is unaware of those obstacles that fall outside this field. If no safety measures are considered, by positioning its center on the borders of the perception field, the body of the robot may crash into the obstacles that are positioned close-by to the borders of its perception field and outside of it.

dynamically tuned via the time-varying bounds w_k^{rob} and $w_k^{\text{dyn,obs}}$, which we explain in the next section how to determine.

The robust TMPC optimization problem is in general complex and may thus become infeasible. Therefore, in case the tracking robust TMPC problem is deemed infeasible, it calls back to the heuristic motion panning system to ask updating the reference trajectories, and after that a new robust TMPC problem is solved. The constraints (3.3g)–(3.3j) are lower bounds and thus non-convex constraints, that determine a risk to obtain local minima. Therefore, the optimization problem of robust TMPC is in general nonlinear and non-convex. To address this issue, it may be solved using state-of-the-art global algorithms, e.g., genetic algorithm [255] or pattern search [256], using multiple starting points.

DYNAMIC ADJUSTMENT OF THE TUBE IN TMPC DUE TO DISTURBANCES AND PERCEPTION ERRORS

We decouple the evolution of the states of the robot due to its dynamics and due to the approaching dynamic obstacles, and independently incorporate the influence of these sources of uncertainties on the tube of robust TMPC. The values of w_k^{rob} and $w_k^{\text{dyn,obs}}$, computed at time step κ and for $k \in \mathbb{P}_\kappa$, in the worst case are determined via the following geometric sequences:

$$w_k^{\text{rob}} = \bar{w}^{\text{rob}} \sum_{i=0}^{k-\kappa-1} (1-\xi^{\text{rob}})^i \tag{3.4}$$

$$w_k^{\text{dyn,obs}} = \bar{w}^{\text{dyn,obs}} \sum_{i=0}^{k-\kappa-1} (1-\xi^{\text{dyn,obs}})^i \tag{3.5}$$

where $\xi^{\text{rob}} \in [0,1]$ and $\xi^{\text{dyn,obs}} \in [0,1]$ are the damping values for the deviation of the robot states and for the perception error regarding the position of the dynamic obstacles, respectively, and $\bar{w}^{\text{rob}}$ and $\bar{w}^{\text{dyn,obs}}$ are the upper bounds for the external disturbances

that impact the robot states and for the perception error of the robot, respectively. Based on these values, constraint tightening may be performed. Note that the damping values should carefully be tuned to provide a balanced trade-off between increased robustness and reduced conservativeness for robust TMPC.

3.6. CASE STUDY

Simulations were run to compare the performance of the proposed control architecture (called HP+TMPC, referring to integrated heuristic motion planning and robust TMPC) with two state-of-the-art methods, Horizon-based Lazy Rapidly-exploring Random Tree (HL-RRT*)[257] and COLREGS Artificial Potential Function (APF)[258]. These two state-of-the-art methods were selected due to their complementary strengths: On the one hand, COLREGS APF represents reactive and safety-focused navigation and offers explicit obstacle avoidance and target convergence mechanisms via attraction–repulsion fields. On the other hand, HL-RRT* exemplifies computationally efficient, heuristic-based planning and responsiveness suited for partially known environments. These methods reflect two critical attributes that our framework is designed to unify: (1) robustness and safety in dynamic, uncertain environments; and (2) real-time feasibility with scalable planning capabilities. Thus, this evaluation can safely be considered sufficient and representative for demonstrating the effectiveness of the proposed method within the scope of this study.

For all the simulations, the CPU used was a 4 core 2.5 GHz Intel® Core™ i7-4710MQ with 8GB of RAM memory and an integrated GPU of Intel® HD Graphics 4600. The operating system was Ubuntu 18.04.6 LTS, a 64-bit OS, with open-source drivers, where applicable. The simulations were done on MATLAB R2020b, where the parameters used have been made publicly available in the 4TU.ResearchData repository [259].

HL-RRT* is a path planning approach based on Rapidly exploring Random Tree (RRT*), which uses random sampling in its search space and builds up a tree with branches that connect the nearest points of the tree to each random sample, when this connection corresponds to a collision-free path. Once the target point is connected to the tree, the suitable path from the starting point to this target is selected. For enhanced computational efficiency, HL-RRT* uses a horizon-based strategy to guide the exploration, where the sampling is biased toward the points that are closer to the horizon and/or to the target. Moreover, HL-RRT* only checks the final candidate path for collision avoidance [260], [261].

COLREGS APF refers to the deployment of APF for navigation, complying with the rules of COLREGS [258], i.e., international regulations for preventing collisions at sea. APF steers the heading and velocity of the robot based on the vector that combines all attraction and repulsion (e.g., due to obstacles on the way) forces between the robot and its target. The core aspect of COLREGS APF used in this case study is a horizon-based collision-avoidance strategy, which makes the comparison with an MPC-based method more relevant.

A square-shaped environment of size 14 m × 14 m was simulated, considering **case 1** with 10 scenarios, each including 6 static and 5 dynamic obstacles, and **case 2** with 10 other scenarios, each including 8 static and 8 dynamic obstacles. In **case 2** in particular initial configurations and kinematics were designed for the obstacles such that a

temporarily infeasible problem would appear for the robot. The scenarios were carefully designed to ensure that reaching the destination for the robot was always feasible in the long term. The random variables (e.g., the external disturbance affecting the position of the robot) were different among the scenarios for each case (for details see the data repository [259]). Note that while the error in perceiving the position of dynamic obstacles and the deviation of the states of the robot due to external disturbances were bounded and randomly generated, the same values were used for different control methods in order to make the comparisons fair.

The simulations were run following two setups: (1) A computation budget (0.15 s per decision making for MPC and HL-RRT*, while APF does not in practice need this time budget, as it solves the problem almost in real time) was considered, where the simulations were terminated as soon as the budget was exhausted. (2) The three approaches ran until either the target was reached by the robot or reaching the target was deemed infeasible. The comparisons of the performance among the three approaches were with regards to the rate of success of the three methods (i.e., whether or not the robot reaches the target without any collisions and without falling into a livelock, e.g., circling) and the length of the path taken by the robot to the target. In setup (2), the overall mission time (i.e., the time taken by the robot to reach its target) was also compared.

To simulate the motion of each dynamic obstacle o, the following nonlinear equations were considered:

$$\begin{aligned} x^{\text{dyn,obs}}_{\kappa+1}(o) &= x^{\text{dyn,obs}}_{\kappa}(o) + \text{RK}\Big(v^{\text{dyn,obs}}_{x,\kappa}(o), c\Big), \\ v^{\text{dyn,obs}}_{x,\kappa+1}(o) &= v^{\text{dyn,obs}}_{x,\kappa}(o) + \text{RK}\Big(\alpha(x^{\text{att}}(o) - x^{\text{dyn,obs}}_{\kappa}(o)), c\Big), \end{aligned} \tag{3.6}$$

$$\begin{aligned} y^{\text{dyn,obs}}_{\kappa+1}(o) &= y^{\text{dyn,obs}}_{\kappa}(o) + \text{RK}\Big(v^{\text{dyn,obs}}_{y,\kappa}(o), c\Big), \\ v^{\text{dyn,obs}}_{y,\kappa+1}(o) &= v^{\text{dyn,obs}}_{y,\kappa}(o) + \text{RK}\Big(\beta(y^{\text{att}}(o) - y^{\text{dyn,obs}}_{\kappa}(o)), c\Big) \end{aligned} \tag{3.7}$$

with $\text{RK}(\cdot, c)$ Runge-Kutta 3/8 operator that integrates the given variable across one sampling time c. The initial positions and the velocities of the obstacles were sampled based on a uniform distribution, and the motions were around fixed attraction points with coordinates $[x^{\text{att}}(o), y^{\text{att}}(o)]^\top$ per obstacle o. For the constant multipliers α and β we considered:

$$\begin{aligned} \alpha &= 0.2\frac{1+4\eta}{v_x^{\text{rob,max}} - v_x^{\text{rob,min}} + \left\| x_0^{\text{dyn,obs}}(o) - x^{\text{att}}(o) \right\|}, \\ \beta &= 0.2\frac{1+4\eta}{v_y^{\text{rob,max}} - v_y^{\text{rob,min}} + \left\| y_0^{\text{dyn,obs}}(o) - y^{\text{att}}(o) \right\|} \end{aligned} \tag{3.8}$$

with $\eta \in [0,1]$ a random number per obstacle that is sampled from a uniform distribution, and $v_x^{\text{rob,max}}$, $v_x^{\text{rob,min}}$, $v_y^{\text{rob,max}}$, $v_y^{\text{rob,min}}$ the maximum and minimum velocities of the robot in the x and y directions, respectively. These choices allow for relative velocities for the robot and the dynamic obstacles that result in obstruction of the path of the robot, thus evaluating the given approaches based on relevant, meaningful simulations. We assume that the robot uses a filter to estimate the motion of the dynamic obstacles,

simply using linear approximation for the velocities. Such an approximation, considering the prediction horizon of $N^{\mathrm{p}} = 5$ used in the case study, results in a bounded error of maximum 3.57%, which is acceptable for the simulations.

3.7. SIMULATION RESULTS

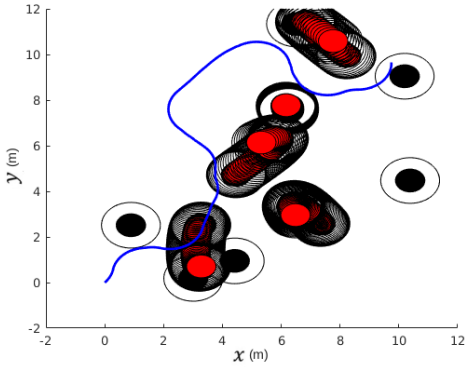
In this section the results of the simulations are presented. Due to the large number of experiments, we have presented only a few representative cases: Figures 3.5-3.6, Figures 3.7-3.8, and Figures 3.9-3.10 correspond to deploying, respectively, HP+TMPC, HL-RRT*, and APF for **case 1** and **case 2**. The dash-dotted blue and solid green trajectories in the plots correspond to, respectively, setup 1 and setup 2. Static and dynamic obstacles are illustrated as solid black and red circles, respectively. In the plots on the right-hand side of these figures, the realized distance of the robot from the closest obstacle during the simulation, as well as the minimum safety radius (black dashed lines) are shown.

Tables 3.2 and 3.3 show the results for the path length and mission time of the robot for **case 1** and **case 2**, respectively. A dash symbol is used to indicate mission failure, i.e., the robot did not reach the target, due to either collisions or falling into livelocks.

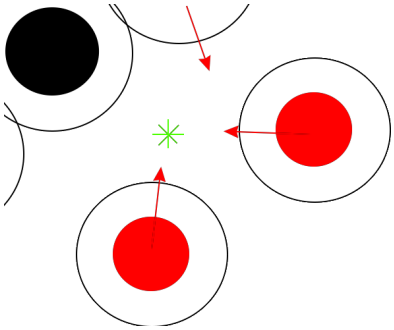
3.8. DISCUSSION OF THE RESULTS

In this section, the results are discussed. In general, APF failed to perform satisfactorily in both **case 1** and **case 2**, showing only a single success from the 10 scenarios for both setup 1 and setup 2. Failures occurred because the robot got stuck in an 8-shaped path (see Figures 3.9a and 3.10a). This turned out to be related to the tuning of APF, since after re-tuning it, the livelock disappeared and a trajectory towards the target was found (see Figure 3.4a). This behavior was ultimately linked to the tuning of the hyperparameters of APF, since after re-tuning them the livelock was eliminated and a feasible trajectory to the target was found (see Figure 3.4a). This, however, highlights a fundamental limitation of APF and similar approaches, i.e., their high sensitivity to hyperparameter settings. This sensitivity stems from the way APF combines attractive and repulsive potential fields, i.e., using fixed weights and influence radii, to compute motion commands. Any small changes in these parameters can significantly distort the resulting gradient field and lead to undesired behaviors, such as livelocks or oscillations, particularly in complex or dynamic environments. In practice, manual re-tuning of these parameters is often infeasible, especially in unstructured or time-critical scenarios. While online adaptation or learning-based tuning mechanisms will theoretically address this issue, such solutions introduce new practical and computational burdens that raise serious concerns about reliability and applicability in safety-critical domains, including SaR robotics.

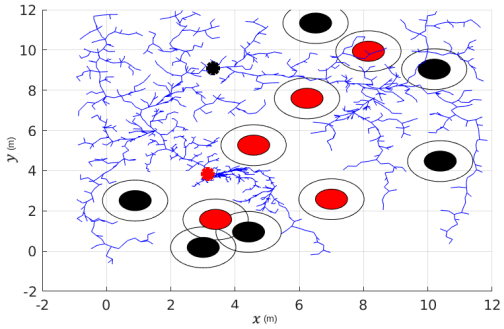
From Table 3.2, HP+TMPC had a higher success rate than HL-RRT* for setup 1. All failures of HL-RRT* were due to crashing of the robot into obstacles, especially in a specific scenario designed to increase the risk of crashing (see Figure 3.4b), where the robot was placed between two dynamic obstacles or one dynamic and one static obstacle. In this case, HL-RRT* did not find an alternative lower cost path in time that keeps the robot safe. These failures reflect key structural limitations of HL-RRT*. Specifically, this planner lacks predictive modeling of obstacle motion and therefore, unlike robust TMPC,



(a) Trajectory towards the target (shown in blue) generated by APF following sub-optimal re-tuning of its hyperparameters.



(b) Crushing motion scenario. Three dynamic obstacles (with arrows showing their movement) move towards the robot (star).



(c) Tree structure (blue) of the HL-RRT* with a high computational budget, at the time of the berth in Figure 3.7a. Collision checking is done from the current position (red star) to the end of the horizon (black star).

Figure 3.4: Particular conditions in the simulations of the case study.

Table 3.2: Results in terms of the path length (m) and the mission time (s) for **case 1**, setup 1 (top table) and setup 2 (bottom table).

	HP+TMPC		HL-RRT*		APF	
scenario	path	time	path	time	path	time
1	18.8	49.8	-	-	-	-
2	20.1	55.7	20.0	44.5	-	-
3	22.3	50.5	-	-	-	-
4	17.9	53.3	19.3	40.2	-	-
5	16.5	41.1	19.8	44.2	-	-
6	21.0	55.2	21.9	52.1	-	-
7	15.2	32.3	15.0	30.9	15.4	30.8
8	22.4	49.7	-	-	-	-
9	20.0	59.5	24.9	22.9	-	-
10	17.8	39.3	19.2	39.2	-	-
mean	19.2	48.6	20.0	44.2	15.4	30.8
standard deviation	2.40	8.50	2.99	8.94	-	-

	HP+TMPC		HL-RRT*		APF	
scenario	path	time	path	time	path	time
1	15.5	40.6	26.4	50.7	-	-
2	16.8	37.1	19.4	41.2	-	-
3	21.0	52.1	24.0	46.3	-	-
4	16.8	37.6	18.7	35.0	-	-
5	16.5	42.7	17.6	32.5	-	-
6	18.2	38.1	21.0	44.5	-	-
7	15.0	31.8	16.0	30.9	15.4	30.8
8	19.1	50.6	22.9	50.4	-	-
9	18.2	41.1	20.5	42.5	-	-
10	16.9	41.9	18.3	33.3	-	-
mean	17.4	41.4	20.5	40.7	15.4	30.8
standard deviation	1.78	6.13	3.18	7.74	-	-

cannot anticipate future constraint tightening. Its reliance on lazy collision checking and a fixed-time planning horizon further exacerbates the problem. Once the robot enters a narrow or transiently safe corridor, the planner may fail to re-plan in time, or may commit to paths that become infeasible during execution. In summary, unlike feedback-based methods, such as robust TMPC, HL-RRT* offers no mechanism for online correction or constraint adaptation. These make the planner vulnerable in scenarios where reactive safety is critical.

Instead, HP+TMPC avoided crashing into the obstacles in this challenging scenario, by either considering an obstacle belt (as a result of modeling the motion of dynamic obstacles within the prediction horizon) to avoid, or setting a reference trajectory outside

Table 3.3: Results in terms of the path length (m) and the mission time (s) for **case 2**, setup 1 (top table) and setup 2 (bottom table).

	HP+TMPC		HL-RRT*		APF	
scenario	path	time	path	time	path	time
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	19.1	47.8	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	20.9	51.0	-	-	-	-
7	-	-	-	-	-	-
8	-	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
mean	20.0	49.4	-	-	-	-
standard deviation	1.26	2.26	-	-	-	-

	HP+TMPC		HL-RRT*		APF	
scenario	path	time	path	time	path	time
1	19.2	50.7	-	-	-	-
2	16.2	43.1	-	-	-	-
3	17.4	41.0	-	-	-	-
4	18.4	45.0	-	-	-	-
5	19.5	52.0	-	-	-	-
6	17.2	48.7	-	-	-	-
7	18.4	46.8	-	-	-	-
8	-	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
mean	18.0	46.8	-	-	-	-
standard deviation	1.16	4.01	-	-	-	-

the unsafe region.

For setup 2, however, HL-RRT* always handled this high-risk scenario, but with a high computational cost. In fact, using HL-RRT* the robot only moves across a path that leads to a lower cost, in this case a path that is closer to the target. Thus, unless a node was found closer to the target than the current position of the robot, it simply refused to move. Since in setup 2 there was always time to find such a node, HL-RRT* showed a 100% success rate.

Comparing the path lengths, HL-RRT* generally took a longer path than HP+TMPC, while the path of HL-RRT* was generally shorter in setup 2 compared to the same approach used in setup 1. In a few cases, HL-RRT* took a slightly shorter path than HP+TMPC,

due to its random nature. See the path made by HL-RRT* in Figure 3.7a, where a wide berth is made to avoid the middle obstacles, partly due to the sampling and partly because of the local cost propagation in the tree, a limitation of RRT*. This is confirmed considering the tree that was used at the time of the berth (see Figure 3.4c).

For HP+TMPC, setup 1 resulted in paths that were approximately 10% longer on average than for the same approach used in setup 2. This can be explained via Figure 3.5a, where a small berth for setup 1 is observed that is avoided in setup 2 (see Figure 3.6a). This is linked to the limited knowledge about dynamic obstacles within a limited computational window. In fact, for setup 2 HP+TMPC found a lower cost path by slightly changing the course and speeding up, and it almost always found a local minimum, while the optimization in setup 1 sometimes stopped prematurely.

For the mission time, in setup 1 in various cases when HL-RRT* has found a path, this path has generally resulted in a smaller mission time than with HP+TMPC. In setup 2, in almost half of the cases HL-RRT* wins in achieving a smaller mission time, while in the other cases HP+TMPC wins, with, on average, 11% reduced time for the winning approach in both cases. Comparing, for instance, Figures 3.5 and 3.7, it is clear that HP+TMPC has opted for the shortest path to the target, but since this requires moving closely to various obstacles, it may have compromised its speed for remaining crash-free, whereas by taking a longer path, HL-RRT* has avoided the obstacles significantly. This is mainly due to the formulation of the optimization problem for HP+TMPC (see (3.2)), where no explicit term for reducing the mission time has been considered in the objective function, but rather the controller is asked to minimize its distance from a reference trajectory that, according to the heuristic motion planning system, provides the shortest path to the target. This implies that a potential point of improvement for reducing the mission time of HP+TMPC is to include the mission time as an additional term in the objective function of robust TMPC.

Based on Table 3.3, for **case 2**, similarly to APF, HL-RRT* failed to show any success in reaching the target point. In particular in setup 2, none of the failures of HL-RRT* was due to colliding with any obstacles, but was mainly because, even with the larger computational budget, the algorithm failed to find any feasible paths. From Figure 3.8a, HL-RRT* explores a variety of options, which are quickly dismissed, because of the lack of dynamic prediction for the moving obstacles. The main reason for the back-and-forth motions is that HL-RRT* followed a path for a while, which turned out to be infeasible later. The consistent failure of HL-RRT* in **case 2**, even under extended computational budget (setup 2), can be attributed again to fundamental limitations of this algorithm in dynamic environments. First, HL-RRT* lacks any mechanism for predicting the motion of dynamic obstacles, causing it to generate paths that are quickly invalidated during execution. Second, its horizon-based sampling strategy and cost bias tend to prune exploratory branches that may be necessary for avoiding moving obstacles, especially in constrained or deceptive regions. Third, the lazy collision checking of this algorithm delays the detection of invalid paths, leading to repeated re-planning cycles without structural adaptation. These factors combined to make the problem effectively infeasible for HL-RRT* in the more complex, dynamic scenarios of **case 2**.

The failures of HP+TMPC in **case 2** were primarily due to the inability of the optimization solver to find a feasible control solution within the allowed number of itera-

tions. Specifically, the heuristic planner is unaware of future obstacle motion and may steer the robot into narrow regions (e.g., Figure 3.4b), where obstacle trajectories eventually close in, forming a so-called 'crushing zone'. Once inside this zone, the robust TMPC controller receives an infeasible problem, as in fact no admissible sequence of control inputs within the velocity and safety constraints can lead to a collision-free trajectory. In these situations, the optimization solver often reaches the iteration limit without success, especially when the prediction horizon is too small to find a viable escape plan for the robot.

In setup 2, we observed additional cases where the controller generated overly aggressive inputs, which lead to a velocity constraint violation. This occurred when robust TMPC attempted to recover from a deteriorating situation introduced by the planner, and accordingly pushed the system close to the limits of feasibility. Without a sufficiently large prediction horizon or relaxed constraints, this led to constraint violations or solver failure.

These observations point to a fundamental limitation of the decoupled heuristic+MPC structure. The heuristic planner lacks awareness of dynamic feasibility, and the controller has limited authority to correct flawed plans, particularly under real-time constraints. Addressing this limitation requires tighter integration, larger prediction horizons, or more predictive planning mechanisms.

To further investigate these failure modes, we tested variations with increased prediction horizons and larger solver budgets. Both mitigated the failure cases, suggesting that windows with longer look-ahead planning improve safety, but this comes at the cost of increased computation time. This highlights the inherent trade-off between prediction depth (safety) and responsiveness (reactivity) that should be carefully balanced in real-time applications.

Under setup 1 with limited computational budget, occasional solver timeouts were observed due to the strict 0.15 s time limit. This particularly occurred in scenarios with dense obstacles or high dynamic constraints (e.g., Figure 3.4b). Nevertheless, feasibility was preserved in the majority of cases because (i) the solver returned the best feasible iterate available at timeout, and (ii) tightened constraints within the optimization loop of robust TMPC inherently maintained safety margins.

For unlimited computation budget (setup 2), whenever the problems were structurally feasible, the optimization algorithm always converged, considering a threshold of 10^{-6} over an average objective function cost in range 20–200.

Overall, these results indicate that, under the given computational resources and simulation setup, robust TMPC achieved real-time feasibility. Given the consistent convergence of the optimization solver in the unlimited computation budget setting (i.e., setup 2) and its robustness under limited computation budget setting (i.e., setup 1), these findings suggest that, with hardware comparable to or exceeding our simulation platform, the proposed architecture is well-positioned for making the next step to real-world deployment. Naturally, transitioning from computer-based simulations to physical experiments will introduce new challenges (e.g., sensor noise, unmodeled dynamics, onboard resource constraints) that will require further validation and potential adaptations to ensure reliable real-time performance.

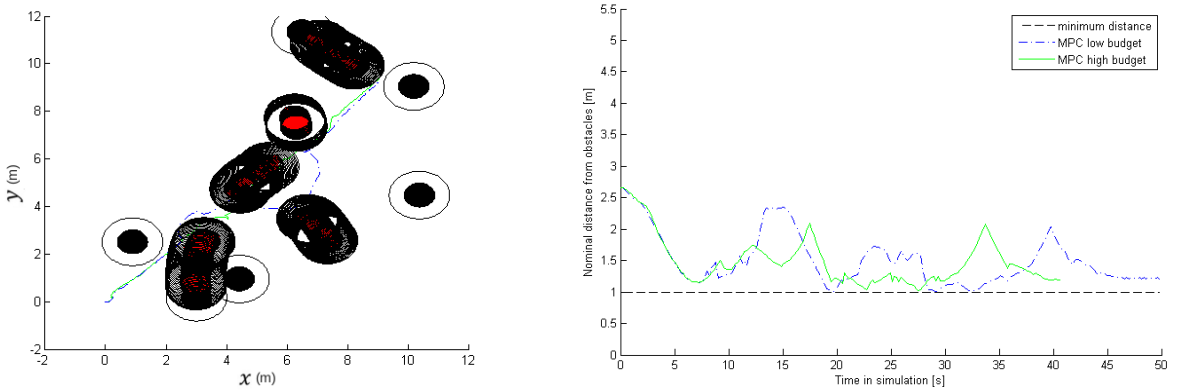
3.9. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

In this chapter, we proposed a novel control architecture for motion planning and reference tracking of autonomous robots in dynamic cluttered environments, based on a modified version of a heuristic motion planning method [193] and Robust Tube-based Model Predictive Control (robust TMPC). In a case study, we compared our proposed approach to two state-of-the-art methods and showed, especially for complex scenarios, to have similar or significantly higher success rates and shorter path lengths with the proposed control architecture, while producing collision-free trajectories despite multiple moving obstacles.

In the future, the proposed control architecture will be validated for different robot models, and for more variations in the motion of the obstacles, where proper filters should be merged into the control architecture to estimate the motion of these obstacles in real time. In addition, more scenarios will be simulated, including environments with varied shapes and obstacle configurations. In missions where time minimization is critical, such as emergency response, evacuation support, or medical delivery, the objective function of robust TMPC can be adapted to explicitly penalize time-to-go or to reward forward progress toward the target. This adaptation shifts the trade-off in favor of mission speed. Exploring such reformulations represents a promising direction for extending the applicability of the proposed framework to a broader range of real-world scenarios. This control architecture should further be extended to multi-robot systems, with potentially heterogeneous characteristics. Expanding the comparative simulations to include more recent methods, such as deep reinforcement learning, imitation learning, or hybrid planning approaches, will provide a broader benchmarking context. This constitutes a relevant future direction, especially for extending our framework to more complex or data-driven navigation scenarios. Accordingly, as part of future work, the comparative analyses should be extended by including more recent learning-based and hybrid planning methods to further benchmark the proposed framework against a broader range of navigation strategies. Finally, real-life experiments should be performed to validate the effectiveness of the control architecture beyond computer-based simulations.

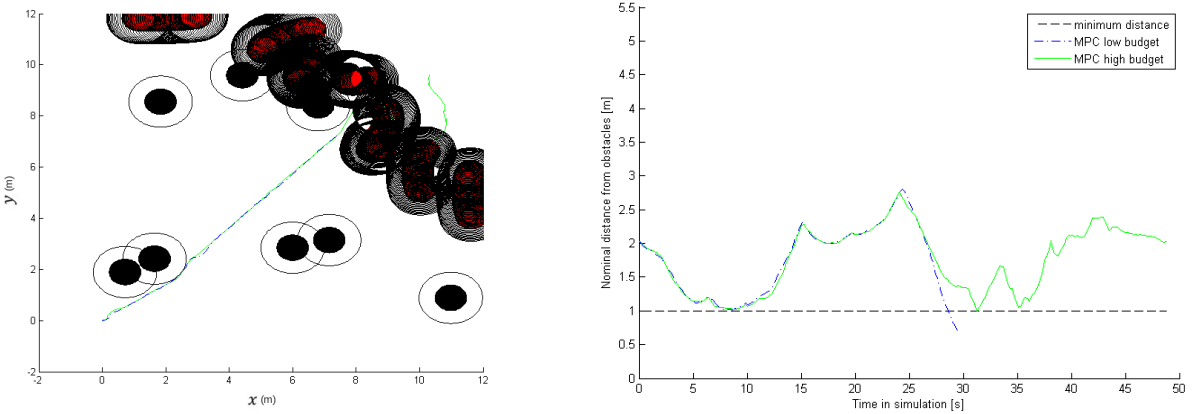
DATA AVAILABILITY

The data and parameters used in the simulations are publicly available in the 4TU.ResearchData repository: Baglioni, M. Tables with parameters values underlying the publication: Enabling robots to autonomously search dynamic cluttered post-disaster environments. https://doi.org/10.4121/aa7528da-0986-453c-b196-4277a2db4daa (2024).



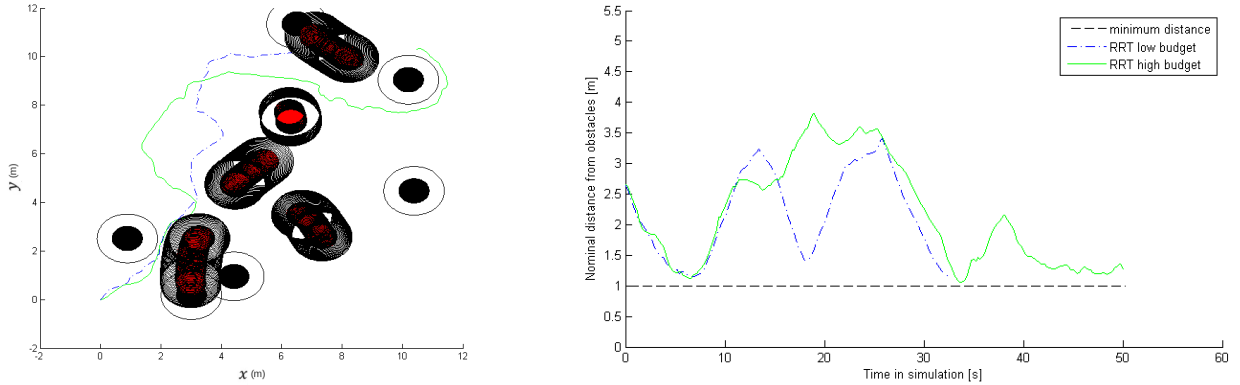
(a) (b)

Figure 3.5: (a) Sample robot paths for **case 1**, setups 1 and 2 (called low and high budget respectively) for the **proposed control architecture** and (b) the distance of the robot from the closest obstacles.



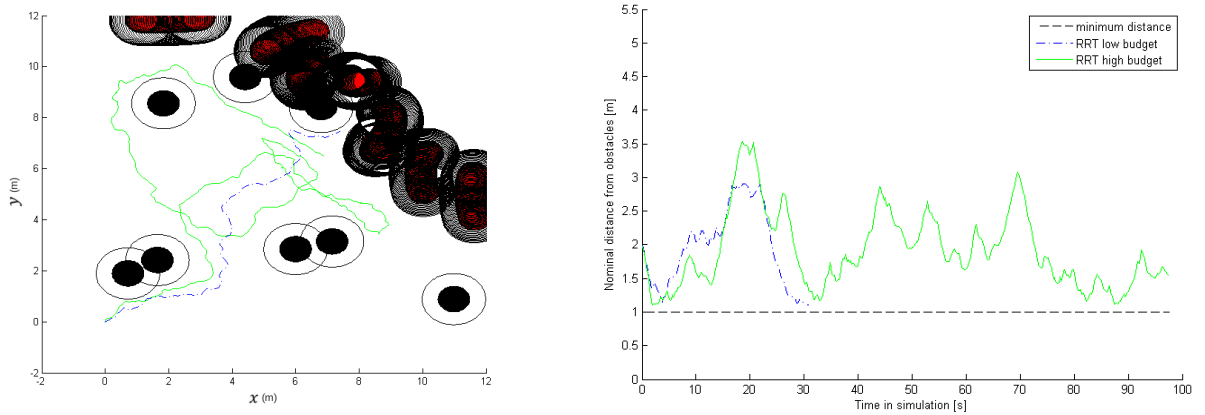
(a) (b)

Figure 3.6: (a) Sample robot paths for **case 2**, setups 1 and 2 (called low and high budget respectively) for the **proposed control architecture** and (b) the distance of the robot from the closest obstacles.



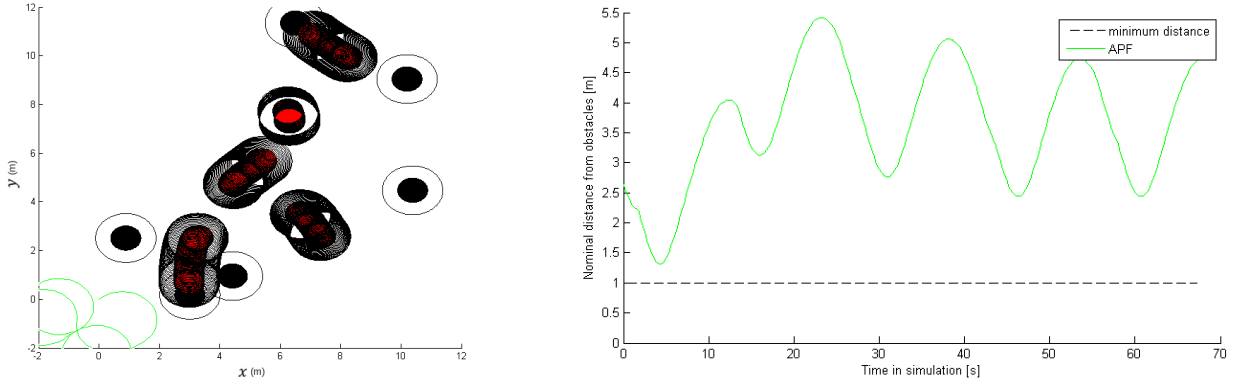
(a) (b)

Figure 3.7: (a) Sample robot paths for **case 1**, setups 1 and 2 (called low and high budget respectively) for **HL-RRT*** and (b) the distance of the robot from the closest obstacles.



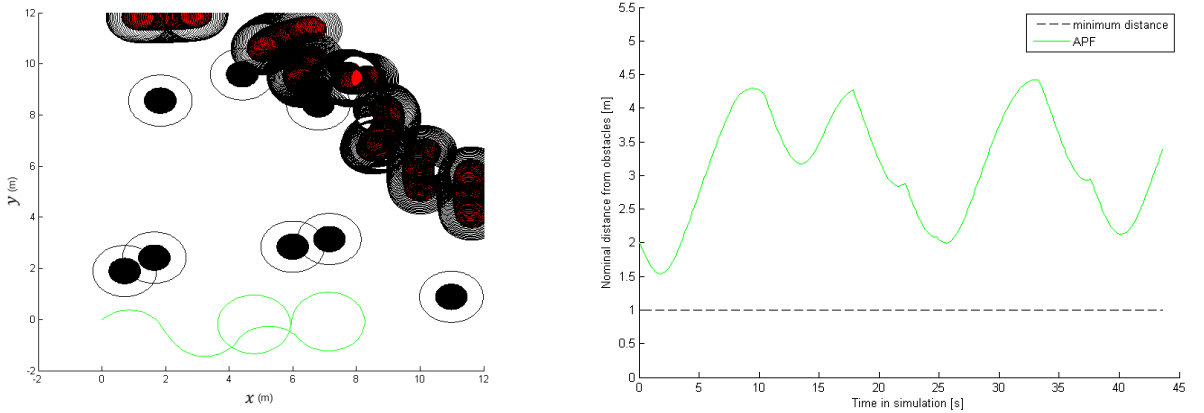
(a) (b)

Figure 3.8: (a) Sample robot paths for **case 2**, setups 1 and 2 (called low and high budget respectively) for **HL-RRT*** and (b) the distance of the robot from the closest obstacles.



(a) (b)

Figure 3.9: (a) Sample robot paths for **case 1** for **APF** and (b) the distance of the robot from the closest obstacles.



(a) (b)

Figure 3.10: a) Sample robot paths for **case 2** for **APF** and (b) the distance of the robot from the closest obstacles.

4

Camera-based Mapping in Search-and-Rescue via Flying and Ground Robot Teams

Search-and-Rescue (SaR) is challenging, due to the unknown environmental situation after disasters occur. Robotics has become indispensable for precise mapping of the environment and for locating the victims. Combining flying and ground robots more effectively serves this purpose, due to their complementary features in terms of viewpoint and maneuvering. To this end, a novel, cost-effective framework for mapping unknown environments is introduced that leverages YOLO (You Only Look Once) and video streams transmitted by a ground and a flying robot. The integrated mapping approach is for performing three crucial SaR tasks: Localizing the victims, i.e., determining their position in the environment and their body pose, tracking the moving victims, and providing a map of the ground elevation that assists both the ground robot and the SaR crew in navigating the SaR environment. In real-life experiments at the CyberZoo of the Delft University of Technology, the framework proved very effective and precise for all these tasks, particularly in occluded and complex environments.

Parts of this chapter have been published in Machine Vision and Applications **35**, 117 (2024) [3].
M. Baglioni contributed to the methodology and the experiments, assessed the results, supervised the MSc student involved, and reviewed the final draft of the chapter. B. Esteves Henriques performed the conceptualization, designed the experiments, contributed to methodology and results, and wrote the first draft. A. Jamshidnejad contributed to methodology, assessed the results, edited the final draft, and performed supervision.

4.1. INTRODUCTION

NUMBER and severity of natural disasters have risen dramatically in recent decades. Between 1991 and 2005, nearly 90% of disaster-related deaths and 98% of people affected by disasters belonged to low-income countries [157]. The survival rate of trapped victims drops from 91% in the first 30 minutes to 36.7% by the end of the second day [262]. Therefore, it is vital to make Search-and-Rescue (SaR) operations both affordable and time-efficient. SaR has become increasingly augmented with robotics in the last 20 years [23]. Besides traversing hazardous environments via their sensors, robots scan their surroundings rapidly and autonomously. In this chapter, we introduce a novel, cost-effective framework for mapping unknown environments via SaR robots that effectively and efficiently combines the strengths of ground and flying robots when teaming up for SaR missions.

4.1.1. MOTIVATIONS

Table 4.1 shows a qualitative cost comparison for conventional sensors used in SaR robotics. More expensive sensing approaches, e.g., thermal imaging, may still fail to detect humans in high-temperature environments, e.g., in case of fire [263], [264]. The main disadvantage of radar and LiDAR sensors is their very high costs. In addition, the applications of RGB images are wider, i.e., while radar and LiDAR provide range information that can be used for obstacle avoidance or for mapping, images taken by RGB cameras can be used for computer vision tasks, including detection and tracking of objects and SaR victims, or extraction of information for training machine learning algorithms and neural networks [265]–[267].

Special sensors, e.g., Doppler-shift sensors for detecting humans based on the motion of their lungs, their heartbeat, or typical Doppler signatures of motion may fail in case of stationary targets or in distinguishing humans from other moving objects [268]–[270].

Acoustic sensors in SaR applications can be subject to various disturbances and noise from the environment. Flying robots are specially impacted by this issue, because of the noise that their propellers create [93].

A main motivation for using vision-based algorithms that rely on inputs from standard RGB cameras is the affordability of these cameras (see Table 4.1) and the capability of such algorithms to perform well in very challenging SaR conditions, thanks to the recent advancements in image processing. In fact, an improved performance with respect to the state-of-the-art is achieved by incorporating the latest advancements in computer vision into the proposed framework. In particular, YOLO (You Only Look Once), which is used in this chapter, has emerged as one of the most promising object detection algorithms and has proven to achieve real-time object detection with high accuracy levels and reduced computational resources.

Our main motivations for teaming up flying and ground robots are the following: On the one hand, with low costs, flying robots provide access to aerial perspectives and are able to swiftly cover expansive areas in a short time. Their imaging quality, however, may be compromised due to the tilting of their cameras while flying. Moreover, they may fail to access and perform properly in very confined spaces and corridors. On the other hand, while ground robots navigate rugged terrains at a slower pace, with a

Sensor type	Cost
RGB cameras, depth cameras	Low
Doppler-shift sensors, acoustic sensors	Low
2D radar, 2D LiDAR, thermal imaging	Medium
3D radar, 3D LiDAR	High

Table 4.1: A qualitative cost comparison of SaR medium-segment sensors based on [265]–[267], [271]–[275].

more restricted field of view, they excel in precise imaging, accessing confined spaces, and transporting heavy payloads. Moreover, as it is illustrated in Figure 4.2, the distinct viewpoints that are captured by flying and ground robots can properly complement each other.

4.1.2. BACKGROUND

The majority of the state-of-the-art literature that considers collaborative teams of flying and ground robots focuses on the problem of navigation of these robots in SaR environments (see, e.g., [276], [277]). An important problem in SaR missions, however, is to map the unknown environment, and to detect the individuals who are in distress and to estimate their position and pose [278].

The only papers that have focused on the simultaneous detection and localization of objects in SaR scenarios, using You Only Look Once (YOLO), include [279] and [280]. Authors in [279] use the Scale-Invariant Feature Transform key point matching algorithm in order to determine the corresponding points in the pictures that have been taken from different viewpoints. Afterwards, using a homography matrix, the coordinates are transformed from one frame to the other. The approach used in [280] differs in determining the corresponding points within distinct frames, where trigonometry is used.

With recent advances in deep learning, victim detection through image processing has gained increased attention. Convolutional Neural Networks (CNNs) play a vital role in such detection methods [281]–[283]. In particular, the CNN YOLO has proven to be very effective in object detection [284]. For instance, in [279] pre-trained deep learning models based on YOLO are used, in order to detect objects in a flooded area. Authors in [280] apply YOLOv5 for detecting and localizing the victims in an outdoor SaR environment. In [285], YOLOv4-tiny (a compact version of YOLOv4) is used to detect the victims and their poses. In addition, the object detection algorithm YOLO is used to detect SaR victims via flying robots in [79], [286], [287]. In all these works, the authors compare to other existing object detection algorithms and show an improvement in performance (including accuracy, speed, and low false detection rate), using YOLO. In [288] YOLO is also used in thermal imaging, for the same purpose of victim detection, while in [289] it is used for object detection in Air-Sea Rescue (ASR). A common drawback of these approaches is that they do not integrate object detection with target tracking, as done in this chapter.

Moving victims should also be tracked in order to save them in a timely way. Thus, state estimation methods, e.g., the Kalman filter (and more advanced versions including the extended and unscented Kalman filter), have been proposed to estimate the trajec-

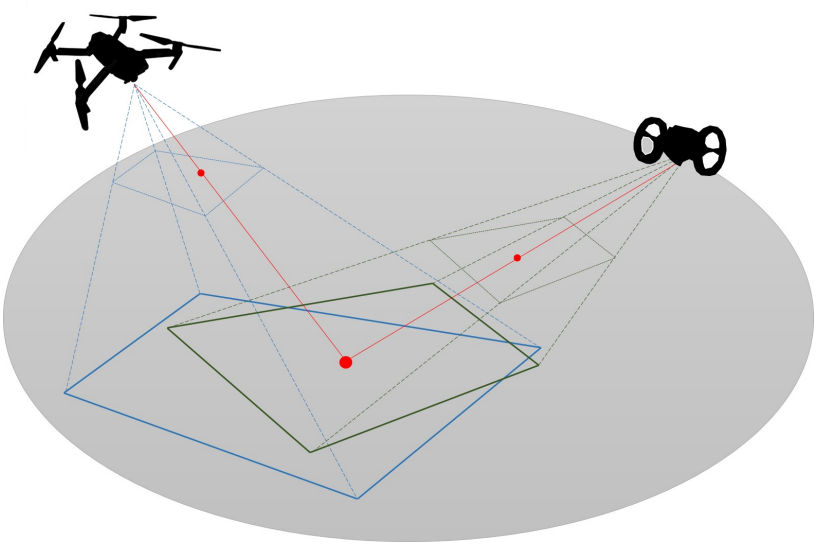


Figure 4.1: Combining a flying and a ground robot with their different specifications and area coverage for mapping unknown environments.

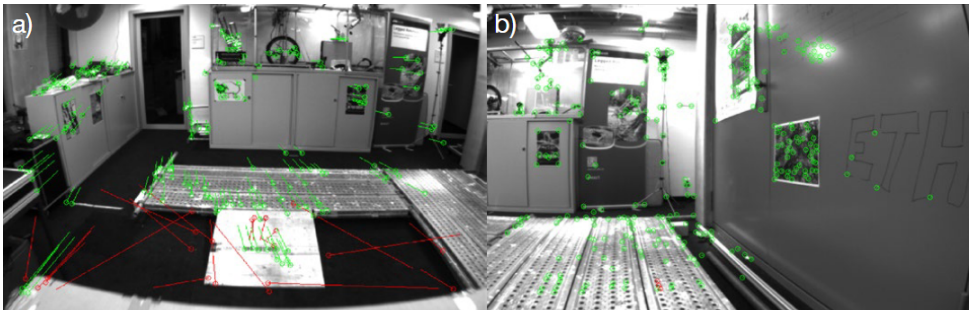


Figure 4.2: Different viewpoints for a flying (left) and a ground (right) robot, retrieved from [276].

tory of moving victims [290], [291]. Regarding victim tracking in ASR, in [292] a Kalman filter is used in a model to predict the position of a vessel in the presence of uncertainties given by the water motion. In [293], instead, an Adaptive Unscented Kalman filter is used to integrate the measurements of Inertial Measurement Units (IMUs) and Global Positioning System (GPS) receivers to localize rescuers in dense forests during SaR operations. They show that their algorithm outperforms other types of Kalman filters in terms of positioning accuracy and reduced prediction errors. In [294], an approach is proposed that leverages YOLO for victim detection in thermal imaging and a Kalman filter for tracking their trajectories based on position measurements. All these approaches, however, consider just one single robot, either grounded or flying, while the synergy of multiple sources of data is not addressed.

Another crucial piece of information in SaR is the traversability and elevation of the

ground of the terrain. Image processing can be used for such applications as well. In particular, authors in [295] present an approach for extracting the ground characteristics (e.g., the roughness, slope, discontinuity, hardness) from images using neural networks. Moreover, in [296] an approach based on stereo images and LiDAR (Light Detection And Ranging) data is used to determine forest terrain elevation and canopy height. The novel proposed method is validated with real-life experiments. In [297], instead, an approach is proposed to determine the elevation of the first floor of buildings, with the aim of mitigating flood risks. They use YOLO to detect doors and windows, and extract the first floor elevation from Light Detection And Ranging (LiDAR) point clouds data. These approaches are affine with estimating the terrain elevation of a SaR environment, but doing this precise task through image processing, as performed in this chapter, is still not addressed in the literature.

4.1.3. MAIN CONTRIBUTIONS AND STRUCTURE OF THE CHAPTER

We propose new time and cost-efficient approaches for mapping unknown SaR environments, based on the fusion of the knowledge that is deduced from images that are taken separately, via autonomous flying and ground robots. The mapping includes localization of and trajectory tracking for victims, and estimating the ground elevation. The main contributions of this chapter are:

1. We extend the pose estimation module of YOLO, in order to estimate the coordinates of the unobserved key points of the body of a victim from an image, based on body proportions and symmetry. Accordingly, YOLO is leveraged to estimate the distance of the victim from the ground robot.

2. YOLO is employed in streams captured from a flying and a ground robot, along with a localization algorithm, in order to map the detected points into real-world coordinates, and to fuse them using a Kalman filter, in order to estimate and track the trajectory of a moving victim.

3. An algorithm is proposed for estimating the elevation of the terrain, using YOLO and homography estimation.

Real-life case studies have been designed and performed in order to validate the three approaches explained above.

The rest of the chapter is organized as follows. Section 4.2 explains the proposed methodologies. Section 4.3 describes the setup and implementation of the case studies that have been carefully designed to validate the developed approaches using real-world data. Then the results of the case studies are presented and discussed. Finally, Section 4.4 concludes the chapter and provides suggestions for relevant future research.

4.2. PROPOSED METHODS

This section is structured in four parts, each explaining one SaR task. The first two parts will address the pose estimation and the victim localization. The third part will address the victim tracking approach using a Kalman filter. The last part will address the terrain elevation mapping. These four tasks are achieved by teaming up flying and ground robots.

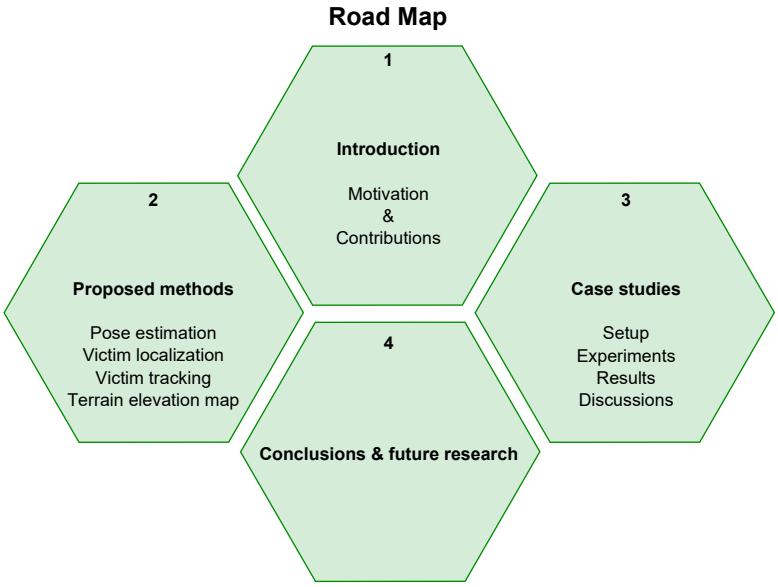


Figure 4.3: Road map of the chapter.

Our methods are based on YOLOv8n [298], since it significantly speeds up the class detection process and maintains high accuracy. In a single pass through a neural network, YOLO identifies both the bounding boxes and the probabilities for objects to belong to specific given classes. In particular, YOLOv8n (i.e., the nano version model) is chosen, since it is faster than other models. For training, the Common Objects in COntext (COCO) dataset, comprising 330,000 diverse images, has been used. From these images, 200,000 are annotated for object detection, segmentation, and captioning tasks across 80 object categories. Annotations encompass bounding boxes, segmentation masks, and captions, that enhance the precision and versatility of the model.

Figure 4.4 shows the output of YOLOv8n for the front and top views of a victim. The bounding boxes are drawn around the detected object, and a confidence score from $[0,1]$ is attributed to each class for that object.

4.2.1. POSE ESTIMATION

This section explains how the pose estimation of the victims works. In order to precisely locate the victims and to estimate their (health/physical) status, awareness about the body pose of the victim is needed. YOLOv8n-Pose has been trained for pose estimation on a COCO dataset, which contains 200,000 images labeled with 17 key body points [298]. Figure 4.5 shows the output of YOLOv8n-Pose for the front view of a victim. The body key points, given in a pre-defined order, include: nose, eyes, ears, shoulders, elbows, hands, hips (the two side points), knees, and feet. These labels are recorded in a list, which we call list "L". Some of these key points may not be visible in or detectable from the images that are received from a SaR robot, while their position is important for the estimation of the pose and distance of the victim from the robot. There-

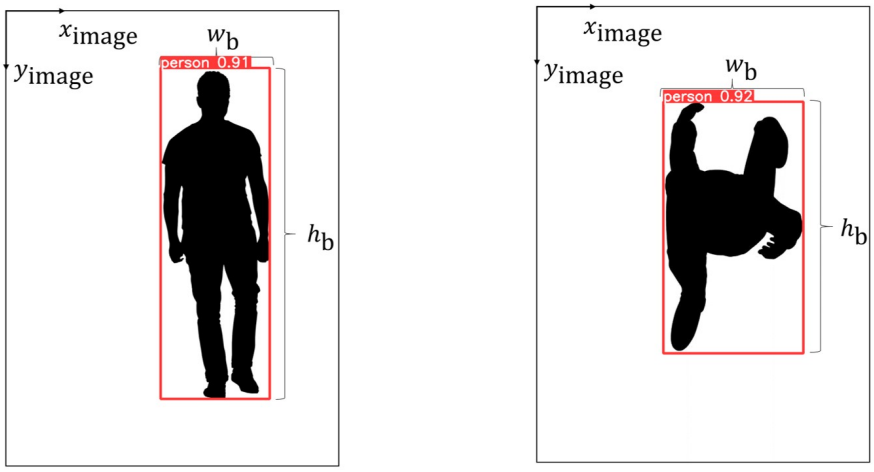


Figure 4.4: YOLOv8n output for front (left-hand plot) and top (right-hand plot) views of a victim.

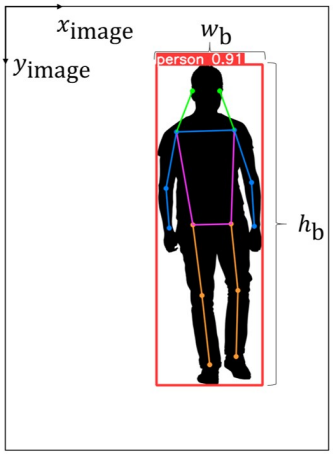


Figure 4.5: YOLOv8n-Pose is extended to perform pose estimation, where based on the previous output of the algorithm, the positions of the body key points are estimated.

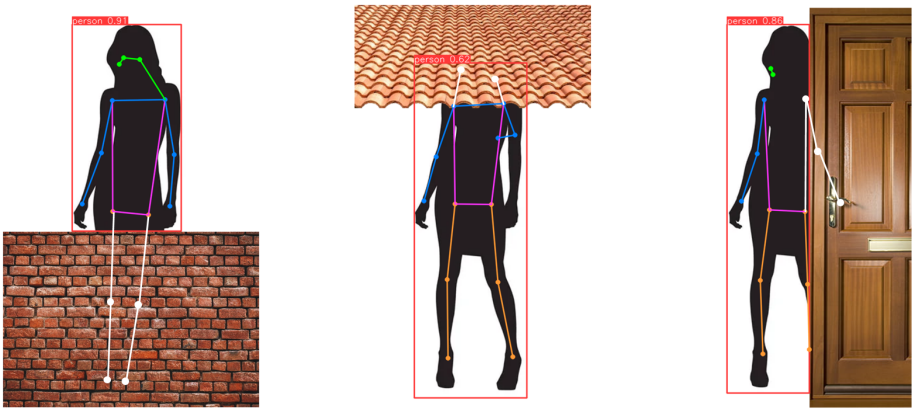


Figure 4.6: Extended YOLOv8n-Pose for locating the missing key points (shown in white) based on typical body proportions and symmetry.

Algorithm 2 Determining the position of all the missing key points

L: An ordered list of n^{keypoint} key point labels (i.e., nose, eyes, ears, shoulders, elbows, hands, hips, knees, feet)
K: An ordered list, with the same size and order as L, including the positions corresponding to the key points that are detected within a captured image, with $K[i]$ the i^{th} coordinate pair in list K
$n^{\text{iteration}}$: Pre-set number of iterations of the algorithm
i and k: Index variables
estimate_keypoint$(\cdot,\cdot)$: A function that receives as input the list K and the index i corresponding to an element in K that is empty, and assigns as output a pair of coordinates to fill in the empty element $K[i]$

for $k \leftarrow 1$ **to** $n^{\text{iteration}}$ **do**
    for $i \leftarrow 1$ **to** n^{keypoint} **do**
        if $K[i]$ is **None then**
            $K[i] \leftarrow$ estimate_keypoint(K, i)
        end if
    end for
end for

return K

Algorithm 3 Function `estimate_keypoint(·,·)`

Input: K, i

$n^{\text{dependency}}[i]$: Number of all sets of dependencies that have been defined for key point $K[i]$
R$[i]$: An ordered list composed of $n^{\text{dependency}}[i]$ sets, each including one set of the dependencies of key point $K[i]$
W$[i]$: A list composed of $n^{\text{dependency}}[i]$ sets of weights for the dependency key points, where the order of the sets and the order of the elements within each set are the same as in R$[i]$
W$[i,j,\ell]$: The weight that is deduced from the ℓ^{th} element within the j^{th} set of W$[i]$
D$[i,j,\ell]$: The position of the ℓ^{th} key point within the j^{th} set for list R$[i]$ where this position is deduced from list K
$n^{\text{set}}[i,j]$: Number of the key points within the j^{th} set of the dependencies of key point $K[i]$
j: An index variable
w: An auxiliary variable
b: A binary variable

$K[i] \leftarrow (0,0)$, $w \leftarrow 0$, $j \leftarrow 1$
while $j < n^{\text{dependency}}[i]$ **do**
    $b \leftarrow$ True
    for $\ell \leftarrow 1$ **to** $n^{\text{set}}[i,j]$ **do**
        if D$[i,j,\ell]$ is None **then**
            $b \leftarrow$ False
            break
        else
            $K[i] = K[i] + W[i,j,\ell] \cdot D[i,j,\ell]$
            $w = w + W[i,j,\ell]$
        end if
    end for
    if b **then**
        break
    end if
    $j \leftarrow j + 1$
end while

if $j = n^{\text{set}}[i,j]$ **then**
    return "CANNOT LOCATE MISSING KEY POINT!!"
else
    $K[i] \leftarrow K[i]/\omega$
end if
return $K[i]$

fore, YOLOv8n-Pose was extended to locate the missing key points, in order to make the pose estimation module robust to occlusions. Figure 4.6 showcases the output of the extended model whenever some body key points are occluded.

Algorithms 2 and 3 have been developed to estimate the position (i.e., the x and y coordinates) of the key points that are missing from an image: Algorithms 2 (lines 7–11) loops through the entire list K of the positions of the key points, that has initially been deduced from an image. This loop assesses whether or not any positions are missing within list K. For those key points that have not been detected by YOLOv8 in the image, a function `estimate_keypoint`$(\cdot,\cdot)$ has been developed (the details are given in Algorithm 3) in order to determine the position of that missing key point and to fill in the empty element in list K. Note that since Algorithm 3 runs within Algorithm 2, we have not repeated, in Algorithm 3, the same definitions that are used in both algorithms.

Function `estimate_keypoint`$(\cdot,\cdot)$ (see Algorithm 3) receives K and the index i of the element that is missing from K, and determines the position of the missing key points. The algorithm needs the position of the other key points that the position of a missing key point directly depends on. Those key points are called the "dependencies" of the key point. For instance, using the body proportions, the position of the missing key point, the right elbow, can be estimated based on the positions of the dependency key points, the right hand and right shoulder. Alternatively, using the body symmetry the position of the right elbow may be estimated using the positions of the two shoulders and the left elbow. Thus, for each key point, more than one list of dependencies may exist (e.g., in the given example both sets {right hand; right shoulder} and {right shoulder; left shoulder; left elbow} belong to the set of dependencies of Key point "right elbow").

Thus, the dependencies for all the key points in list L (which includes all the n^{keypoint} key point labels) are defined a-priori in n^{keypoint} lists called "R[r]", which, for $r \in \{1,\ldots,n^{\text{keypoint}}\}$, is composed of $n^{\text{dependency}}[r]$ sets of dependencies for key point r in list L. Moreover, a list "W[r]" with a similar number of sets and elements within each set as in R[r] is pre-defined and includes the weights that the algorithm associates with each corresponding dependency key point within R[r] (see lines 1–3 in Algorithm 3). For the given example, suppose that the order of the right elbow in list L is 8. Then, as an example, R[8] = { {right hand; right shoulder}, {right shoulder; left shoulder; left elbow} } and W[8] = { {0.5; 0.5}, {0.2; 0.4; 0.4} }. These weights are used to compute the position of a missing key point, based on a weighted average of the positions of the dependencies of that missing key points.

Note that the procedure of determining the positions of the missing points should better be run in more than only one iteration. This is because the positions that have been determined per iteration for the missing key points may be the positions of the dependency key points of other missing key points that could not be positioned in that iteration. Thus, the next iterations will help to position those key points as well. This is why in line 6 of Algorithm 2, a loop of $n^{\text{iteration}}$ iterations has been designed. In the real-life experiments in the CyberZoo, the number of the iterations, $n^{\text{iteration}}$, was set to 10, which was deemed sufficient to determine all the missing key points.

Note that when a list or parameter is always fixed, we have used a regular font for the corresponding notations, whereas for varying lists and for other variables we have used an italic font.

Remark 6. *For unconventional body poses (e.g., when one arm is stretched and the other arm is bent), for symmetrical parts of the body (e.g., for the left and right arms), the algorithm considers the most reliable estimate of the position of one of the two parts (e.g., the estimate for the right arm) from the image. For the other part (in this case the left arm), a virtual symmetrical body part is considered and the coordinates are estimated accordingly (similarly to the right-hand side plot in Figure 4.6). Thus, whether or not a human has a conventional pose does not impact the performance of the algorithms.*

4.2.2. VICTIM LOCALIZATION

In this part of the methodology, determining the position of the victims from the ground and aerial images is explained. Localizing the victims autonomously via robots requires homography and distance estimation, as explained next.

Homography Estimation An indispensable method for mapping image pixels into real-world coordinates, especially when dealing with planar surfaces, is homography estimation. In our research, homography estimation serves as a valuable auxiliary method that facilitates the transformation of 2D image points into their corresponding real-world coordinates relative to the robot platforms. The homography transformation of 2D points is performed using a 3×3 homography matrix $\mathbf{H}$, given by:

$$\mathbf{H} = \begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} & \tilde{h}_{13} \\ \tilde{h}_{21} & \tilde{h}_{22} & \tilde{h}_{23} \\ \tilde{h}_{31} & \tilde{h}_{32} & 1 \end{bmatrix} \tag{4.1}$$

which implies that 8 parameters (3 rotational, 3 translational, 2 scalars) should be identified, requiring a system of 8 equations. These equations are generated by choosing 4 reference points, for which the corresponding x and y coordinates in both the camera view and the augmented view are known. Consider a set $P = \{(x_i, y_i)$ for $i = 1, \ldots, n^{\mathbf{H}}\}$ of $n^{\mathbf{H}}$ points co-planar in the camera view, and the corresponding set in the augmented view, i.e., $P'' = \{(x_i'', y_i'')$ for $i = 1, \ldots, n^{\mathbf{H}}\}$. The points within set P may be selected arbitrarily. However, considering points that are more distant from one another will help with the precision of the approach that will be explained next. This is because by selecting points that are farther from one another, a larger portion of the image frame is considered in the calibration. Furthermore, such a selection helps the algorithm to be more robust to errors in the estimation of one or a few of the reference points. Thus, the most efficient approach is to select the four corners of the image frame. For translation of the positions of these points into their corresponding real-world positions, a reference object may be placed at a known position with respect to the global reference frame, such that the object appears in the corner of the image. This procedure can be done in a laboratory before deployment in a real SaR setting because the calibration refers to the robot and to the orientation of its camera, and not to any external factors. Therefore, changing the x and y coordinates of the robot position or the yaw angle of the robot will not affect the calibration. Only a change in the tilt angle would require a different calibration of the system.

The following direct linear transform is used to obtain 2 linear equations per point:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i'' & -y_i x_i'' & -x_i'' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i'' & -y_i y_i'' & -y_i'' \end{bmatrix} \cdot$$
$$\begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} & \dots & \tilde{h}_{32} & 1 \end{bmatrix}^\top = \mathbf{0} \tag{4.2}$$

Alternatively, with more than 4 reference points the estimation of the homography matrix $\mathbf{H}$ transforms into an optimization problem that should determine an optimal homography matrix $\mathbf{H}^*$ that minimizes a defined cost function, which typically includes the algebraic sum, across all reference points, of the geometric distances between the projected points $(\tilde{x}_i'', \tilde{y}_i'')$ obtained using the candidate $\mathbf{H}$, and the actual corresponding points (x_i'', y_i'') in the map view.

Given $\mathbf{H}$, a point (x_i, y_i) in the camera view is transformed into (x_i'', y_i'') in the augmented view, using the following equation from [279]:

$$\begin{bmatrix} x_i''/\lambda & y_i''/\lambda & \lambda \end{bmatrix}^\top = \mathbf{H} \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^\top \tag{4.3}$$

with λ a scaling factor that ensures the transformed points maintain their relative positions after the transformation.

Distance Estimation Knowing the distance of a detected victim from the ground robot is crucial in SaR. Under a simplified pinhole camera model, the distance d of a detected victim from the camera of the ground robot, is given via:

$$d = \frac{s^{\text{real}} l^{\text{focal}}}{s^{\text{image}}} \tag{4.4}$$

with s^{real} the real-world size of the victim (where knowing the position of the victim allows the robot to have a more realistic estimation of s^{real} for the victim), l^{focal} the focal length of the camera, and s^{image} the size of the victim in the image. It is assumed that the camera is calibrated. Note that s^{real} is a value chosen before the SaR mission and kept constant based on average values typically observed in human anatomy. This reference value can be taken as full height, shoulder-to-shoulder distance, and shoulder-to-elbow distance (see next Table 4.2 for a practical example), depending on the body parts that are visible in the image during the SaR mission. In addition, l^{focal} is usually provided by the manufacturer, or is alternatively computed from the height H^{image} and width W^{image} of the image in pixels, and the horizontal FOV_h and vertical FOV_v fields of view of the camera (see Figure 4.7). We have:

$$l^{\text{focal}} = \frac{H^{\text{image}}}{2\tan\left(\frac{\text{FOV}_\text{v}}{2}\right)} = \frac{W^{\text{image}}}{2\tan\left(\frac{\text{FOV}_\text{h}}{2}\right)} \tag{4.5}$$

Two sources of position measurement are obtained from the ground and the flying robot (see Figure 4.8). Each measurement is computed by vector summation of the position of the robot and the position of the victim relative to that robot. The position of the robots is generally measured via a position-determination system, e.g., via GPS. The position of the victim relative to each robot can be computed using (4.3) and (4.4).

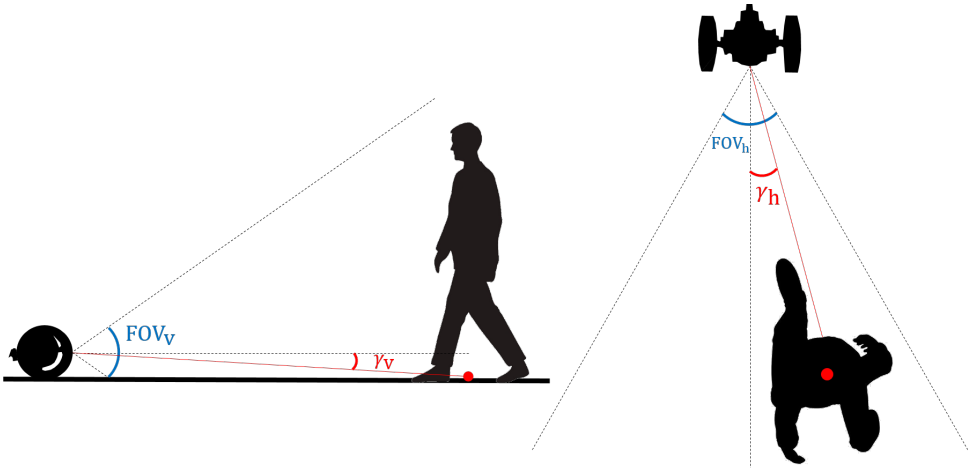


Figure 4.7: The vertical (left) and the horizontal (right) FOVs of a camera, where γ represents the angle between the camera and the victim, with a subscript 'v' and 'h' for, respectively, the vertical and the horizontal cases.

4.2.3. VICTIM TRACKING

In this part of the methodology, we explain in detail how the trajectory of a moving victim is tracked through the images that are taken by the ground and aerial robots.

State estimation is a fundamental requirement in several SaR contexts since it enables one to make inferences about the states of a dynamic system based on noisy sensor measurements. Our primary objective of state estimation is to infer the trajectory of a moving victim after being detected. A Kalman filter, using a model of the victim motion, is employed to fuse the measurements of the two robots. Such models of the victim motion in disaster situations are available in the literature (see, e.g., [146]). By incorporating knowledge of how the victim's trajectory is expected to evolve over time, the Kalman filter makes predictions about the future states of the victim and generates a continuous and coherent trajectory according to the model. The state estimation approach is explained next.

State Transition Model The state transition model describes how the states of a system evolve over time. In the discrete time, the evolution of the state vector, $\boldsymbol{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top$, including the position and the velocity of the victim in a 2D space at time step k, is via the following state-space equation, which is a general form of a dynamic model for the victim motion that has been simplified via linearization and excluding the influence of external forces:

$$\boldsymbol{x}_{k+1} = \mathbf{F}\boldsymbol{x}_k + \boldsymbol{w}_k \tag{4.6}$$

with $\mathbf{F}$ the state transition matrix and $\boldsymbol{w}_k$ a random variable representing the process noise, assumed to be drawn from a zero mean normal distribution with covariance matrix $\mathbf{Q}$. In real life, the actual state $\boldsymbol{x}_k$ may be unavailable, and thus estimated sequentially via (4.6). A hat symbol is used to represent the estimated states.

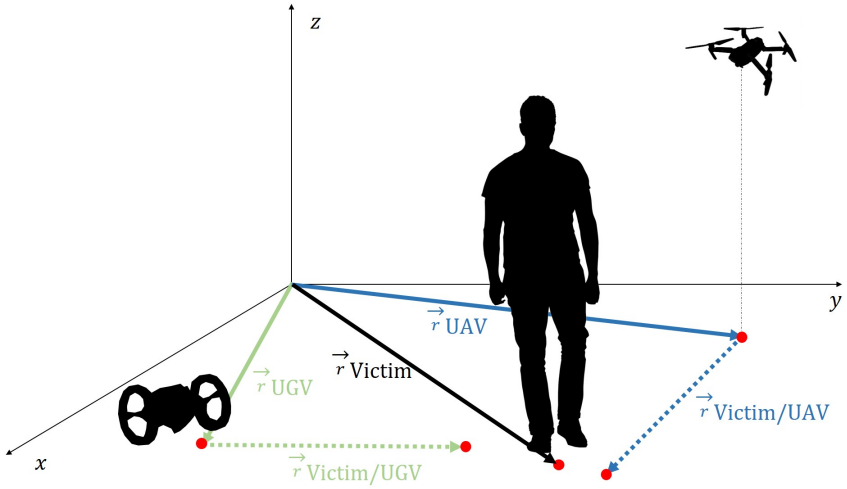


Figure 4.8: Victim location per time step: Vector sum of the (flying or ground) robot position, and the relative position of the victim and the robot.

Observation Model The observation model relates the noisy or uncertain measurements $\boldsymbol{x}_k^{\mathrm{m}} = [x_k^{\mathrm{m}}, y_k^{\mathrm{m}}, \dot{x}_k^{\mathrm{m}}, \dot{y}_k^{\mathrm{m}}]^\top$ obtained from the sensors of the robots to the actual states of the system per time step via:

$$\boldsymbol{x}_k^{\mathrm{m}} = \mathcal{O}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{4.7}$$

with $\mathcal{O}$ the observation matrix and $\boldsymbol{v}_k$ a random variable representing the measurement noise, assumed to be drawn from a zero mean normal distribution with covariance matrix $\mathbf{R}$. The Kalman filter operates in two main steps, prediction and update, explained next. The superscript $^-$ for a variable indicates the variable before being updated.

Prediction Step At time step k, the most recent estimated state $\hat{\boldsymbol{x}}_k^-$ is updated (details are given next) to $\hat{\boldsymbol{x}}_k$ and is then used by the Kalman filter to estimate the future state of the system, i.e.:

$$\hat{\boldsymbol{x}}_{k+1}^- = \mathbf{F}\hat{\boldsymbol{x}}_k \tag{4.8}$$

The error covariance matrix $\mathbf{P}_{k+1}^-$ (which indicates how uncertain the state prediction is) is estimated by propagating the current updated error covariance matrix, $\mathbf{P}_k$, through the state transition model and by adding the process noise covariance matrix, $\mathbf{Q}$, i.e.:

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k\mathbf{F}^\top + \mathbf{Q} \tag{4.9}$$

Update Step At time step $k+1$, when new measurements are obtained, the state predicted at time step k for time step $k+1$ is updated to improve the certainty. First, the Kalman gain $\mathbf{K}_{k+1}$ is obtained, based on the error covariance prediction matrix, $\mathbf{P}_{k+1}^-$, the observation matrix, $\mathcal{O}$, and the measurement noise covariance matrix, $\mathbf{R}$, via:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^-\mathcal{O}^\top(\mathcal{O}\mathbf{P}_{k+1}^-\mathcal{O}^\top + \mathbf{R})^{-1} \tag{4.10}$$

The Kalman gain reflects the relative importance of the prediction and the measurements.

Next, the state $\hat{\boldsymbol{x}}_{k+1}^{-}$ predicted at time step k via (4.8) is updated using the Kalman gain and the difference of the actual measurements obtained from the sensors at time step $k+1$ and the predicted value for the measurements using $\hat{\boldsymbol{x}}_{k+1}^{-}$ and (4.7), i.e.:

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^{-} + \mathbf{K}_{k+1}(\boldsymbol{x}_{k+1}^{\mathrm{m}} - \mathcal{O}\hat{\boldsymbol{x}}_{k+1}^{-}) \tag{4.11}$$

Similarly, the error covariance matrix is updated at time step $k+1$ to incorporate the reduced uncertainty. We have:

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathcal{O})\mathbf{P}_{k+1}^{-} \tag{4.12}$$

Smoothening Filter In order to refine and smoothen the output of the Kalman filter, an exponential moving average filter is employed that assigns exponentially decreasing weights to older estimations. This allows the filter to adapt more quickly to recent changes in the measurements, while incorporating past information. The smoothening for time step k is performed via:

$$\hat{\boldsymbol{x}}_{k}^{\mathrm{EMA}} = \alpha\hat{\boldsymbol{x}}_{k} + (1-\alpha)\hat{\boldsymbol{x}}_{k-1}^{\mathrm{EMA}} \tag{4.13}$$

where $\hat{\boldsymbol{x}}_{k}^{\mathrm{EMA}}$ is the smoothened state estimate and α is the smoothening factor of the exponential moving average filter. Larger values for α result in higher weights for recent changes.

4.2.4. TERRAIN ELEVATION MAP

In the last part of the methodology, we explain how the ground and aerial robots will collaboratively determine the ground elevation of the terrain.

First, the flying robot carries and drops a standard object with a known shape and size on a point where the terrain elevation is to be estimated. Then, the distance of the ground robot from this object is sent via the flying robot (by leveraging homography estimation and trigonometry) to the ground robot. Alternatively, the ground robot estimates its relative distance to the object using (4.4). This distance, together with the coordinates of the ground robot, is used by the robot to determine the expected position (blue point in Figure 4.9) of the object, assuming a flat ground. Then, from the image the camera of the ground robot captures, the real point (shown in red in Figure 4.9) that the object touches the ground. The pixel distance between these two points is used for the estimation of the ground elevation.

When the object sits close to the ground robot or at a high elevation, the object is perceived as smaller, and thus, the distance is perceived as larger than it is (see Figure 4.10). This impacts the estimated terrain elevation. To mitigate this, depending on the geometry of the object, it is possible to take its width as a reference, since the width is only distorted in extreme scenarios.

Remark 7. *The approach introduced in this section for terrain elevation mapping is mainly meant to be performed randomly, on a large scale, within the first stages of mapping the*

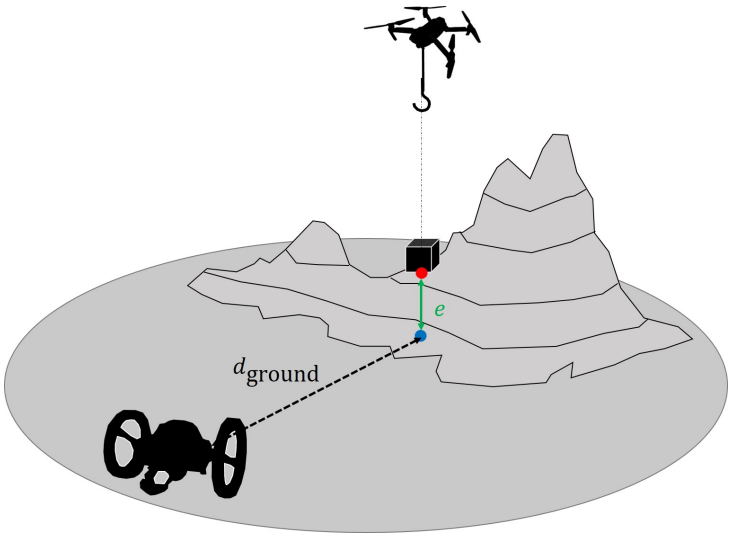


Figure 4.9: In determining the terrain elevation, the flying robot drops a standard object with a known size in the desired place. The ground robot then receives via the flying robot its distance to the dropped object, or else estimates this distance using (4.4). This information is used by the ground robot to determine the point (shown in blue) that the object would touch the ground if the ground were flat, using (4.4). Then via its camera, the ground robot captures sight of the point (shown in red) that the object actually touches the ground. The pixel distance between these two points is used to compute the elevation of the ground (shown in green).

environment. In other words, a group of drones will be deployed to drop standard objects, not necessarily tennis balls, but, for instance, more flexible objects with possibly an adhesive texture that are more suitable for simplifying their placement on a non-smooth ground. After a large number of such objects have been distributed around the environment, the ground robots will be deployed to detect these objects and the corresponding elevation of the ground underneath them, as a part of mapping the unknown environment. Therefore, no precision in locating these objects via the flying robots in the environment is in general required. In case such precision is needed for some reasons, as it was mentioned, an adhesive material may be used to place the object more precisely. In such a case, the movements and altitude of the flying robots should also more precisely be controlled.

4.3. CASE STUDIES

This section explains the setup, implementation, and results of real-life experiments that were conducted at the CyberZoo of the Delft University of Technology, in order to validate our proposed approaches. In the experiments, challenging scenes that encompass occlusions and the temporary unavailability of the sensors have also been generated.

4.3.1. SETUP OF THE EXPERIMENTS

The experiments were conducted at the CyberZoo of the Delft University of Technology (see Figure 4.11), a research and test laboratory that embeds a 10×10 m^2 synthetic

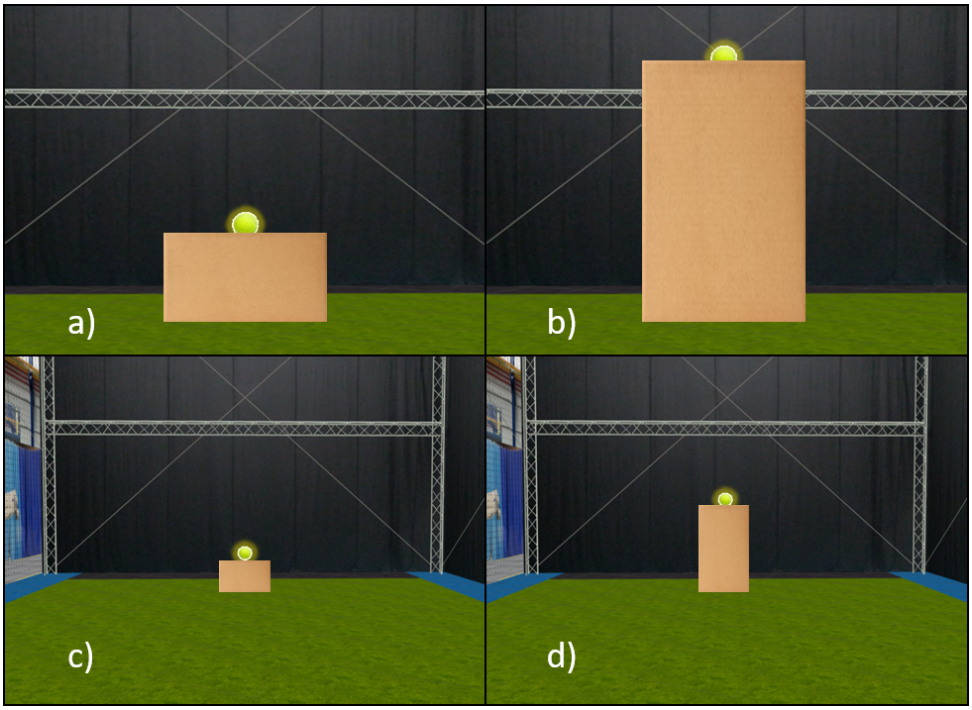


Figure 4.10: When the object sits close to the ground robot, as in a) and c), or is at a high elevation, as in b) and d), partial occlusion is more significant. In the image a) $d_{\text{ground}} = 1$ m, $e = 25$ cm, b) $d_{\text{ground}} = 1$ m, $e = 75$ cm, c) $d_{\text{ground}} = 3$ m, $e = 25$ cm, d) $d_{\text{ground}} = 3$ m, $e = 75$ cm.

turf surrounded by safety nets, for protecting the participants and robots during the experiments. Furthermore, the experimental facilities at the CyberZoo are equipped with twelve high-tech cameras and Motive, a software platform designed to control motion capture systems for various tracking applications. By placing markers asymmetrically in a rigid body, it is possible to track them using Motive to obtain their position, velocity, and 3D orientation (pitch, roll, and yaw).

The drone used in the experiments was a Parrot Bebop 2 (see the right-hand side photo in Figure 4.12). Parrot Bebop 2 is a small quadcopter that measures 382 mm in front, 328 mm on either side and 89 mm in height. It weighs 500 g and has a 2700 mAh battery. Depending on the circumstances, the drone can fly continuously for up to 25 minutes on this battery power. The 14 MP front camera on Parrot Bebop 2 can record 1080p video at 30 fps. The drone is also equipped with a bottom camera used to estimate the velocity of the drone. This built-in camera is not suited for video recording. Thus, a 14 MP bottom camera, which can also record 1080p video at 30 fps was attached to the bottom of the drone. The drone has its own WiFi network, allowing the drone to connect to other devices. It boasts a dual-core processor operating at 500 MHz per core, orchestrating seamless flight control and navigation. Complemented by a quad-core GPU, the drone efficiently processes video and image data and ensures smooth and

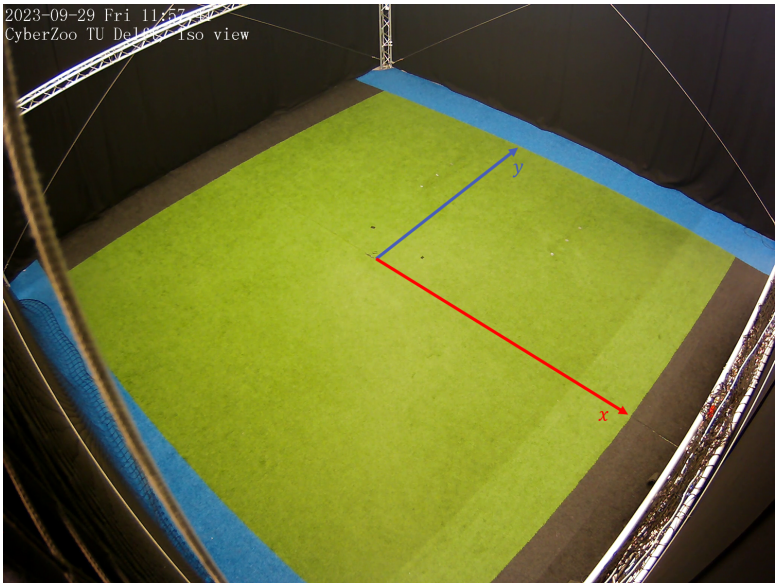


Figure 4.11: CyberZoo including the reference frame, where the origin corresponds to the center of the laboratory field.

Figure 4.12: Parrot Jumping Sumo (left) and Parrot Bebop 2 (right).

high-quality visuals.

As for the ground robot, the Parrot Jumping Sumo (see the left-hand side photo in Figure 4.12), which is a compact, wheeled ground robot, was used. Displaying a movable base with two independently driven wheels, Parrot Jumping Sumo measures 185 mm in length, 150 mm in width, and 110 mm in height. Weighs 180 g, and features a 550 mAh battery that grants an operational time of up to 20 minutes. The robot can perform horizontal and vertical jumps up to 80 cm, and incorporates a wide-angle camera providing 480 p at 15 fps. Just like Parrot Bebop 2, Parrot Jumping Sumo is equipped with its own WiFi network, allowing it to connect to other devices. Parrot Jumping Sumo is powered by an ARM Cortex A9 processor running at a clock speed of around 1 GHz, facilitating swift and responsive execution of commands and sensor data processing, enabling agile movements and interactions.

Measurement	Reference [m]	μ [m]	σ [m]
Height	1.7000	1.7538	0.0732
Shoulder-to-shoulder	0.3800	0.4075	0.0398
Shoulder-to-elbow	0.3000	0.3054	0.0218

Table 4.2: Measurements and reference values for the participants

Overall, the autonomy of the robots, the sufficient quality of their cameras, and their affordability, ready-to-use nature, and user-friendly interfaces make them suitable for this research. The processing of the images and videos was performed via an external computer that was equipped with an Intel Core i7-1185G7 processor, part of the 11th generation, operating at a base frequency of 3.0 GHz and reaching up to 4.8 GHz with Turbo Boost capability. It boasted 16 GB of LPDDR4x RAM and integrated Intel Iris Xe Graphics. The storage was facilitated by a 512 GB PCIe NVMe SSD. The operating system utilized was Ubuntu 20.04.6 LTS. The source codes were composed using Python, owing to its compatibility with external code, versatility, and robust libraries.

4.3.2. EXPERIMENTS FOR DISTANCE ESTIMATION VIA GROUND IMAGES

A collection of images of various body poses from volunteer participants were collected. In order to represent a general population, the participants were selected to have diverse physical characteristics, e.g., different heights, and shoulder-to-shoulder and shoulder-to-elbow distances. The experiments included 24 participants of 12 different nationalities. The reference values used for the height, shoulder-to-shoulder distance, and shoulder-to-elbow distance were selected based on [299] and are given in Table 4.2.

The corresponding measurements were also gathered from the participants, in order to gain awareness of how closely they matched the reference values. Larger deviations from the reference values in the participant sample are expected to yield larger errors. The average value, μ, and standard deviation, σ, for these measurements are also given in Table 4.2. For height, shoulder-to-shoulder distance, and shoulder-to-elbow distance, the average values are, respectively, 3.16%, 7.24%, and 1.80% larger than their reference values, whereas the corresponding standard deviations are, respectively, 4.17%, 9.77%, and 7.14% of their average values. Thus, the measurements from the participants matched the reference values relatively closely, with the shoulder-to-shoulder distance the least reliable measurement in terms of both the average value and the standard deviation. While the average value for the shoulder-to-elbow distance was closer to the reference value than that for the height, the worst-case measurement was still less reliable than that of the height due to its increased standard deviation. On grounds of the challenge of determining these distances precisely in a conventional SaR scenario, the algorithm privileges using the height, shoulder-to-shoulder distance, and shoulder-to-elbow distances owing to their decreasing absolute values.

Each participant was asked to take 7 different poses for the camera (see Figure 4.13): standing facing the camera, standing in profile, standing back to the camera, sitting facing the camera, sitting in profile, lying on the side facing the camera, and lying face-up. These poses have been repeated for distances of 1.5 m and 3 m from the ground robot.

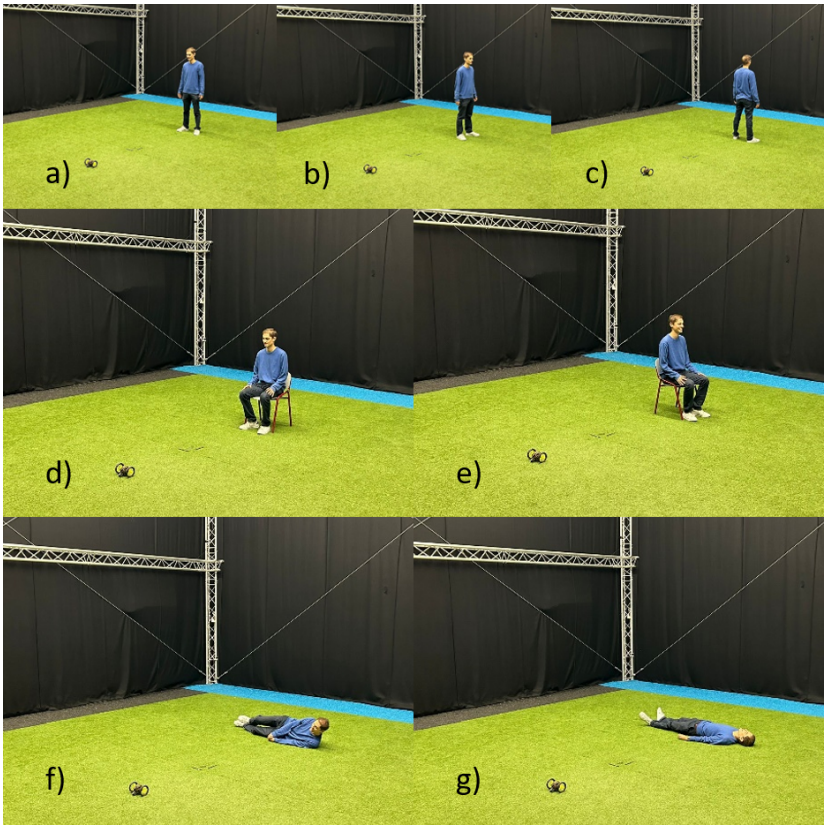


Figure 4.13: The participants were asked to take specific poses for the ground vehicle camera. The images were captured systematically from varying distances of 1.5 m and 3 m. The sequence of the poses encompasses a) standing facing the camera, b) standing in profile, c) standing back to the camera, d) sitting facing the camera, e) sitting in profile, f) lying on the side facing the camera, and g) lying face-up.

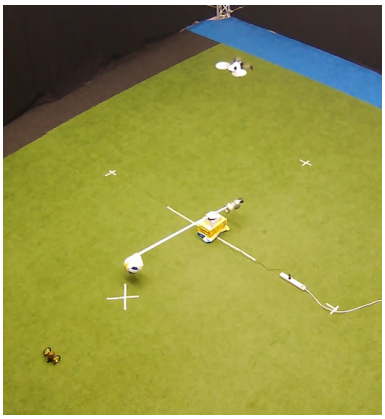


Figure 4.14: The motorized rotating stand makes the volleyball move in a circle with constant angular velocity $\omega = 6$ rpm. During the experiments, the flying and ground robots captured video streams of the movement of the volleyball in an unobstructed environment (left) and in a cluttered environment (right). The radius of the trajectory of the volleyball was adjustable.

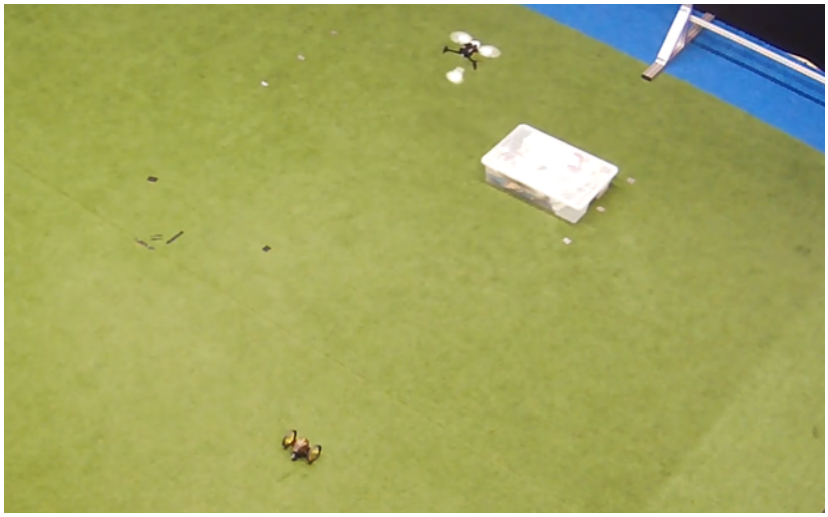


Figure 4.15: The flying robot carried and dropped a tennis ball where the elevation of the terrain was to be estimated. The ground robot positioned itself in an adequate perspective, in order to estimate the elevation.

The experiments thus generated 14 images per participant and 336 images in total. All images were post-processed using the extended YOLOv8n-Pose, to predict the bounding boxes and the location of the key points. The key points are augmented by leveraging typical body proportions and body symmetry as described in Section 4.2.1. This ensured that all key points were estimated even when some body parts were not detected or captured by the camera. The distance between the key points were then inserted into (4.4) to determine the distance of the participant from the ground robot in pixels, and to compare this with a reference real-world value for that distance. As different poses distort different key point distances, the camera frame distance substituted in the equation depended on the pose. When the participant was standing or lying on the floor, the distance between the feet and the head was compared against a reference value for the human height. When the participant was sitting facing the camera or in profile, the shoulder-to-shoulder distance or the shoulder-to-elbow distance was compared to the reference values, respectively.

4.3.3. EXPERIMENTS FOR TRACKING A MOVING OBJECT

The main aim of this set of experiments was to track the movement of a real-world object by leveraging the video streams of the ground and flying robots. Due to the limited space at the experimental facility and the need to compare the obtained movement trajectory with a ground truth, a precise trajectory had to be replicated. Therefore, a volleyball was taken as the moving object to be tracked. The volleyball was attached to a rod, tied to a 360-degree motorized rotating stand. By moving the rod, it was possible to manipulate the radius of the trajectory. The motorized rotating stand allowed for adjustable angles of rotation, directions of rotation, and angular velocities. During the experiments, different obstacles were placed around the trajectory of the volleyball to occlude both the aerial and the ground views of the motion (see Figure 4.14).

In all the experiments, the center of rotation was placed at the center of the CyberZoo, i.e., at $(x, y) = (0, 0)$. The ground robot was kept static at $(0, -3)$, and the flying robot hovered with an altitude of 3 m, as closely as possible above the center of the CyberZoo. It was crucial to keep the altitude of the flying robot approximately constant, so that the homography matrix $\mathbf{H}$ does not regularly need to be calibrated. Correcting a frame manually took 10 seconds on average. The videos that were being processed had a length of around 30 seconds. The frame rate was 15 fps for the ground robot and 30 fps for the aerial robot. This editing step was done before feeding the videos to the algorithm.

Remark 8. *In order to reduce the computational errors in the estimation of the trajectory of the victim that are sourced from the relative motion of the robots or from the tilting motion of the aerial robot, a GPS may be used on the robots. With a GPS, the position of the robots with respect to a fixed reference point (e.g., the center of rotation of the volleyball in the case studies) is obtained precisely and the relative distance of the robot and the moving target is corrected accordingly. This, however, may not be feasible in complex SaR applications. Moreover, using GPS on various robots increases the costs of SaR robotics (which contradicts the main goal of this chapter, i.e., to propose a framework that also suits low-income countries). Therefore, an alternative option, that was also used in our case studies, is to apply a video editing software, e.g., the daVinci Resolve, which allows*

editing the frames and to ensure that the center of rotation of the volleyball is placed in the center of each frame.

Throughout the experiments, the angular velocity and the direction of the rotation were kept constant at 6 rpm counterclockwise. Five main settings were tested: r = 0.75 m, r = 1.00 m, and r = 1.25 m in an unobstructed environment, r = 1.25 m in a cluttered environment for the flying robot only, and r = 1.25 m in a totally cluttered environment.

For each experiment, both video streams were post-processed frame by frame and off-board on an external laptop, using YOLOv8n. For the ground robot, (4.3) and (4.4) were employed to determine the position of the volleyball in the reference frame depicted in Figure 4.11. For the flying robot, (4.3) was sufficient to determine the position of the volleyball. Since the ground and flying robots recorded videos at, respectively, 15 fps and 30 fps, the frames were aligned before inputting the measured locations to the Kalman filter. The Kalman filter was updated at 30 fps, i.e., without any measurements from the ground robot in half of the updates, due to the halved frame capture rate. The estimated trajectory was compared with the ground truth per setting.

4.3.4. EXPERIMENTS FOR ESTIMATION OF THE GROUND ELEVATION

In these experiments, the flying robot carried and dropped a signaling object where the elevation of the terrain was to be determined. A remote air-dropping system was incorporated into the setup and a thrower was attached to the bottom of the flying robot. Whenever the object needed to be released, a remote control was manually operated. Even though the maximum payload of the airdrop system is 750 g, weights above 150 g seriously jeopardize flight stability. Therefore, a tennis ball weighing 58 g was selected as the signaling object and boxes with varying heights were used to model the terrain elevation (see Figure 4.15). The ground robot was positioned at different distances from the signaling object and 12 images were captured via the camera of the ground robot for distances $1, 2, 3, 4$ m and for elevations $25, 50, 75$ cm. These images were then processed off-board, considering 2 options: First, the ground robot estimated the terrain elevation on its own, as explained in Section 4.2.4. Second, the ground vehicle received its distance from the tennis ball via the flying robot.

4.3.5. RESULTS AND DISCUSSIONS

Next, we present and discuss the results of our experiments.

RESULTS FOR DISTANCE ESTIMATION VIA GROUND IMAGES

After batch-processing the images, the average, maximum, and minimum distances, along with their upper and lower dispersion, were obtained. Only 5 of 336 images (i.e., less than 1.5%), did not result in detecting a human above a confidence threshold of 20%. For the rest, the results were divided based on the body pose and the distance from the camera (see Figure 4.16).

From the plots, the distance to the camera appeared to have the largest impact when the participant was standing. When the person was closer to the camera, the results were promising given the proximity to the ground truth, the small upper and lower dispersion, and the non-substantial errors in the extremum. For larger distances, however,

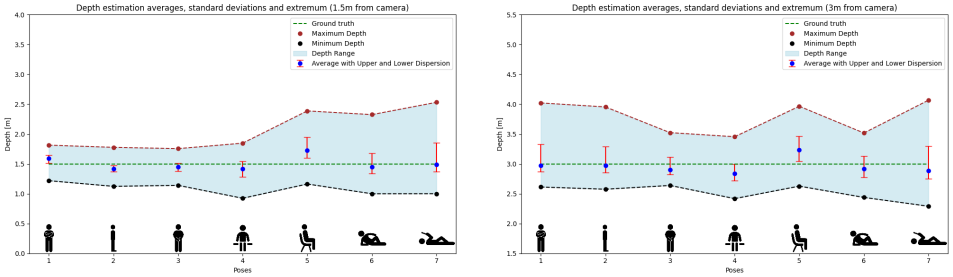


Figure 4.16: Average values, standard deviations, and extremum for estimation of the distance for 1.5 m (left) and 3 m (right) ground truth distances.

4

these metrics degraded. When the participant was sitting or lying on the floor, the pattern seemed to be consistent, regardless of the distance of the participant to the ground camera. A possible explanation is the lens distortion (i.e., the optical anomalies in camera lenses that result in deviations from ideal light projection), since this effect is not considered in modeling a simplified pinhole camera.

The worst cases appeared when the distance was estimated in excess, rather than in defect. This is compatible with the fact that the participants were, on average, taller than the reference values, which increases the estimated distance. Therefore, the reference values should be adapted depending on the average height of the participating people.

Remark 9. *When the height of the victims is considered as the reference value for determining their distance from the ground robot, errors may occur especially whenever the height of a victim varies significantly from the values that have been used in the calibration. In fact, the distance of a person with a significantly larger height may by mistake be estimated smaller than it really is and vice versa. In order to mitigate such errors, instead of the height, the relative distance between other key points of the body that is not subject to significant variations among different people may be considered, e.g., the relative distance of the eyes or other key points in the face.*

The standing poses appeared to be, on average, the least prone to errors. While their error pattern changed significantly with the distance from the camera, the average distance remained close to the ground truth. Sitting and lying poses appeared to be more challenging. In fact, whenever the participants have a wider range of motion and poses where a smaller measurement is taken to compare against the reference (e.g., shoulder-to-shoulder distance as opposed to the height) are more prone to uncertainties. For instance, the participants stood still in a similar fashion while standing. However, while sitting, they placed their legs more openly or closely, and curved their back to different extents. The same goes for lying positions, where the participants tilted their bodies and stretched their arms and legs to different degrees.

The average relative errors of the estimated distances per pose across all participants and for an arbitrarily chosen participant are given in Figure 4.17. As expected, the curves kept the same pattern except for the standing poses. Moreover, the sitting and lying

poses generally displayed larger relative errors, than the standing poses. The slight discrepancy in the pattern for the standing poses is because the faces of some participants standing 1.5 m from the camera were not captured in the image. Thus, the position of their faces was estimated by the algorithm, giving rise to larger errors. Nevertheless, sitting in profile and lying face-up still appeared to be the most challenging cases.

The average across all poses for the relative errors is 13.73% and 9.67% for ground truth distances of, respectively, 1.5 m and 3 m. This 4.06% discrepancy was because the participants did not always place themselves in the exact expected distance from the ground robot. These slight displacements of a few centimeters have a larger impact on the average relative errors when the distance is smaller. Assuming that this is the only external source of the error, the misplacement amounts on average to 12.18 cm. This implies that the actual relative error resulting from the algorithm itself ranges from around 4% to 10%.

Indeed, most of the relative errors lie between $< 1\%$ to 12%. Moreover, the relative errors appear to have no or negligible correlation with the distance from the camera. For the arbitrarily chosen participant, the relative error is suspected to arise from the physical discrepancies of the participant, as well as the slight deviations from the desired pose. The participant had a height of 1.83 m, shoulder-to-shoulder distance of 40 cm, and a shoulder-to-elbow distance of 31 cm, which are, respectively, 7.65%, 5.26%, and 3.33% above the reference values. Therefore, poses that rely on the height of the participant, i.e., poses $1, 2, 3, 6, 7$, are more prone to errors compared to other poses.

Finally, the average relative errors were shown to be distance-independent, which means that the absolute errors varied linearly with the distance from the camera. The relative errors varied from $< 1\%$ to $> 20\%$, depending on the pose and the physical characteristics of the participant. In 10 runs of the algorithm, the 336 images needed 83.38 s to 93.43 s to be processed, averaging to 0.248 s to 0.278 s per image.

RESULTS FOR TRACKING A MOVING OBJECT

The volleyball trajectories obtained after analyzing the videos were compared with the ground truth, based on various performance metrics, including the root mean square error (RMSE), extreme deviation, and area ratio. The RMSE shows how close the estimated volleyball trajectories are on average to the ground truth trajectories. Extreme deviation spots where these estimates and the ground truth stray the farthest. The area ratio reflects the percentage of the area enclosed by the ground truth that is also covered by the estimated trajectory, thus the match between the overall shapes. Considering the three metrics provides a more comprehensive insight about how accurate the estimations are. Using a volleyball instead of a human increased the difficulty of the tracking task for the flying robot, due to the smaller size and a less distinctive pattern of movement. Thus, in the presence of a cluttered environment with shadows, the flying robot had difficulty detecting the volleyball. The analysis was performed for a turning radius of $r = 0.75, 1.0, 1.25$ m with unobstructed views for both the flying and the ground robots, and for $r = 1.25$ m with obstacles placed for only for the flying robot.

The performance metrics for cases with only the ground robot, only the flying robot, and both robots are shown in Table 4.3, where the best performance per metric and scenario are shown with bold fonts. Moreover, Figures 4.18-4.21 illustrate the estimated

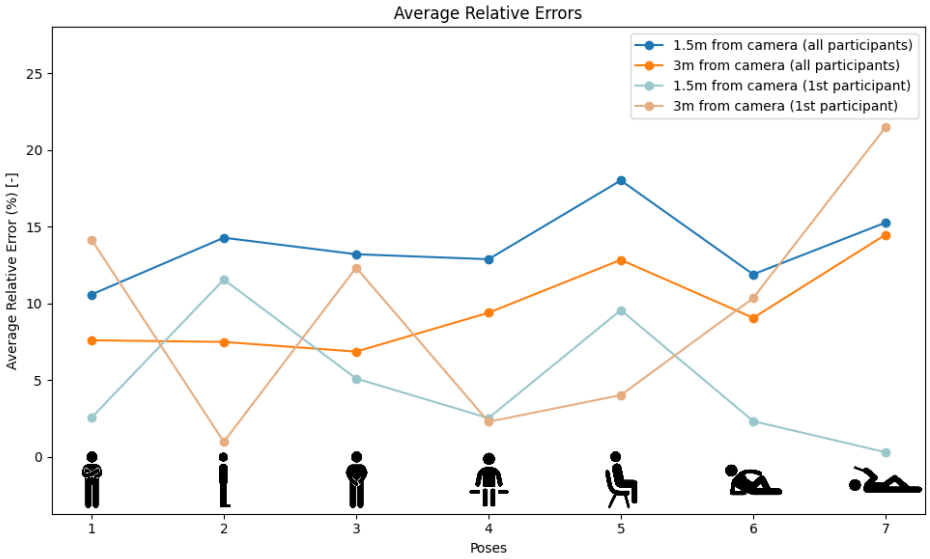


Figure 4.17: Average relative errors for a 1.5 m and 3 m distance to the ground robot across all the participants and for an arbitrarily chosen participant.

r (m)	RMSE/r (%) [−]			Extreme deviation [m]			Area ratio (%) [−]		
	G	F	G & F	G	F	G & F	G	F	G & F
0.75	**77.218**	89.492	88.090	1.5811	**1.5600**	**1.5600**	39.769	**97.895**	96.557
1.0	93.440	**79.492**	85.697	2.1608	**1.7575**	1.9561	64.832	86.568	**93.551**
1.25	95.028	**62.681**	79.992	3.4031	**1.7921**	2.3525	55.183	26.980	**65.447**
1.25-obstacles	**95.028**	100.946	98.035	3.4031	3.6800	**2.4621**	55.183	37.896	**78.164**

Table 4.3: Performance metrics for different setups and trajectories (G: Ground robot alone; F: Flying robot alone; G & F: Collaboration of both robots).

trajectories for $r = 0.75, 1, 1.25$ m and for $r = 1.25$ m with obstacles placed for the flying robot only. As expected, the metrics were degraded with increasing the radius of the trajectory in cluttered environments.

From Table 4.3, the RMSE for the 4 scenarios with the collaborating robot lies between the RMSE when the robots are used solely. However, not the same robot always performed the best. In fact, the standalone performance of these robots depended on their measurements, making it impractical to predict beforehand robots should be applied to an unknown scenario. Therefore, using a collaborative team of a ground and a flying robot is the most promising for handling an unknown scenario.

Regarding the extreme deviations from the ground truth, the flying robot produced the best results, whenever its view was unobstructed. This is because the camera of the flying robot has a larger fps, capturing points closer to each other, which prevents large errors by the Kalman filter. Nevertheless, when the view of the flying robot was

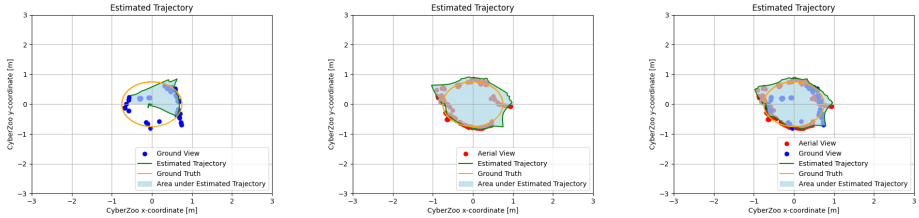


Figure 4.18: Estimated object trajectories for $r = 0.75$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right).

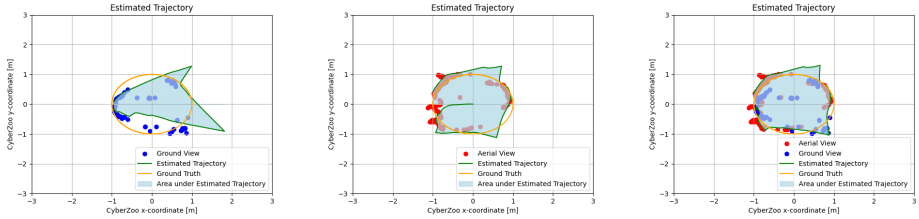


Figure 4.19: Estimated object trajectories for $r = 1.0$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right).

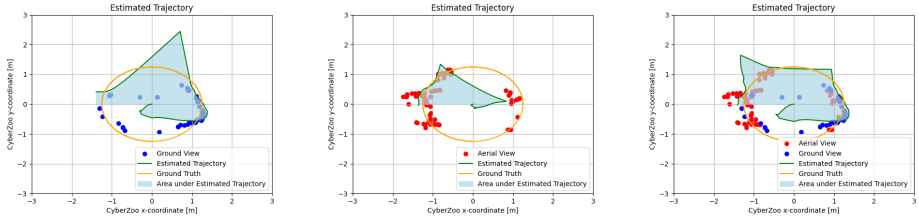


Figure 4.20: Estimated object trajectories for $r = 1.25$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right).

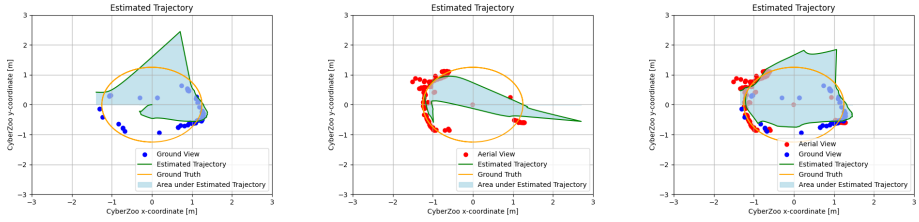


Figure 4.21: Estimated object trajectories for $r = 1.25$ m when obstacles were placed in the sight of the aerial robot, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right).

Figure 4.22: The algorithm draws the red point where the tennis ball touches the ground, the blue point where the tennis ball would touch the ground if the ground was flat, and the line connecting these points. The reduced quality of the image is because the ground robot captured it while moving.

obstructed, due to the missing measurements the estimated trajectory deviated significantly from its ground truth. Therefore, for an unknown SaR scenario, including the ground robot next to the flying robot can significantly improve the estimations for tracking a moving object.

For the area ratio, the team of both flying and ground robots stood out the most. The figures show that the estimated trajectories remained close circular shapes. For r = 1.25 m with an unobstructed view, the metric degraded since none of the robots detected the volleyball in the first instants. Thus, the volleyball was assumed to be at the center of the CyberZoo, impacting the final shape of the estimated trajectory of the volleyball. These findings are supported by the results presented in Section 4.3.

In 10 runs of a 10 s video for the ground view, it took between 19.41 s and 22.30 s to obtain the trajectories. Similarly, it took between 20.16 s and 22.68 s to process the aerial videos. For the team of robots, the procedure took between 36.68 s and 42.19 s. This is the time that the algorithm takes to process the edited videos, excluding the time that is needed to edit them. The editing has been done in advance, as addressed in Section 4.3.3.

RESULTS FOR ESTIMATION OF THE GROUND ELEVATION

Unless specified, the algorithm assumed by default that the flying robot did not provide assistance in the elevation estimation. An example of the output of the algorithm is shown in Figure 4.22. In this frame, the ground distance of the point, for which the elevation should be estimated, from the ground robot is 2 m and the elevation is 20 cm. The algorithm estimates a distance of 2.091 m and an elevation of 18.13 cm, indicating relative errors of, respectively, 4.55% and 9.35%.

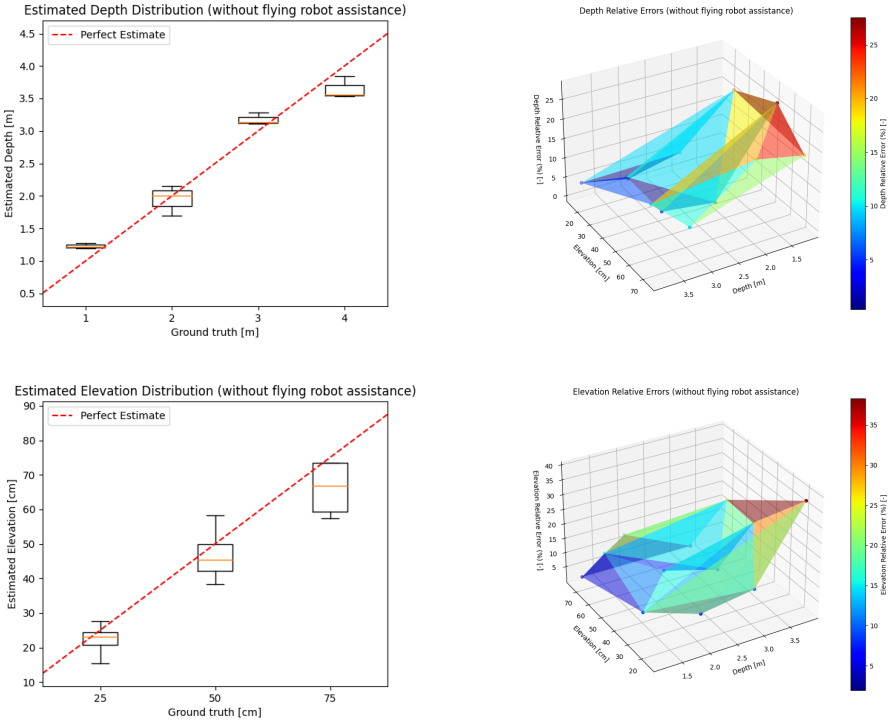


Figure 4.23: Distance distribution (top, left) and relative errors (top, right), elevation distribution (bottom, left) and relative errors (bottom, right), solely by the ground robot.

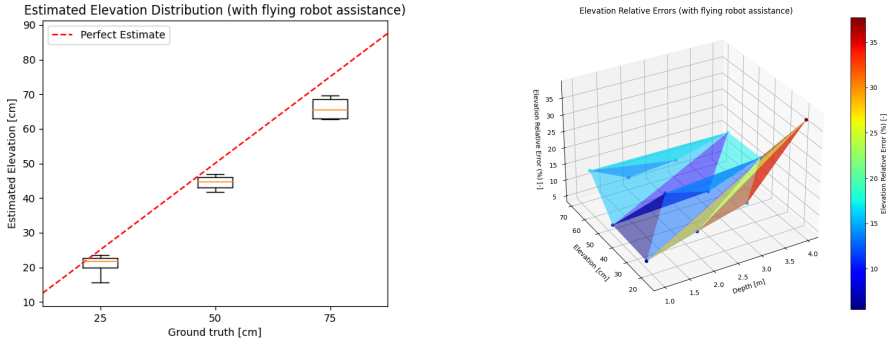


Figure 4.24: Elevation distribution (left) and relative errors (right), assuming that the flying robot provides a precise estimation of the distance for the ground robot.

Figure 4.23 illustrates the distribution of the measurements and relative errors for different distance and elevation values, when the ground robot had to estimate the elevation alone. From the top left plot, the distribution of the distance estimated by the ground robot follows the perfect estimate closely. For smaller (larger) distances, the algorithm slightly overestimates (underestimates) the distance. This may be related to lens distortion, reduced resolution for larger distances, or the need for more precise calibration for larger distances. The curves for the relative errors show inverse trends, i.e., while the relative errors for the distances tend to increase with decreasing the distance, the relative errors for the terrain elevation tend to increase with increasing the distance, and more specifically, with decreasing the elevation. Both of these trends are according to the expectation. For smaller distances, any deviation from reality has a higher impact on the relative error of the estimated distance. Since the estimate for the elevation depends on the estimate for the distance, the errors for the elevation are a consequence of both error sources. Even though for larger distances, the relative errors for the distance typically decrease, the absolute errors slightly increase as shown in the figure. This error propagates and has a higher impact on the elevation relative error when the elevation is small. As shown in the 3D graphs, the relative errors for the distance range from around 2% to 25%, whereas the elevation relative errors range from around 0% to 35%.

Figure 4.24 shows the elevation distribution and relative errors for the same scenarios as before, this time with the assistance of the flying robot in providing the distance values. Although the median values for the estimates of the terrain elevation did not change, the dispersion of the elevations around the mean value was significantly reduced. In this layout, the average relative errors were also notoriously mitigated. Except for one data point, the relative errors were lower than 13% in all instances. The data point corresponding to an elevation of 25 cm and a distance of 4 m appears to be an outlier, where nearly no error was mitigated. This may be because the homography matrix responsible for mapping the distance into real-world coordinates was calibrated using reference points up to distances of 3 m.

Finally, in 10 runs, the processing of the images took on average between 0.54 s and 0.62 s.

4.4. CONCLUSIONS AND FUTURE RESEARCH

We leveraged the complementary capabilities of flying and ground robots, in order to deliver affordable, effective solutions based on image processing for unknown SaR environments. In particular, the following SaR tasks were considered: estimating the location and distance of a human from the ground robot based on the images captured by the robot, tracking the trajectory of a moving object by applying data fusion and a Kalman filter, and estimating the terrain elevation. In order to ensure a simplistic setup with low hardware costs, these tasks are performed via visual depiction only, through images and videos.

We performed real-life experiments in order to validate the proposed approaches, which are founded on YOLO. The experiments proved the efficiency of the proposed methods in the accurate estimation of the position of humans in various poses from their images. In fact, the average relative errors remained mainly below 10%. In tracking the trajectory of a moving object, the advantage of deploying a collaborative team of

a flying and a ground robot was evident, especially in properly re-generating the overall shape of the trajectory. Furthermore, this setup is more fault-safe in keeping a low root mean square error and in preventing large deviations from the ground truth, especially in cluttered environments. In fact, for the widest trajectory simulated and in the presence of obstacles to the aerial view, compared to the best-performing robot, the collaboration of the robots resulted in a reduction of 28% in the extreme deviation from the true trajectory, and an increase of 42% in the area covered ratio. Finally, the estimation of the terrain elevation was prone to two primary sources of error: error in the estimation of the distance and error in the subsequent elevation estimation. The distance estimation inaccuracies stem from simplified camera models and low image resolution, which impact object detection, compounded by occasional object occlusion. The elevation errors result from poor homography calibration in specific regions, which is exacerbated by the double application of the simplified camera model. Without the assistance of the flying robot in providing the distance of the human from the ground robot, the relative errors in estimation of the terrain elevation were up to 35% in some cases. This error was reduced to less than 13% when the flying robot estimated the distance for the ground robot.

In summary, these results represent notable progress in leveraging visual data and robotic capabilities for SaR. Since calibrating the homography matrix is expected to improve with the number of reference points, which slows down the computations, it is crucial to investigate the optimal number of reference points that provides a trade-off between accuracy and computational burden. Furthermore, the results hint that implementing a simplified pinhole camera model may result in significant errors. Thus, including a more realistic camera model that captures optical phenomena, e.g., lens distortion and parallax, is worth looking into. As an extension to the current work, the proposed algorithm for locating the victims should also be validated for non-conventional poses of the victims that are much different from the 7 poses that have been shown in Figure 4.16 and that are not common in the COCO dataset. Moreover, it will be advantageous to make the flying robot autonomously capable of identifying the regions of interest for scanning, in collaboration with the ground robot. This may require further interaction and exchange of data/information between the two robots. Finally, validating our proposed approaches in various real-life SaR scenarios, where a combination of pose estimation, victim localization, victim tracking, and terrain elevation mapping should be conducted, is a topic for further future work.

5

M2PFC: Multi-Agent Hierarchical MPC-Tuned Fuzzy Logic Control for Context-Aware Adaptation and Real-time Responsiveness

Multi-agent robotic systems offer significant potential for autonomous operation in dynamic and uncertain environments. This allows for scalable coordination and efficient decision making. In high-stakes applications, including Search-and-Rescue (SaR), autonomous robots can significantly enhance the mission efficiency by accelerating localisation and rescue of the trapped victims. However, coordinating such robots to autonomously map dynamic, unstructured environments without human supervision poses considerable decision making, robustness, and computational challenges. We introduce a novel hybrid, two-layer control architecture, called Multi-agent Model-Predictive Fuzzy logic Control (M2PFC), for autonomous mission planning in multi-robot systems, particularly suited for SaR missions. M2PFC integrates Fuzzy Logic Control (FLC) for real-time, local decision making, and Model Predictive Control (MPC) for global, event-triggered or periodic tuning of the FLC parameters. While MPC is typically computationally intensive, M2PFC avoids this bottleneck by decoupling high-level planning from low-level control: (i) The FLC layer executes control actions at a high frequency with minimal computa-

Parts of this chapter are under review in Engineering Applications of Artificial Intelligence.
M. Baglioni contributed to the conceptualization and the methodology, assessed the results, supervised the MSc student involved, and reviewed the final draft of the chapter. C. Maxwell performed the simulations, contributed to methodology and results, and wrote the first draft. A. Jamshidnejad contributed to the conceptualization and the methodology, assessed the results, edited the final draft, and performed supervision.
This chapter is written in British English instead of American English.

tional overhead, and (ii) the MPC layer, triggered less frequently, refines the FLC parameters using predicted global information. This architecture offers four main advantages: (1) Real-time efficiency *through lightweight FLC decision making. (2)* Global coordination and optimisation *via the MPC layer, which adjusts local behaviour to align with mission-level objectives. (3)* Hybrid intelligence*, where heuristic FLC decisions benefit from MPC-guided parameter tuning, which enhances local performance, without sacrificing scalability. (4)* Enhanced robustness*, achieved by integrating the uncertainty-handling capabilities of FLC, the structured planning of MPC, and the inherent distributed nature of M2PFC — together improving resilience of M2PFC in dynamic, uncertain environments. We validate M2PFC using a MATLAB-based agent-level simulator for disaster scenarios, built on a discrete grid environment model. Results obtained from extensive simulations demonstrate that M2PFC outperforms purely decentralised FLC in mission performance, while requiring substantially fewer computational resources and possessing enhanced robustness compared to fully centralised MPC. M2PFC, thus, achieves a favourable trade-off between autonomy, optimality, robustness, and computational efficiency in multi-robot SaR operations.*

5.1. INTRODUCTION

ROBOTS have been increasingly deployed in recent years to enhance safety and effectiveness of Search-and-Rescue (SaR) missions by contributing to various tasks, including mapping unknown environments and locating victims [14], [15], [19]. The main benefits of employing SaR robots include reduced operational cost, improved time efficiency, and lower risks for the SaR crew. While human operators rely on heuristics to handle uncertain and dynamic SaR environments, robots offer the potential to optimise mission planning and to systematically explore unknown areas using formal mathematical frameworks.

In this chapter, we introduce a novel mission planning framework for multi-agent systems, particularly designed for a team of autonomous robots that are tasked with mapping an unknown SaR environment. Our proposed approach introduces a two-layer control architecture, called Multi-agent Model-Predictive Fuzzy logic Control (M2PFC), which leverages heuristic-based, intelligent decision making and integrates coordinated multi-agent planning within a systematic optimisation-based framework. At the lower layer, Fuzzy Logic Control (FLC) — using expert-defined fuzzy rules — steers the local path planning of individual agents. At the top layer, Model Predictive Control (MPC) performs global coordination by dynamically tuning the parameters of the local FLC controllers of the agents in real time. This hierarchical integration of MPC and FLC enhances prediction and collective performance of the agents.

To evaluate the proposed control architecture, we develop a detailed mathematical model capturing the dynamics of outdoor disasters and SaR operations. Based on this, we construct an agent-based simulation environment that allows to model diverse SaR scenarios. These simulations are used to assess the effectiveness of M2PFC in steering multi-robot teams to autonomously and efficiently map environments that are affected by disasters.

5.1.1. MOTIVATIONS

Efficient, well-performing mission planning is vital for enabling autonomous robots to operate effectively in SaR scenarios. This section elaborates our main motivations for introducing M2PFC as an autonomous mission planning architecture for multi-robot SaR systems.

Deploying Robots for SaR: Natural and man-made disasters typically cause rapid and extensive damage across a widespread area, often leaving numerous victims trapped in hazardous situations. SaR operations aim to locate and safely rescue trapped victims as quickly as possible. Therefore, reducing the SaR mission time and increasing the area coverage to the utmost possibility are both critical SaR objectives [160]. However, several challenges complicate SaR missions: disaster zones may span very large areas, the locations of victims are typically unknown, and the SaR environment may involve dynamic and life-threatening hazards, such as fires, explosions, and unstable structures. Additionally, certain SaR regions may be inaccessible for human responders. These factors render SaR operations resource-intensive, time-consuming, and hazardous for human operators. Robots, when designed and equipped with appropriate algorithms for real-time response in uncertain SaR missions, offer a promising alternative to human involvement in such scenarios [18], [19].

Control Methodologies for SaR Robots: Robotic systems have the potential to significantly support SaR missions, especially by automating routine tasks, enhancing efficiency in resource allocation, and reducing the exposure of human operators to hazardous situations [300], [301]. In doing so, robots free human operators to focus on less risky, crucial tasks, e.g., providing medical assistance and coordinating logistics [302]. This underscores the need for advanced robotic control algorithms that ensure reliable autonomy and enhance effectiveness in disaster response.

FLC is a heuristic-based control method capable of incorporating expert knowledge into control systems without requiring precise mathematical models of system dynamics or large datasets. FLC is particularly effective for real-time decision making in systems that possess nonlinear, complex dynamics and that are prone to uncertainties. FLC offers low computational overhead, while maintaining interpretability as a white box approach [45].

In contrast, MPC provides a systematic framework for incorporating constraints into decision making and optimising multiple — often conflicting — objectives across a prediction horizon [20]. The complementary strengths of FLC and MPC make them well-suited for autonomous control of SaR robots.

Novel Hybrid Control Architecture for Multi-robot SaR: In this chapter, we introduce M2PFC, a hybrid architecture that exploits the intuition and responsiveness of FLC with the predictive, constraint-guaranteeing, and optimisation-based capabilities of MPC. Our main goal is to develop an intelligent, flexible, and time-efficient control strategy that enhances autonomy and performance of multi-robot SaR missions, particularly in systematic mapping of unknown or partially known environments.

Multi-agent systems are controlled through centralised, decentralised, or distributed control architectures [59]. Each architecture offers distinct trade-offs. Centralised control can yield globally optimal solutions, but is computationally intensive, especially for large-scale systems, and is vulnerable to single-point failures [60]. Decentralised control is more scalable and robust, but often lacks coordination when inter-agent dynamics are non-negligible, leading to sub-optimal system behaviour [61]. Distributed control seeks a balance by enabling local decision making with limited communication among agents that exhibit significant inter-agent dynamics. Distributed control, however, typically imposes high communication overhead and requires complex coordination algorithms, both being highly challenging in SaR contexts, where bandwidth and reliability are usually limited.

Our proposed M2PFC architecture addresses these limitations by integrating the strengths of centralised and decentralised paradigms through a two-layer structuring. Local autonomy is maintained using FLC, with global coherence achieved via MPC-based tuning, offering a scalable and robust dynamic mission planning method for multi-robot SaR that maintains high-performance functionality under adverse conditions.

5.1.2. MAIN CONTRIBUTIONS AND STRUCTURE OF THE CHAPTER

The main contributions of this chapter are outlined next:

- We introduce Multi-agent Model-Predictive Fuzzy logic Control (M2PFC), a novel hybrid two-layer control architecture for multi-agent systems, which integrates

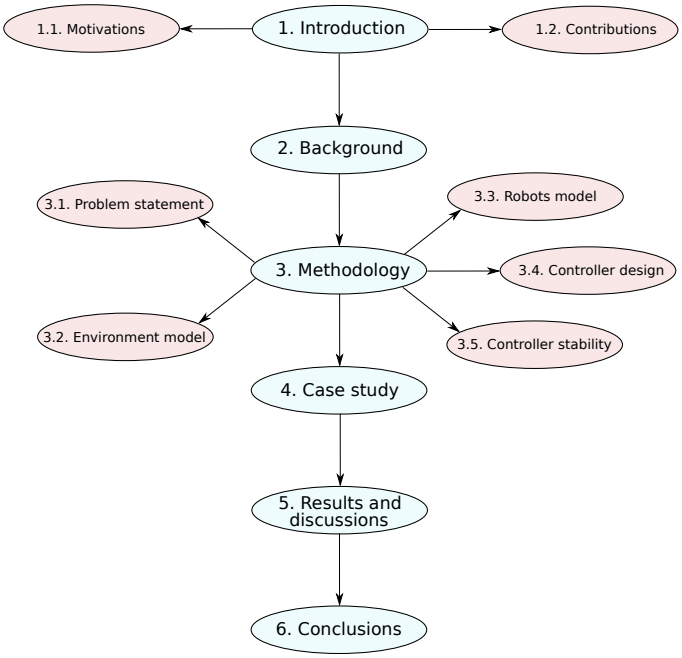


Figure 5.1: Roadmap for the sections of the chapter.

decentralised Fuzzy Logic Control (FLC) systems with a centralised Model Predictive Control (MPC) layer for periodic or event-triggered tuning of FLC parameters. M2PFC balances real-time responsiveness, leveraging human-inspired reasoning of FLC and computational efficiency of decentralised control, with strategic adaptation and coordination enabled by the global MPC layer. It effectively bridges the gap between scalable, context-aware, and robust autonomy in dynamic, uncertain environments.

- A key innovation of M2PFC lies in its hierarchical structure and time-scale separation: MPC does not directly generate control actions, but facilitates context-aware adaptation of local FLC controllers by re-optimising their parameters. While preserving decentralised autonomy, this indirect coordination mechanism reduces computational overhead and enhances robustness and extensibility to large-scale systems.

- We conduct a comprehensive case study for evaluating M2PFC in both centralised and decentralised configurations. We compare M2PFC with conventional MPC-only and pre-tuned FLC systems. The study includes multiple prediction strategies and extensive sensitivity analyses. Results demonstrate that M2PFC achieves a favourable trade-off between autonomy, optimality, robustness, and computational efficiency across diverse disaster scenarios and under varying environmental conditions.

The remainder of the chapter is organised as follows, and as outlined in the roadmap in Figure 5.1. Section 5.2 reviews the existing literature relevant for multi-agent control systems. Section 5.3 formulates the problem addressed in this chapter and details mathematical modelling of the disaster environment and agents, as well as the formulation of M2PFC. Section 5.4 describes the case study developed to evaluate the proposed approach. Section 5.5 presents and discusses the simulation results of the case study. Finally, Section 5.6 summarises key findings of the chapter and outlines directions for future research.

For reference, Table 5.8 in Appendix 5.7 lists the mathematical notations frequently used throughout the chapter, while Table 5.9 includes the abbreviations used in the chapter.

5.2. BACKGROUND

This chapter focuses on mission planning for a team of SaR flying robots, involving path planning for mapping the environment, detecting victims, and identifying hazards. Accordingly, this background section covers topics relevant for multi-robot SaR systems.

5.2.1. MULTI-AGENT CONTROL ARCHITECTURES

For multi-robot systems, effective coordination of robots is essential. Control strategies for mission planning of multi-robot systems are, generally, divided into *centralised*, *decentralised*, and *distributed*.

Centralised control manages all agents via a supervisory system and is suited when full system data is available and computationally intensive tasks should be offloaded to a ground station. Centralised control may yield larger area coverage, but often suffers from scalability issues in dynamic environments and communication constraints [60], [303]–[306].

With decentralised control, robots act independently based on local sensing. Decentralised control is preferred when communication and intense centralised computations are unreliable or costly, e.g., in environments with limited bandwidth or energy. While simple and scalable, decentralised methods may struggle with low sensor range and lack of global coordination [307]–[313].

Distributed control offers a middle ground where agents steered by local controllers share information to maintain coordination. Distributed control is well-suited for complex SaR tasks, but introduces additional computational complexity, due to the need for coordinating decisions of local controllers [30], [181], [314]–[319].

5.2.2. OPTIMISATION-BASED CONTROL STRATEGIES

Optimisation-based control approaches have demonstrated high effectiveness in controlling both single-robot and multi-robot systems in SaR missions [1], [25], [41]. These methods allow systematic optimisation of the mission objectives, e.g., minimising the time to locate victims, maximising the area coverage in unexplored environments, and minimising energy consumption to extend operational duration.

Within optimal control methods, linear quadratic control (LQC) is suitable for generating optimal trajectories that ensure safe and fast robot navigation [320]. Moreover,

stochastic control methods provide a framework for handling probabilistic objectives and constraints, particularly useful when system dynamics and environmental factors are uncertain [46], [321]. One example is path integral control, which suits disaster scenarios, offering computationally efficient trajectory planning [5].

Among optimal control approaches, MPC stands out as one of the most effective. MPC not only performs optimisation, but also allows systematic integration of hard and soft constraints — e.g., maintaining specific velocities or avoiding obstacles — into decision making [2]. MPC incorporates models of the robotic system and the SaR environment to predict future states of the system and to make informed decisions [1], [30]. Robust variants of MPC have been established [41], allowing reliable operation under uncertainties — an intrinsic challenge in dynamic and partially known SaR environments.

5.2.3. INTELLIGENT CONTROL STRATEGIES

Intelligent control methods play a crucial role in autonomous SaR robotics. These methods enable robotic teams to efficiently handle uncertainties, adapt to dynamic environments, and make real-time decisions. For instance, heuristic methods based on Fuzzy Logic (FL) have been proposed for prioritizing high-risk SaR regions during mapping the environment, allowing dynamic adjustment of the mission based on the likelihood of visiting victims [182]. Reinforcement learning, particularly in multi-agent frameworks, has shown promise in establishing cooperative operation of robots [185].

A wide range of heuristic methods has been adapted for multi-agent control systems. Bug algorithms [172], for instance, provide simple, effective strategies for obstacle avoidance and reaching targets. Potential field methods [173] offer reactive obstacle avoidance and target approaching by treating hazards as repellers and targets as attractors. Particle swarm optimisation (PSO) exploits the emergent coordination of swarms to optimise objectives of SaR missions, e.g., area coverage, energy efficiency, and target navigation [137]. FLC [48] handles imprecision and uncertainties in data by encoding existing expert knowledge into fuzzy rules. This makes FLC well-suited for complex SaR scenarios where precise models of the environment are unavailable.

FLC is often used for navigating robots. For example, authors in [322] propose a dual FLC framework for real-time path tracking and dynamic obstacle avoidance. FLC is chosen for this problem to handle the dynamic aspects of the navigation, since it has a responsive and adaptive behavior. Therefore, the proposed framework demonstrates adaptive trajectory generation. In addition, by decoupling the motion control into two dedicated fuzzy controllers, the method should also create collision-free trajectories, even if FLC does not provide hard constraints satisfaction. In fact, a limitation shown in the simulations is the ability to handle dynamic obstacles that only move in straight lines, making it difficult to generalize to real-world scenarios where obstacles have more complex trajectories. Sousa et al. [323] integrate fuzzy logic with a Convolutional Neural Network (CNN) to achieve obstacle avoidance for mobile robots in human-robot interaction contexts, enabling real-time adaptability and safe navigation near humans. The CNN performs object detection to define a dynamic safety zone for the robot, while fuzzy logic determines the speed of the robot accordingly. Therefore, FLC is employed in order to reactively adjust the robot velocity and orientation in order to avoid obstacles, similarly to [322], and although the system is supposed to account for dynamic obstacles,

this happens for relatively simple scenarios, thus further improvements in environmental complexity and obstacle diversity are needed for wider use of the approach. In [324], fuzzy logic is combined with A* path planning to design controllers for multiple autonomous vehicles that navigate unknown environments with dynamic obstacles. Compared to existing approaches, including particle swarm optimization, genetic algorithm, and artificial potential fields, the introduced combined framework shows improved performance in terms of path efficiency and mission time for a vehicle to reach its destination, in accordance with the nature of A* that generates shortest paths. Nonetheless, this framework has not been tested in highly cluttered environments with an increased number of obstacles, limiting the demonstrated robustness of the approach, therefore artificial potential fields method may continue to be preferable to FLC for obstacle avoidance.

5.2.4. HIERARCHICAL AND HYBRID ARCHITECTURES

Hierarchical and hybrid control systems involve, respectively, more than one layer of control and an integration of various decision making strategies. They, thus, enable combinations of various control categories (centralised, decentralised, distributed) and systematic and intelligent control strategies [63]. These architectures are particularly well-suited for complex SaR environments, where modularity and flexibility are essential.

Hierarchical control systems typically employ top layers for global planning and coordination of multi-agent systems, while lower layers manage local execution and reactive behaviour of agents. This results in enhanced scalability and operational robustness. Hybrid control architectures combine complementary methods to exploit their respective strengths. In particular, hybrid FLC-MPC systems have shown to improve performance over purely centralised systems [25]. A key reason is that FLC provides fast, interpretable, and computationally efficient responses to local uncertainties, while MPC enables constraint handling and long-term planning for global optimisation. As such, hybrid FLC-MPC systems effectively balance reactivity and foresight in multi-agent control. In this chapter, we also adopt a hybrid FLC-MPC architecture, introducing a novel, highly efficient, and operationally robust integration framework.

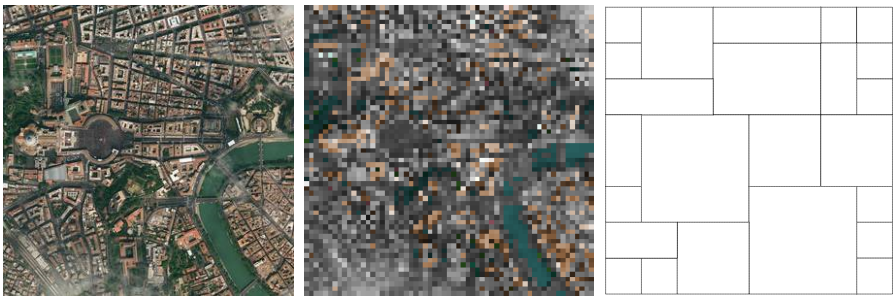
5.3. PROPOSED MULTI-AGENT CONTROL ARCHITECTURE: M2PFC

This section outlines the proposed Multi-agent Model-Predictive Fuzzy logic Control (M2PFC) architecture.

We first define the problem, including the key assumptions and the mathematical models used for the environment and robotic agents. We then formulate M2PFC in detail.

5.3.1. PROBLEM STATEMENT

We consider a discrete-space, discrete-time setting for both modelling and control. Let n^{rob} denote the number of flying robots that are tasked with exploring and mapping an outdoor SaR environment. After discretisation, the environment is represented by grid $\mathscr{E}$, which contains a given number of static and dynamic elements, e.g., victims, spread-



(a) (b) (c)

Figure 5.2: Illustrating a SaR environment: (a) 2D satellite image, (b) 2D representation discretised spatially into cells of identical size, and (c) 2D grid modelling with cells of varying sizes (merging cells with homogeneous or partially similar characteristics).

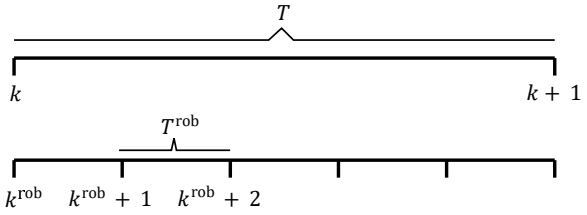


Figure 5.3: Different time scales used for modelling the SaR environment: All local time steps k^{rob} that fall in between the global simulation time steps k and $k+1$ are included in set $\mathbb{K}^{\text{rob}}(k)$.

ing fire, and debris (see Figure 5.2). Variable n^{victim} represents the maximum number of victims per cell in $\mathscr{E}$.

Time is discretised at two levels (see Figure 5.3):

1. Local sampling time T^{rob}, with step counter k^{rob}, governs the simulation of individual robot dynamics.
2. Global sampling time T, with step counter k, governs the evolution of the SaR environment model.

For simplicity, as is illustrated in Figure 5.3, we assume that T is an integer multiple of T^{rob}, and that the local and global step counters are initially aligned. All local time steps k^{rob} between two consecutive global time steps k and $k+1$ are indexed by set $\mathbb{K}^{\text{rob}}(k)$.

Each flying robot is represented by a state vector that includes its position and heading angle (i.e., counter-clockwise orientation from east) per time step, and is equipped with sensors capable of scanning the cells. At each local time step k^{rob}, a robot performs either of the tasks "traverse" or "scan" based on its current cell and action plan. If "scan" is performed, the robot generates a local map of the sub-area in $\mathscr{E}$. The scan duration depends on the capabilities of robot sensors and on the size of the scanned area.

Key assumptions used throughout this section include:

A1 The environment, thus grid $\mathscr{E}$, has finite dimensions.

A2 Robots incorporate no victim motion model, i.e., they assume that victims remain stationary throughout the entire SaR mission.

A3 The amount and distribution of debris remain unchanged during the SaR mission.

A4 Robots use a cellular-automaton-based model of fire propagation (detailed in Section 5.3.2) to update fire states of each cell — based on the perceived material properties and ignition time — categorised into five levels: non-flammable, flammable, igniting, burning, extinguished.

A5 Cells holding fire state "extinguished" do not reignite.

A6 Each robot, indexed r, travels across a straight line, with its maximum velocity $v_r(\cdot)$, from its current cell to a given target cell, flying at altitudes high enough for avoiding debris, obstacles, and other robots.

A7 When close to each other, robots fly at a different altitude to prevent collisions, where this action does not affect the mission planning.

A8 From any location in the environment, at least one feasible path to any other target cell exists for robots.

A9 When tasked with "scan", each robot scans exactly one cell per sampling time.

A10 Each time, a cell is scanned by maximum one robot.

A11 Robots have perfect self-localisation and state awareness at all time steps.

5.3.2. MODELLING THE SAR ENVIRONMENT

The SaR environment is discretised into a 2D grid $\mathscr{E}$ composed of individual cells. Figure 5.2a shows a satellite image of such an environment, which, in Figure 5.2b, is discretised into equally sized cells. As shown in Figure 5.2c, cells may vary in size after the final discretisation. In other words, the process begins with an initial uniform discretisation into equally sized cells (Figure 5.2b). These cells are then refined by clustering neighbouring cells — particularly those expected to be of similar importance for exploration and which, when merged, still fall within the scanning range of robots — into larger, composite cells.

The environment is modelled using an $n^{\mathrm{h}} \times n^{\mathrm{v}}$ matrix M^{grid}, called the *grid matrix*, with each element $m^{\mathrm{grid}}_{(i,j)}$ storing the coordinates (x_i, y_j) of the centre of cell (i, j), along with the horizontal $\ell^{\mathrm{h}}_{(i,j)}$ and vertical $\ell^{\mathrm{v}}_{(i,j)}$ dimensions of the corresponding cell. When cells vary in size — as in Figure 5.2c — the size $n^{\mathrm{h}} \times n^{\mathrm{v}}$ of the grid matrix M^{grid} is chosen based on the maximum number of cells along the horizontal and vertical directions. Grid elements that do not correspond to an actual cell in the discretised environment are assigned a value of zero.

SaR robots incrementally build and update a map of the environment that is represented through several matrices of size $n^{\mathrm{h}} \times n^{\mathrm{v}}$. These matrices encode information relevant to each cell every time step.

The following matrices define the environment map. Except for the first one, i.e., the "structure matrix", which is static, other matrices evolve in time.

STRUCTURE MATRIX

The structure matrix M^{struct} is time-invariant and its element (i, j), i.e., $m^{\mathrm{struct}}_{(i,j)}$, defines the combustibility score of cell $(i, j) \in \mathscr{E}$, based on the materials present in the cell and their relative proportions.

The combustibility scores range from 0 — fire-proof — to 1 — fully combustible — with 0.6 indicating fire-resistant materials. For cells embedding heterogeneous materials, the combustibility score is estimated as the product of the proportion of the most dominant material and its combustibility score. For example, a cell composed of 80% combustible and 20% fire-proof material has a combustibility score of 0.8.

SCAN CERTAINTY MATRIX

In the scan certainty matrix $M^{\mathrm{scan}}(k)$, the element (i, j) at time step k represents the scan certainty $m^{\mathrm{scan}}_{(i,j)}(k) \in [0, 1]$ for cell (i, j).

The scan certainty matrix is dynamic and is initialised at zero. The update rule for

elements $m^{\text{scan}}_{(i,j)}(k)$ of this matrix, with $i \in \{1,\dots,n^{\text{h}}\}$ and $j \in \{1,\dots,n^{\text{v}}\}$, follows:

$$m^{\text{scan}}_{(i,j)}(k) = \begin{cases} \max\left\{m^{\text{scan}}_{(i,j)}(k-1) - \sigma, 0\right\} \\ \quad \text{if cell } (i,j) \text{ is not scanned at simulation} \\ \quad \text{time step } k \\ \max\left\{m^{\text{scan}}_{(i,j)}(k-1) - \sigma, \eta\right\} \\ \quad \text{if cell } (i,j) \text{ is scanned at simulation} \\ \quad \text{time step } k \text{ by a SaR robot, for which} \\ \quad \text{the sensor accuracy is } \eta \end{cases} \tag{5.1}$$

Here we assume that, due to environmental dynamics, the scan certainties decay by a fixed ratio σ per time step, unless the cell is re-scanned. Moreover, $\eta \in [0,1]$ denotes the sensor accuracy of the scanning robot. The second branch of (5.1) ensures that when the sensor accuracy is lower than the current certainty, this higher certainty is restored.

5

VICTIM PROBABILITY MATRIX

The victim probability matrix $M^{\text{victim}}(k)$ includes the perceived likelihood of finding victims in cells of $\mathscr{E}$ at time step k. This dynamic matrix is initialised at zero and is updated every time step k for $i \in \{1,\dots,n^{\text{h}}\}$ and $j \in \{1,\dots,n^{\text{v}}\}$ according to:

$$m^{\text{victim}}_{(i,j)}(k) = \begin{cases} \dfrac{n^{\text{perceive}}_{(i,j)}(k)}{n^{\text{victim}}} m^{\text{scan}}_{(i,j)}(k) \\ \quad \text{if cell } (i,j) \text{ is scanned at time step } k \text{ and} \\ \quad n^{\text{perceive}}_{(i,j)}(k) > 0 \text{ number of victims are} \\ \quad \text{perceived in the cell} \\ 1 - m^{\text{scan}}_{(i,j)}(k) \\ \quad \text{if cell } (i,j) \text{ is scanned at time step } k \text{ and no} \\ \quad \text{victim is perceived in the cell, i.e., } n^{\text{perceive}}_{(i,j)}(k) = 0 \\ m^{\text{victim}}_{(i,j)}(k-1) \dfrac{m^{\text{scan}}_{(i,j)}(k)}{m^{\text{scan}}_{(i,j)}(k-1)} \\ \quad \text{if cell } (i,j) \text{ is not scanned at time step } k \end{cases} \tag{5.2}$$

Here $n^{\text{perceive}}_{(i,j)}(k)$ represents the perceived number of victims within cell (i,j) at time step k, with $n^{\text{perceive}}_{(i,j)}(k) \in \{1,\dots,n^{\text{victim}}\}$.

According to Assumption **A2**, the robots rely only on the information perceived through their sensors about victims and do not incorporate any pedestrian dynamics model into the update equations for $M^{\text{victim}}(k)$.

DEBRIS OCCUPANCY MATRIX

The debris occupancy matrix $M^{\text{debris}}(k)$ captures the perceived proportion $o_{(i,j)}(k) \in [0,1]$ of each cell (i,j) that is occupied by debris (e.g., rubble, walls, buildings.).

Although, according to Assumption **A3**, status of debris in $\mathscr{E}$ remains unchanged, perception through sensors may be refined over time. Thus, the debris occupancy matrix is updated per time step k for all $i \in \{1,...,n^{\mathrm{h}}\}$ and $j \in \{1,...,n^{\mathrm{v}}\}$ according to:

$$m^{\mathrm{debris}}_{(i,j)}(k) = \begin{cases} o_{(i,j)}(k)\, m^{\mathrm{scan}}_{(i,j)}(k) \\ \quad \text{if cell } (i,j) \text{ is scanned at time step } k \text{ and} \\ \quad o_{(i,j)}(k) > 0 \text{ debris proportion is perceived} \\ 1 - m^{\mathrm{scan}}_{(i,j)}(k) \\ \quad \text{if cell } (i,j) \text{ is scanned at time step } k \text{ and} \\ \quad \text{no debris is perceived, i.e., } o_{(i,j)}(k) = 0 \\ m^{\mathrm{debris}}_{(i,j)}(k-1)\dfrac{m^{\mathrm{scan}}_{(i,j)}(k)}{m^{\mathrm{scan}}_{(i,j)}(k-1)} \\ \quad \text{if cell } (i,j) \text{ is not scanned at time step } k \end{cases} \tag{5.3}$$

WIND VELOCITY MATRIX

The wind velocity matrix $M^{\mathrm{velocity}}(k)$ holds the magnitudes of the wind velocity, with $m^{\mathrm{velocity}}_{(i,j)}(k)$ denoting the wind velocity in cell (i,j) at time step k.

We assume that the magnitude of the wind velocity is externally measured and provided to robots every time step.

This matrix is required for modelling the fire spread, as explained in Section 5.3.2.

WIND DIRECTION MATRIX

The wind direction matrix $M^{\mathrm{direction}}(k)$ stores the wind angle $m^{\mathrm{direction}}_{(i,j)}(k)$ in degrees — measured counter-clockwise from the east — in each cell (i,j) at time step k.

As with the wind velocity, we assume that the direction of the wind is externally measured and provided to robots per time step.

Similarly, this matrix is required for modelling the fire spread, as explained in Section 5.3.2.

FIRE STATE MATRIX

Element $m^{\mathrm{fire}}_{(i,j)}(k)$ of fire state matrix $M^{\mathrm{fire}}(k)$ represents the fire state of cell (i,j) at time step k. Each fire state is encoded as one of the five values listed in Table 5.1: 0 (non-flammable), 1 (flammable), 2 (igniting), 3 (burning), and 4 (extinguished).

At the beginning of the SaR mission, fire states are mainly initialised to 0 or 1, depending on whether or not the cell is flammable. A small number of cells are initialised to fire state 2, representing ignition points where the fire begins. The fire states are then dynamically updated using the model explained next.

The fire propagation is modelled using a discrete cellular automaton approach based on Ohgai et. al. [151] and Freire and DaCamara [150]. The probability $\pi^{\mathrm{fire}}_{(i,j),(l,q)}(k)$ that fire spreads to cell (i,j) from a neighbouring cell (l,q) at time step k is given by:

$$\pi^{\mathrm{fire}}_{(i,j),(l,q)}(k) = \alpha_1 m^{\mathrm{struct}}_{(i,j)} o_{(i,j)}(k) f_{(l,q)}(k) \cdot \exp\left(\alpha_2 \delta\big((l,q),(i,j)\big) + \alpha_3 \bar{v}_{(i,j),(l,q)}(k) + \alpha_4 \bar{v}_{(i,j),(l,q)}(k) \cos\psi_{(i,j),(l,q)}(k)\right) \tag{5.4}$$

Table 5.1: Different fire states per cell

Realisation	State	Description
0	Non-flammable	The cell cannot catch fire
1	Flammable	The cell is not burning yet, but can potentially catch fire
2	Igniting	The cell has just ignited, but cannot yet spread the fire
3	Burning	The cell is on fire and can spread the fire
4	Extinguished	The fire in the cell has been extinguished and cannot reignite

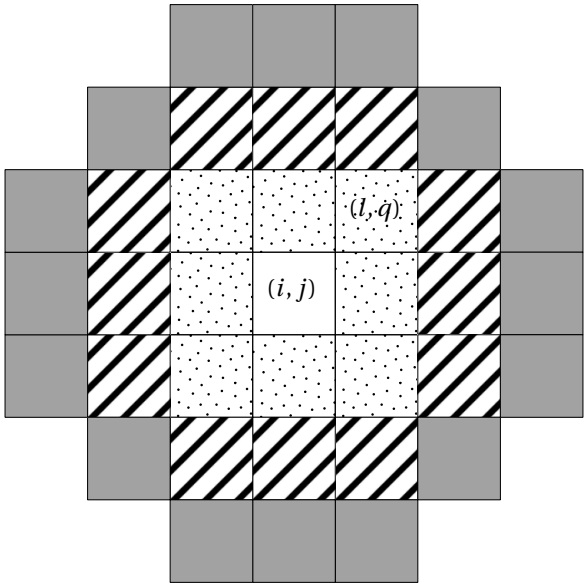


Figure 5.4: Neighbouring cells (l,q) of cell (i,j), from which fire may spread to cell (i,j) depending on the wind velocity, where this includes the dotted cells for $m^{\text{velocity}}_{(l,q)}(k) < 1\ \frac{\text{m}}{\text{s}}$, the dotted and lined cells for $1\ \frac{\text{m}}{\text{s}} \leq m^{\text{velocity}}_{(l,q)}(k) \leq 5\ \frac{\text{m}}{\text{s}}$, and the dotted, lined, and grey cells for $m^{\text{velocity}}_{(l,q)}(k) > 5\ \frac{\text{m}}{\text{s}}$.

where, depending on the wind conditions, for the neighbouring cell (l,q) we have:

- (l,q) belongs to the set of dotted cells in Figure 5.4, when $m^{\text{velocity}}_{(l,q)}(k) < 1\ \frac{\text{m}}{\text{s}}$
- (l,q) belongs to the set of dotted and lined cells in Figure 5.4, when $1\ \frac{\text{m}}{\text{s}} \leq m^{\text{velocity}}_{(l,q)}(k) \leq 5\ \frac{\text{m}}{\text{s}}$
- (l,q) belongs to the set of dotted, lined, and grey cells in Figure 5.4, when $m^{\text{velocity}}_{(l,q)}(k) > 5\ \frac{\text{m}}{\text{s}}$

In (5.4), α_1, α_2, α_3, α_4 are tuning parameters, $\delta : \mathscr{E}^2 \to \mathbb{R}$ is a distance metric between the source and target cells of fire spread, $\bar{v}_{(i,j),(l,q)}(k)$ is the average magnitude of the wind velocity between cells (i,j) and (l,q) at time step k (i.e., mean of $m^{\text{velocity}}_{(l,q)}(k)$ and $m^{\text{velocity}}_{(i,j)}(k)$), $\psi_{(i,j),(l,q)}(k)$ is the angle between the average wind direction (i.e., mean of $m^{\text{direction}}_{(l,q)}(k)$ and $m^{\text{direction}}_{(i,j)}(k)$) and the direction of the fire propagation, and $f_{(l,q)}(k)$ is the spreading potential of cell (l,q) to its neighbouring cells, including cell (i,j), at time step k. This function is computed via:

$$f_{(l,q)}(k) = \begin{cases} \dfrac{4\left(kT - t^{\text{fire}}_{(l,q)}\right) + 0.2\,t^{\text{burnt-out}}_{(l,q)} - 4.2\,t^{\text{spread}}_{(l,q)}}{t^{\text{burnt-out}}_{(l,q)} - t^{\text{spread}}_{(l,q)}} \\ \text{if } t^{\text{spread}}_{(l,q)} + t^{\text{fire}}_{(l,q)} \leq kT \leq 0.2\,t^{\text{burnt-out}}_{(l,q)} + 0.8\,t^{\text{spread}}_{(l,q)} + t^{\text{fire}}_{(l,q)} \\ 1.25\dfrac{t^{\text{burnt-out}}_{(l,q)} - kT + t^{\text{fire}}_{(l,q)}}{t^{\text{burnt-out}}_{(l,q)} - t^{\text{spread}}_{(l,q)}} \\ \text{if } 0.2\,t^{\text{burnt-out}}_{(l,q)} + 0.8\,t^{\text{spread}}_{(l,q)} + t^{\text{fire}}_{(l,q)} \leq kT \leq t^{\text{burnt-out}}_{(l,q)} + t^{\text{fire}}_{(l,q)} \end{cases} \tag{5.5}$$

where $t^{\text{spread}}_{(l,q)}$ and $t^{\text{burnt-out}}_{(l,q)}$ are the elapsed time (in the continuous domain) after ignition of cell (l,q), when it becomes able to spread the fire and when it is burnt out, respectively. Based on [151], $t^{\text{spread}}_{(l,q)}$ and $t^{\text{burnt-out}}_{(l,q)}$ are set to, respectively, 2 minutes and 10 minutes. Moreover, $t^{\text{fire}}_{(l,q)}$ is the time instant (in the continuous domain) when cell (l,q) ignites.

Finally, the fire state $m^{\text{fire}}_{(i,j)}(k)$ of cell (i,j) is updated via:

$$m^{\text{fire}}_{(i,j)}(k) = f^{\text{fire}}\left(m^{\text{fire}}_{(l,q)}(k), m^{\text{fire}}_{(i,j)}(k-1)\right) = \tag{5.6}$$

$$\begin{cases} 2 & \text{If } m^{\text{fire}}_{(i,j)}(k-1) = 1 \text{ and } m^{\text{fire}}_{(l,q)}(k) = 3 \\ & \quad \text{and } \pi^{\text{fire}}_{(i,j),(l,q)}(k) > \zeta \\ 3 & \text{If } m^{\text{fire}}_{(i,j)}(k-1) = 2 \text{ and} \\ & \quad kT \geq t^{\text{fire}}_{(i,j)} + t^{\text{spread}}_{(i,j)} \\ 4 & \text{If } m^{\text{fire}}_{(i,j)}(k-1) = 3 \text{ and} \\ & \quad kT \geq t^{\text{fire}}_{(i,j)} + t^{\text{burnt-out}}_{(i,j)} \\ m^{\text{fire}}_{(i,j)}(k-1) & \text{otherwise} \end{cases}$$

where ζ is the ignition threshold.

Fire Risk Time: We define the fire risk time $t^{\text{risk}}_{(i,j)}(k)$ for all cells (i,j) that possess fire state $m^{\text{fire}}_{(i,j)}(k) = 1$ at time step k, as the time estimated until cell (i,j) ignites. Note that while $t^{\text{fire}}_{(i,j)}$ indicates the real time of ignition for cell (i,j), the fire risk time $t^{\text{risk}}_{(i,j)}(k)$ is a prediction about this ignition time made at time step k, considering the worst case, i.e., the soonest possible ignition.

The fire risk time is determined based on the fire states of all neighbouring cells of cell (i,j) (see Figure 5.4), their distance from cell (i,j), and the average wind velocity in the region that embeds cell (i,j) and its neighbouring cells, as illustrated in Figure 5.4 (see Appendix 5.8, Algorithm 4, for estimating the fire risk time for our example case study assuming $\zeta = 0$, i.e., when cells are assumed to immediately ignite upon the reach of fire). For all other cells with fire state non-equal to 1, the fire risk time is set to zero.

Remark 10. ***Translating Local Maps into Global Maps:*** *Within the multi-robot SaR system, each robot $r \in \{1,\dots,n^{rob}\}$ locally scans and updates information related to the cells assigned to it by its local controller to scan along its path. As such, these robots update their local maps according to (5.1)–(5.6) within their local time setting.*

Then, the global environmental matrices at global time step k are constructed by aggregating these local updates, where this aggregation is mathematically represented by:

$$m^{\epsilon}_{(i,j)}(k) = \mu^{\epsilon,local}_{(i,j),r}\left(k^{rob}\right), \quad k^{rob} = \max_{\kappa \in \mathbb{K}^{rob}(k)} \kappa \tag{5.7}$$

with $\epsilon \in \{scan, victim, debris, fire\}$ and $\mu^{\epsilon,local}_{(i,j),r}(k^{rob})$ denoting the value of element (i,j) in the corresponding local matrix maintained by robot r at the most recent local time step k^{rob} belonging to set $\mathbb{K}^{rob}(k)$.

*Note that (5.7) reflects the scanning rationale explained in Assumption **A10**.*

5.3.3. MODELLING DYNAMICS OF SAR ROBOTS

This section explains modelling the task execution and motion dynamics of SaR robots.

TASK EXECUTION

Each SaR robot $r \in \{1,\dots,n^{\text{rob}}\}$ positioned at $\left(\rho^{\text{h}}_r\left(k^{\text{ctrl}}\right), \rho^{\text{v}}_r\left(k^{\text{ctrl}}\right)\right)$ is assigned a target cell $\left(\tau^{\text{h}}_r\left(k^{\text{ctrl}}\right), \tau^{\text{v}}_r\left(k^{\text{ctrl}}\right)\right)$ by its local controller at control time step k^{ctrl}. For brevity, we define notations $\boldsymbol{\tau} := \left(\tau^{\text{h}}_r\left(k^{\text{ctrl}}\right), \tau^{\text{v}}_r\left(k^{\text{ctrl}}\right)\right)$ and $\boldsymbol{\rho} := \left(\rho^{\text{h}}_r\left(k^{\text{ctrl}}\right), \rho^{\text{v}}_r\left(k^{\text{ctrl}}\right)\right)$, for coordinates of the target cell and current position of robot r, respectively.

The objective of the robot is to reach this target cell as quickly as possible in order to perform a scan. Based on Assumption **A6**, it travels across a straight line from its current location $\boldsymbol{\rho}$ to the target cell $\boldsymbol{\tau}$, moving at its maximum feasible velocity $v_r\left(k^{\text{rob}}\right)$, with k^{rob} the local time step corresponding to the current control time step k^{ctrl}. Along its path, the robot performs the action "traverse", unless it is located at the target cell, where the action is "scan".

Suppose that the robot is given a maximum of $n_r\left(k^{\text{rob}}\right) \in \mathbb{N}$ sampling times of duration T^{rob} to reach its target cell. This number should be selected to ensure feasibility for reaching the given target with any admissible velocity profile for the robot. The set

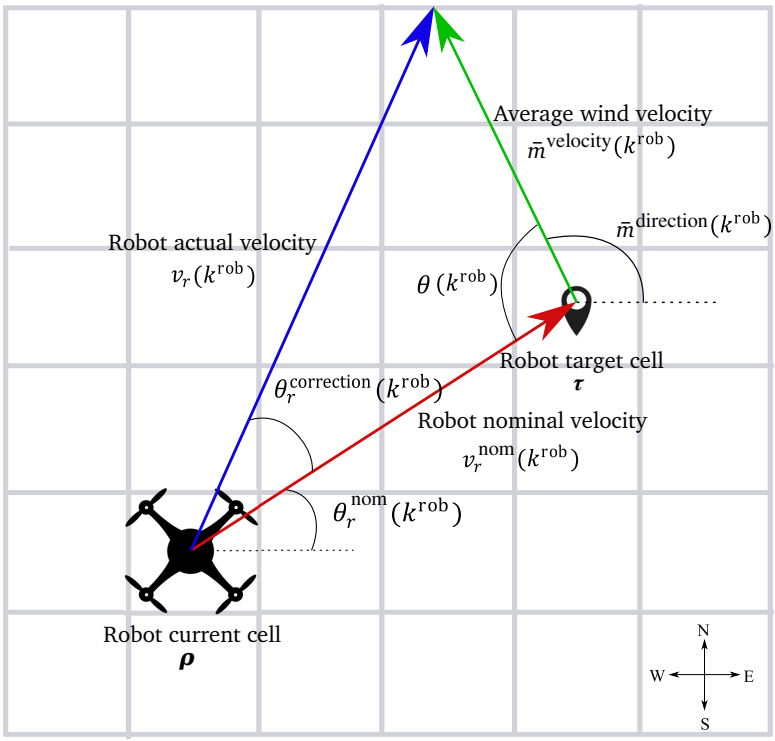


Figure 5.5: Corrected heading angle for achieving a desired velocity for the flying robot that, despite the wind flow, allows the robot to fly in the desired direction (aligned with the nominal velocity).

$E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$ of cells that robot r can feasibly reach within this given time from its current position $\boldsymbol{\rho}$ is given by:

$$E_r^{\text{feasible}}\left(k^{\text{rob}}\right) = \left\{(i,j) \in \mathscr{E} \middle| \delta\left((i,j),\boldsymbol{\rho}\right) \leq v_r\left(k^{\text{rob}}\right) \cdot n_r\left(k^{\text{rob}}\right) \cdot T^{\text{rob}}\right\} \tag{5.8}$$

where $v_r\left(k^{\text{rob}}\right)$ is the maximum velocity of robot r at local time step k^{rob}, and $\delta(\cdot,\cdot)$ is a distance metric. We must have $\boldsymbol{\tau} \in E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$.

When executing the "scan" action, the robot remains in the target cell $\boldsymbol{\tau}$ for a duration given by:

$$t_r^{\text{scan}}\left(k^{\text{rob}}\right) = \bar{t}_r \cdot \ell_{\boldsymbol{\tau}}^{\text{h}} \cdot \ell_{\boldsymbol{\tau}}^{\text{v}} \tag{5.9}$$

where $\bar{t}_r$ is a finite magnitude for the robot-specific scan time per unit area, and $\ell_{\boldsymbol{\tau}}^{\text{h}}$ and $\ell_{\boldsymbol{\tau}}^{\text{v}}$ are, respectively, the horizontal and vertical lengths of the target cell. Each robot r scans its target cells using a sensor with accuracy $\eta_r \in [0,1]$ to update its local scan certainty matrix $M_r^{\text{scan}}\left(k^{\text{rob}}\right)$, explained in Section 5.3.2.

MOTION DYNAMICS

Flying robots follow a 2D motion model adapted from [325], where the position of the robot is represented by the cell over which the robot flies. This section describes how the

state vector of each robot — including its position and heading angle, as introduced in Section 5.3.1 — is updated.

Upon assignment of a new target, the robot moves with its maximum velocity $v_r\left(k^{\text{rob}}\right)$, which is influenced by wind conditions and is derived from its nominal velocity $v_r^{\text{nom}}\left(k^{\text{rob}}\right)$. The average wind velocity and direction over the area corresponding to the feasible set $E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$ of cells for robot r are obtained through the following averages:

$$\bar{m}^{\text{velocity}}\left(k^{\text{rob}}\right) = \underset{(i,j)\in E_r^{\text{feasible}}\left(k^{\text{rob}}\right)}{\text{mean}} \left\{ m_{(i,j)}^{\text{velocity}}(k) \right\} \tag{5.10}$$

$$\bar{m}^{\text{direction}}\left(k^{\text{rob}}\right) = \underset{(i,j)\in E_r^{\text{feasible}}\left(k^{\text{rob}}\right)}{\text{mean}} \left\{ m_{(i,j)}^{\text{direction}}(k) \right\} \tag{5.11}$$

with k the most recent global time step associated with local time step k^{rob}. Additionally, the values of $m_{(i,j)}^{\text{velocity}}(k)$ and $m_{(i,j)}^{\text{direction}}(k)$ are obtained from the wind velocity and wind direction matrices, explained in Sections 5.3.2 and 5.3.2. Then, the actual, maximum velocity of the robot, using the law of cosines based on Figure 5.5, is computed by:

$$\begin{aligned} v_r\left(k^{\text{rob}}\right) = \Bigg(& \left(\bar{m}^{\text{velocity}}\left(k^{\text{rob}}\right)\right)^2 + \left(v_r^{\text{nom}}\left(k^{\text{rob}}\right)\right)^2 - \\ & 2\bar{m}^{\text{velocity}}\left(k^{\text{rob}}\right) \cdot v_r^{\text{nom}}\left(k^{\text{rob}}\right) \cdot \cos\left(\theta\left(k^{\text{rob}}\right)\right) \Bigg)^{0.5} \end{aligned} \tag{5.12}$$

where the angle $\theta\left(k^{\text{rob}}\right)$ between the nominal velocity of the robot and the average wind angle is given by:

$$\theta\left(k^{\text{rob}}\right) = 180° - \left(\bar{m}^{\text{direction}}\left(k^{\text{rob}}\right) - \theta_r^{\text{nom}}\left(k^{\text{rob}}\right)\right) \tag{5.13}$$

Suppose that at local time step k^{rob}, robot r is located at $\boldsymbol{\rho} := \left(\rho_r^{\text{h}}\left(k^{\text{rob}}\right), \rho_r^{\text{v}}\left(k^{\text{rob}}\right)\right)$ and is targeting cell $\boldsymbol{\tau} := \left(\tau_r^{\text{h}}\left(k^{\text{rob}}\right), \tau_r^{\text{v}}\left(k^{\text{rob}}\right)\right) \in E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$. Then, $\theta_r^{\text{nom}}\left(k^{\text{rob}}\right)$, which is the desired direction of motion in absence of wind, is given by:

$$\theta_r^{\text{nom}}\left(k^{\text{rob}}\right) = \arctan\left(\frac{\tau_r^{\text{v}}\left(k^{\text{rob}}\right) - \rho_r^{\text{v}}\left(k^{\text{rob}}\right)}{\tau_r^{\text{h}}\left(k^{\text{rob}}\right) - \rho_r^{\text{h}}\left(k^{\text{rob}}\right)}\right) \tag{5.14}$$

The actual heading $\theta_r^{\text{heading}}\left(k^{\text{rob}}\right)$ of the robot (measured counter-clockwise from east; see Figure 5.5) is computed via:

$$\theta_r^{\text{heading}}\left(k^{\text{rob}}\right) = \theta_r^{\text{nom}}\left(k^{\text{rob}}\right) + \theta_r^{\text{correction}}\left(k^{\text{rob}}\right) \tag{5.15}$$

where $\theta_r^{\text{correction}}\left(k^{\text{rob}}\right)$, which accounts for deviation due to including the wind influence, is given by:

$$\theta_r^{\text{correction}}\left(k^{\text{rob}}\right) = \arcsin\left(\frac{\bar{m}^{\text{velocity}}\left(k^{\text{rob}}\right)}{v_r\left(k^{\text{rob}}\right)} \cdot \sin\left(\theta_r^{\text{nom}}\left(k^{\text{rob}}\right) - \bar{m}^{\text{direction}}\left(k^{\text{rob}}\right)\right)\right) \tag{5.16}$$

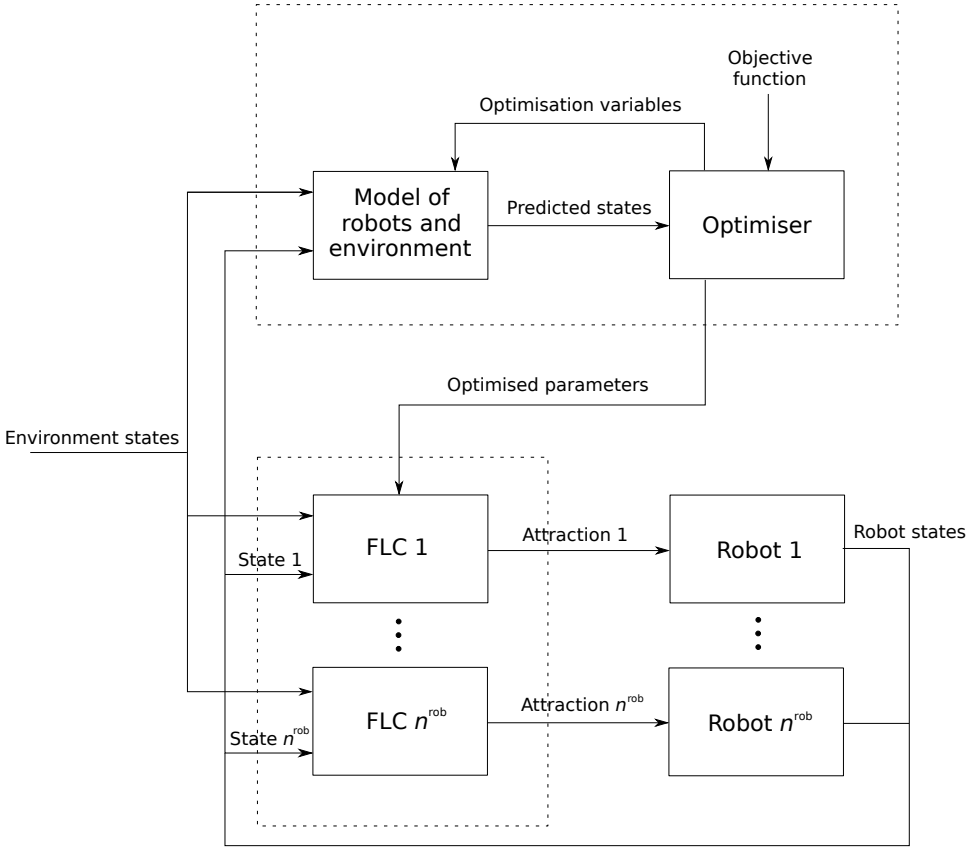


Figure 5.6: M2PFC architecture at time step k^{ctrl}, with two layers specified by dashed rectangles: The FLC controllers control the multi-agent system directly, in a decentralised way. The model predictive control layer tunes, with a frequency that is generally lower than the control frequency, the parameters of the local FLC controller in order to improve the global performance of the multi-agent system.

Finally, the time $t^{\text{travel}}_{\boldsymbol{\tau},r}\left(k^{\text{rob}}\right)$ required by the robot to reach the target cell is:

$$t^{\text{travel}}_{\boldsymbol{\tau},r}\left(k^{\text{rob}}\right) = \frac{\delta\left(\boldsymbol{\rho},\boldsymbol{\tau}\right)}{v_r\left(k^{\text{rob}}\right)} \tag{5.17}$$

which allows to update the position of the robot to $\boldsymbol{\tau}$ at local time step $k^{\text{rob}} + \lceil t^{\text{travel}}_{\boldsymbol{\tau},r}\left(k^{\text{rob}}\right) / T^{\text{rob}} \rceil$, where $\lceil\cdot\rceil$ generates the smallest integer larger than or equal to a given real number.

5.3.4. ADAPTIVE, TIME-EFFICIENT STRATEGY OF M2PFC

The proposed M2PFC architecture is a two-layer hierarchical system designed to balance computational efficiency with effective coordination in multi-agent systems, particularly multi-robot SaR (see Figure 5.6). M2PFC integrates local, real-time decision making with global, predictive optimisation.

At the lower layer, each robot is governed by a decentralised FLC controller, which enables fast, heuristic-based decision making and navigation. Each FLC controller operates autonomously and independently at every control time step, exploiting its current local information.

The higher layer consists of a centralised MPC module, which tunes decentralised local FLC controllers based on system-wide performance metrics. This layer leverages a global map of the environment and a model of the multi-agent system over a finite prediction horizon.

The hybrid structure of M2PFC preserves real-time responsiveness at the local level, while enabling global strategic adaptation and coordination.

Next, we detail the components and operation of the M2PFC architecture and its two layers.

For the multi-robot SaR system, the main goal of M2PFC is to maximise the number of cells that are visited, with higher chances of embedding victims and holding higher risks of catching fire.

CONTROL ARCHITECTURE

M2PFC (see Figure 5.6) operates within a discrete-time framework, with a control sampling time denoted by T^{ctrl} and a corresponding control time step index k^{ctrl}. For simplicity, we assume that T^{ctrl} is an integer multiple of the local (i.e., robot-level simulation) sampling time T^{rob}.

At the lower layer, path generation is performed locally and independently for each robot, at every control time step using its associated FLC controller.

In contrast, at the higher layer, MPC tunes specific parameters of local FLC controllers, either periodically or in response to certain performance criteria. This tuning is based on a globally optimal perspective over a finite prediction horizon, and occurs less frequently than local control updates. This allows to reduce computational load of M2PFC, while enhancing performance.

DECENTRALISED FLC LAYER

Each robot is equipped with its own local FLC controller for path planning, operating in a decentralised way, without communication or coordination with local controllers of other robots.

A target cell $\boldsymbol{\tau} := \left(\tau_r^{\text{h}}\left(k^{\text{ctrl}}\right), \tau_r^{\text{v}}\left(k^{\text{ctrl}}\right)\right)$ per robot r is determined every control time step k^{ctrl} via its FLC controller, based on the "attraction" of cells within feasible region $E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$ of the robot. The attraction value $a_{\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right)$ of candidate target cell $\boldsymbol{\tau} \in E_r^{\text{feasible}}\left(k^{\text{ctrl}}\right)$ for robot r at control time step k^{ctrl} is generated based on $c_{i,\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right)$, the input value for characteristic i of the cell, for all $i \in \{1,\ldots,n^{\text{ch}}\}$ with n^{ch} the total number of cell characteristics that define its attraction.

In the context of SaR multi-robot systems, cell characteristics injected into local FLC systems include:

1. **Cell Response Efficiency** – *identified by* $t_{\boldsymbol{\tau},r}^{travel}\left(k^{ctrl}\right)$ *and* $t_r^{scan}\left(k^{ctrl}\right)$*:* Time required for robot r at control time step k^{ctrl} — computed by (5.17) — to reach cell $\boldsymbol{\tau}$, and to perform scan — computed by (5.9) — when assigned as a task to the robot;

2. **Cell Priority** – *identified by* $m_{\boldsymbol{\tau}}^{victim}\left(k^{ctrl}\right)$*:* The probability of a victim being present in cell $\boldsymbol{\tau}$ at time step k^{ctrl}, given by (5.2) (see Section 5.3.2);

3. **Cell Fire Risk Time** – *identified by* $t_{\boldsymbol{\tau}}^{risk}\left(k^{ctrl}\right)$*:* The remaining time until cell $\boldsymbol{\tau}$ is expected to ignite, as explained in Section 5.3.2 (also see Algorithm 4 in Appendix 5.8);

4. **Cell Scan Certainty** – *identified by* $m_{\boldsymbol{\tau}}^{scan}\left(k^{ctrl}\right)$*:* A measure of how confidently cell $\boldsymbol{\tau}$ has most recently been scanned, evaluated using (5.1).

Using a type-1 Takagi-Sugeno-Kang (TSK) fuzzy inference system [326], these input cell characteristics are mapped onto a linear output surface. The fuzzy rules are parameterised to support adaptability, with each rule indexed by ℓ formulated as:

If $c_{1,\boldsymbol{\tau},r}\left(k^{ctrl}\right)$ *is* $\tilde{A}_{1,\ell}$ *and* $\ldots$ *and* $c_{n^{ch},\boldsymbol{\tau},r}\left(k^{ctrl}\right)$ *is* $\tilde{A}_{n^{ch},\ell}$*, then cell attraction is* $a_{\boldsymbol{\tau},r}^{(\ell)}\left(k^{ctrl}\right)$*.*

Here $\tilde{A}_{i,\ell}$ represents a fuzzy set that associates input characteristic i with a verbal term (e.g., "low response efficiency" or "high priority"), and the superscript (ℓ) on the attraction output denotes the contribution of fuzzy rule ℓ to the final estimate of this variable. Considering $f_{\ell}^{\text{TSK}}(\cdot)$ as the output generator of the rule, we have:

$$a_{\boldsymbol{\tau},r}^{(\ell)}\left(k^{\text{ctrl}}\right) = f_{\ell}^{\text{TSK}}\left(c_{1,\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right),\ldots,c_{n^{\text{ch}},\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right),\boldsymbol{\theta}_{r,\ell}\left(k^{\text{ctrl}}\right)\right) \tag{5.18}$$

where $\boldsymbol{\theta}_{r,\ell}(k^{\text{ctrl}}) \in \mathbb{R}^{n^{\text{ch}}+1}$ is the vector of all output parameters of rule ℓ within the TSK-FLC system used by local controller of robot r. This vector is in general time-varying, tuned via the supervisory MPC layer for enhanced performance of local TSK-FLC controllers.

The overall cell attraction $a_{\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right)$ is then computed by aggregating the outputs of all applicable fuzzy rules, weighted by their respective normalised *firing strengths* [326] that add up to unity, as per the TSK methodology.

CENTRALISED MPC LAYER

Within the M2PFC framework, global coordination of decentralised robots is achieved by MPC, which plays a supervisory role by re-tuning the parameters of local FLC controllers assigned to each robot. The MPC layer within the M2PFC framework is structured around two key components: an optimiser, which seeks the best possible configuration of local FLC tuning parameters to maximise a predefined objective function, and a prediction model that informs the optimiser about the impact of candidate configuration parameters on the objective function within the prediction window (see Figure 5.6).

At every control time step k^{ctrl}, the prediction model receives the current states of all robots, as well as the updated environmental matrices (see Section 5.3.2 for details). The optimiser proposes a candidate vector $\tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right)$ including tuning parameters for all local FLC controllers. These parameters affect the behaviour of each robot, and the prediction model evaluates their influence on the global objective across the prediction horizon.

In environments with inherent uncertainty, predictions generated by this model may not be perfectly accurate. To address this, robust MPC variants designed to account for model imperfections and environmental uncertainties, may be deployed [31], [220].

When re-tuning is triggered, the M2PFC supervisory layer solves the following constrained optimisation problem at control time k^{ctrl}:

$$J^{\text{MPC}^*}\left(\boldsymbol{r}^{\text{rob}}\left(k^{\text{ctrl}}\right), \mathscr{M}\left(k^{\text{ctrl}}\right)\right) = \max_{\tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right)} \sum_{\kappa=k^{\text{ctrl}}}^{k^{\text{ctrl}}+N^{\text{p}}} \sum_{r=1}^{n^{\text{rob}}} f_r^{\text{cell}}\left(\hat{a}_{\boldsymbol{\tau},r}\left(\kappa | k^{\text{ctrl}}\right)\right) \tag{5.19}$$

such that for $\kappa = k^{\text{ctrl}}, \ldots, k^{\text{ctrl}} + N^{\text{p}}$ and for $r = 1, \ldots, n^{\text{rob}}$

$$\begin{cases} \boldsymbol{\tau} := \left(\tau_r^{\text{h}}(\kappa), \tau_r^{\text{v}}(\kappa)\right) \in E_r^{\text{feasible}}(\kappa) \\ \\ \boldsymbol{r}^{\text{rob}}(\kappa) \text{ is updated following the motion dynamics} \\ \text{explained in Section 5.3.3} \\ \\ \hat{a}_{\boldsymbol{\tau},r}\left(\kappa | k^{\text{ctrl}}\right) \text{ is obtained by aggregating } \hat{a}_{\boldsymbol{\tau},r}^{(\ell)}\left(\kappa | k^{\text{ctrl}}\right) \\ \text{computed by (5.18) for all rules using } \tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right) \\ \\ \tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right) \in \Theta \end{cases}$$

Here $\boldsymbol{r}^{\text{rob}}\left(k^{\text{ctrl}}\right)$ is a vector that includes the updated position $\left(\rho_r^{\text{h}}\left(k^{\text{rob}}\right), \rho_r^{\text{v}}\left(k^{\text{rob}}\right)\right)$ for all robots at current control time step k^{ctrl} and $\mathscr{M}\left(k^{\text{ctrl}}\right)$ includes all updated environment matrices. Notation $J^{\text{MPC}^*}\left(\boldsymbol{r}^{\text{rob}}\left(k^{\text{ctrl}}\right), \mathscr{M}\left(k^{\text{ctrl}}\right)\right)$ denotes the optimal value of the MPC objective function determined via the MPC optimisation problem for the given control time step. Parameter N^{p} is the prediction horizon. The optimisation variable $\tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right)$ includes all tuning parameters $\boldsymbol{\theta}_{r,\ell}\left(k^{\text{ctrl}}\right)$ used in (5.18) to determine the output attraction values per TSK-FLC rule. Function $f_r^{\text{cell}}(\cdot)$ represents a stage-wise objective. The hat symbol indicates the predicted value of a variable and the notation $(\cdot|k)$ implies that the prediction is made at time step k; we also have $\hat{a}_{\boldsymbol{\tau},r}\left(k^{\text{ctrl}} | k^{\text{ctrl}}\right) = \hat{a}_{\boldsymbol{\tau},r}\left(k^{\text{ctrl}}\right)$. The admissible set $\Theta \subseteq \mathbb{R}^{n^{\text{rob}}\left(n^{\text{ch}}+1\right)}$ ensures that tuning parameters remain within feasible bounds, safeguarding the system against unsafe or unstable behaviours.

5.3.5. STABILITY OF M2PFC

Here we provide insights about the stability considerations and boundedness of the overall closed-loop system under M2PFC. In the context of M2PFC, stability refers to the property that robot states (i.e., position and heading), control inputs (i.e., attraction values), and tuning parameters remain bounded under all operating conditions. Note that boundedness of the attraction values are crucial to prevent robots from repeatedly switching targets erratically and the MPC optimisation problem (5.19) from becoming ill-conditioned, leading to unreliable or oscillatory tuning.

Although the primary focus of M2PFC is to provide real-time adaptability and context-aware coordination of local controllers, its design inherently embeds mechanisms to avoid the specified types of instability.

More specifically, since robots operate within a grid $\mathscr{E}$ of finite dimensions (Assumption **A1**), yield bounded heading angles within $[-\pi, \pi]$ (see (5.14)–(5.16)), follow a bounded velocity (as per Assumption **A6**), and possess bounded travel times (as per (5.17)), all parameters related to robot dynamics are ensured to remain bounded.

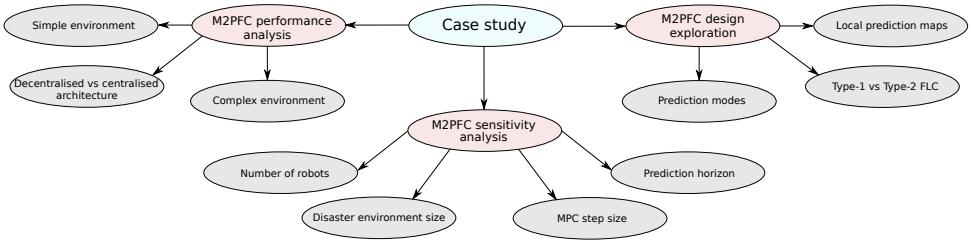


Figure 5.7: Roadmap for the categories of analyses and simulations performed in the case study.

Considering the task parameters, the travel times, scan times (as per (5.9)), victim probabilities (as per (5.2)), fire risk time — upper-bounded by fire propagation dynamics, and scan certainty (as per (5.1)) are bounded, thus all input quantities fed into the local FLC layer are inherently bounded by the problem formulation. Each local FLC controller operates with a rule base consisting of finite number of rules, each generating an output through the TSK inference mechanism given by (5.18), which possesses Bounded-Input-Bounded-Output (BIBO) stability. In other words, the output $a_{\boldsymbol{\tau},r}^{(\ell)}\left(k^{\text{ctrl}}\right)$ of each rule ℓ is an affine combination of bounded input quantities. Since the overall value of the output is obtained via convex aggregation of these rule outputs weighted by their associated normalised firing strengths adding up to 1, the mapping executed per control time step via each local FLC controller is BIBO stable.

Consequently, stability requires that the hierarchical tuning between the FLC and MPC layers does not induce unbounded or oscillatory behavior. For stability under MPC tuning, we know that this layer updates parameter vector $\tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right)$ within a restricted, admissible set Θ (see (5.19)). This bounded parameter space is designed to preserve the monotonicity and smoothness of the surfaces generated through fuzzy inference mechanisms, thereby preventing abrupt changes in these parameters that may destabilize the local FLC controllers. Furthermore, the MPC optimisation problem (see (5.19)) inherently respects constraints on robot dynamics, hence ensuring that each tuning step maintains the feasibility and stability of the underlying multi-robot system.

In summary, the hybrid structure of M2PFC may be viewed as a hierarchical system, where the fast, low-level FLC loop ensures immediate bounded responses to the environment, while the slower, supervisory MPC layer introduces gradual parameter adjustments that converge towards optimal, coordinated behavior. The time-scale separation between the fast-operating FLC controllers and slower MPC layer further ensures that tuning occurs gradually. This separation of time scales aligns with classical stability results for hierarchical and supervisory control frameworks, where slow tuning of bounded parameters preserves the stability of the underlying closed-loop dynamics [327], [328].

5.4. CASE STUDY

This section presents an extensive case study to evaluate the effectiveness, adaptability, and scalability of the proposed M2PFC architecture within a simulated SaR multi-robot environment. The performance of M2PFC — under both centralised and decentralised supervisory architectures — is compared against two alternative control approaches:

centralised and decentralised MPC-only system, and decentralised pre-tuned FLC controller. These comparisons aim to highlight the added value of the hybrid design offered by M2PFC, as well as its hierarchical tuning mechanism.

The case study consists of three main categories of analyses, including:

1. Performance evaluation of M2PFC
2. Sensitivity analysis of M2PFC
3. Design space exploration for M2PFC configuration

Each category is further divided into sub-experiments, as outlined in the roadmap in Figure 5.7.

5.4.1. CASE STUDY ASSUMPTIONS

In addition to the assumptions outlined in Section 5.3.1, the following assumptions, indexed by CS-A (Case Study Assumption), hold throughout the case study:

CS-A1 The 2D environment grid consists of identical cells.

CS-A2 All sampling times, at the global, local, control, and tuning levels have equal durations, and their corresponding time steps are synchronised. Only in some sensitivity analyses run in this case study, the control and tuning sampling times, while equal, may differ from other sampling times.

CS-A3 Wind velocity and direction remain constant across the entire environment, so do the elements of matrices $M^{\text{velocity}}(k)$ and $M^{\text{direction}}(k)$.

CS-A4 Unless specified otherwise, the wind velocity denoted by $m^{\text{velocity}}_{(i,j)}(k)$ is always less than 1 m/s, restricting fire spread to dotted neighbouring cells shown in Figure 5.4.

CS-A5 The value of threshold ζ in (5.6) is set to zero, meaning that a cell ignites immediately once the fire reaches it.

CS-A6 For computational efficiency, robots operate using coarsened environment matrices, where the environment is divided into larger, aggregated cells.

CS-A7 Each robot maintains one additional parameter representing its location within the coarsened cell, that is used to determine the travel time between non-coarsened cells.

CS-A8 The FLC controllers employ a type-1 TSK fuzzy inference system [329].

CS-A9 The supervisory MPC layer is invoked periodically to update the tuning parameters $\tilde{\boldsymbol{\theta}}\left(k^{\text{ctrl}}\right)$ of the FLC controllers.

5.4.2. COMPARISON APPROACHES AND MECHANISMS

To assess the effectiveness and adaptability of the M2PFC framework, we compare it against alternative control strategies and architectural setups commonly used in multi-agent robotic systems. This section outlines the different control approaches/architectures used in the comparison study.

Note that based on Assumption **CS-A2**, in the case study we simply use the notation k to represent the local, simulation, control, and tuning time steps.

MPC-ONLY SYSTEM

In the MPC-only system, robots are assigned a queue of target cells to visit and scan. The key difference in the optimisation problem compared to (5.19) is that, in this case, there is no mapping from attraction values to the objective through a parameterised function to be tuned. Instead, the optimisation variables of MPC are the cell coordinates planned for robots. The design details for the MPC-only system used in the case study are given in Sections 5.4.4 and 5.4.6.

Algorithm 5 in Appendix 5.8 illustrates how robots select and execute actions based on the queue of cells generated by MPC. While this queue-based scheduling allows robots to anticipate future tasks, it lacks the real-time efficiency of FLC and the adaptability offered by the M2PFC framework.

PRE-TUNED FLC SYSTEM

The pre-tuned FLC system involves decentralised FLC controllers with fixed, pre-set parameters. Unlike M2PFC, this approach does not include a supervisory controller for dynamic tuning of local controllers and relies instead on static control rules.

The details of the design of pre-tuned FLC controllers are given in Sections 5.4.5 and 5.4.6, while Algorithm 6 in Appendix 5.8 illustrates how robots select and execute actions with pre-tuned FLC controllers.

CENTRALISED AND DECENTRALISED ARCHITECTURES

For comparison purposes, both centralised and decentralised architectures are implemented for the M2PFC framework and for the MPC-only system. This setup allows evaluation of the impact of coordination strategies on performance and scalability.

In the centralised implementation, a single MPC system is responsible for tuning all local controllers in the M2PFC framework or for directly controlling the robots in the MPC-only system.

In contrast, the decentralised implementation assigns each local controller its own dedicated supervisory MPC system in the M2PFC framework, and each robot an independent local MPC controller. These MPC systems operate in a fully decentralised manner, meaning no information is exchanged among them.

5.4.3. COARSENING OF THE ENVIRONMENT MATRICES

In accordance with Assumption **CS-A6**, the environment matrices are coarsened using a coarsening factor ξ in order to improve computational efficiency in real-time decision making. Specifically, in order to reduce the number of variables involved in the prediction and optimisation steps of the M2PFC framework and the MPC-only system, the

environment parameters sensed by robots are consolidated from a $\xi \times \xi$ block of cells in the environment model into a single cell in the internal measurement model of the robot. Whenever $\xi = 1$, the measurement model of robots corresponds exactly to the environment model with a one-to-one mapping of cells.

In line with this, a generic environment matrix M, representing any of the matrices defined in Section 5.3.2 for simulating the environment, is coarsened for decision making via:

$$m^{\epsilon,\mathrm{c}}_{(i,j)}(k) = \frac{1}{\xi^2} \sum_{p=1}^{\xi} \sum_{q=1}^{\xi} m^{\epsilon}_{(\xi(i-1)+p\,,\,\xi(j-1)+q)}(k) \tag{5.20}$$

where $m^{\epsilon,\mathrm{c}}_{(i,j)}(k)$ denotes an element of the coarsened matrix, with $\epsilon \in \{\text{scan, victim, debris, fire}\}$. The same coarsening method is applied during the decision making to the structure matrix, which, however, is time-invariant. Moreover, ξ is the coarsening factor determining the resolution reduction for computationally efficient decision making, such that the original environment grid of dimensions $n^{\mathrm{h}} \times n^{\mathrm{v}}$ is reduced to a coarsened grid of $\left(n^{\mathrm{h}}/\xi\right) \times \left(n^{\mathrm{v}}/\xi\right)$ cells.

5

Remark 11. *Based on Assumption **CS-A3** the matrices defining the state of wind in the region, as outlined in Section 5.3.2, remain constant.*

5.4.4. MPC-BASED CONTROLLERS

This section presents the design and implementation of MPC-based strategies used in the supervisory layer of the M2PFC framework and within the MPC-only system. These controllers coordinate robot actions over a prediction horizon of size N^{p} by solving an optimisation problem that incorporates environmental forecasts and task priorities. The optimisation problem is a nonlinear constrained optimisation, solved with pattern search or genetic algorithm solvers (see also Section 5.4.6). A central component of these controllers is the objective function, which is explained next.

OPTIMISATION OBJECTIVE FUNCTION

In the case study, the MPC layer uses a tailored objective function based on (5.19) that has been designed to steer robots towards high-priority areas with larger attraction, while considering fire hazards. This objective function is formulated via:

$$J^{\mathrm{MPC}}(k) = \sum_{\kappa=k}^{k+N^{\mathrm{p}}} \sum_{r=1}^{n^{\mathrm{rob}}} J^{\mathrm{cell}}_{\boldsymbol{\tau},r}(\kappa|k) \tag{5.21}$$

where the cell-wise objective function is formulated via:

$$J^{\mathrm{cell}}_{\boldsymbol{\tau},r}(\kappa|k) = \hat{m}^{\mathrm{victim}}_{\boldsymbol{\tau}}(\kappa|k)\, \hat{m}^{\mathrm{scan}}_{\boldsymbol{\tau}}(\kappa|k) \cdot \left(w_1 - w_2 \hat{m}^{\mathrm{hazard}}_{\boldsymbol{\tau}}(\kappa|k)\right) \tag{5.22}$$

In (5.22), $\boldsymbol{\tau} = \left(\tau^{\mathrm{h}}_r(\kappa), \tau^{\mathrm{v}}_r(\kappa)\right)$ is the target cell assigned to robot r at time step κ. The coefficients w_1 and w_2 balance the trade-off between scanning high-priority cells and avoiding fire-prone regions.

The term $\hat{m}^{\mathrm{hazard}}_{\boldsymbol{\tau}}(\kappa|k)$ denotes the predicted value of a dynamic variable derived from the *hazard matrix*, $M^{\mathrm{hazard}}(k)$. At time step k, $M^{\mathrm{hazard}}(k)$ encapsulates the combined effect of the relative proximity of each cell that is burning or igniting and the

wind direction, thereby capturing both the potential of directional spread and proximity-based hazard. Its elements are normalised to $[0,1]$ based on the map dimensions and are updated following Algorithm 7 in Appendix 5.8, using two components: hazard direction and hazard distance.

The hazard direction matrix $M^{\text{hazard,dir}}(k)$ has elements given by:

$$m_{(i,j)}^{\text{hazard,dir}}(k) = \exp\Bigg(m_{(i,j)}^{\text{velocity}}(k)\Big(c_1^{\text{hazard}} + c_2^{\text{hazard}}\Big(\cos\big(m_{(i,j)}^{\text{direction}}(k) - \theta_{(i,j)}^{\text{hazard}}(k)\big) - 1\Big)\Big)\Bigg) \tag{5.23}$$

where c_1^{hazard} and c_2^{hazard} are constants, and $\theta_{(i,j)}^{\text{hazard}}(k)$ is the angle between cell (i,j) and the closest cell $(l(k),q(k))$ which is burning at time step k, defined as:

$$\theta_{(i,j)}^{\text{hazard}}(k) = \arctan\left(\frac{j-l(k)}{i-q(k)}\right) \tag{5.24}$$

The elements of the hazard distance matrix $M^{\text{hazard,dist}}(k)$, under Assumption **CS-A1**, are given by:

$$m_{(i,j)}^{\text{hazard,dist}}(k) = 1 - \frac{\sqrt{(i-l(k))^2+(j-q(k))^2}}{\sqrt{\left(n^{\text{h}}\right)^2+(n^{\text{v}})^2}} \tag{5.25}$$

Note that when the cells are coarsened for efficient decision making, the hazard matrix is estimated according to (5.20).

In summary, the objective function given by (5.21) and (5.22) captures a key trade-off in SaR: prioritizing likely victim locations, while avoiding areas at risk due to fire spread. To compute this objective, future values of victim priority, scan, and hazard matrices should be predicted within the prediction window. How to do so is explained next.

PREDICTION MODES FOR MPC-BASED LAYERS

Prediction modes represent two approaches for estimating future environmental states within the predictive controllers, i.e., the supervisory layer of the M2PFC framework and the MPC-only system.

Two distinct prediction modes are employed to forecast future states of the environment: the *probability threshold* mode and the *exact* mode. These modes are incorporated into the comparisons executed for the case study to assess the overall performance and robustness of the control systems. The probability threshold prediction mode is used as the default prediction mechanism in all simulations, based on the practical assumption that future states can, in general, not be known with certainty. However, selected simulations also explore the use of the exact prediction mode, which serves as an idealised baseline for comparative purposes.

In the probability threshold prediction mode, it is assumed that the most probable state transition occurs at all time steps. For a system with state vector $\boldsymbol{x}(k)$ and a state transition function $\mathscr{F}(\cdot)$, the next predicted state is given by:

$$\hat{\boldsymbol{x}}(k+1|k) = \mathscr{F}\left(\boldsymbol{x}(k)\right) \tag{5.26}$$

where $\mathscr{F}(\cdot)$ is a deterministic function that maps the current state $\boldsymbol{x}(k)$ to the most likely successor state $\hat{\boldsymbol{x}}(k+1|k)$.

In contrast, the exact prediction mode assumes perfect knowledge of future states, thereby removing uncertainty from the prediction process. In this mode, the prediction function $\mathscr{F}(\cdot)$ is assumed to map the current state $\boldsymbol{x}(k)$ precisely to the state that will actually take place, i.e., it perfectly reflects the actual system dynamics. Although this level of foresight is not practically achievable, the exact prediction mode serves as an idealised benchmark for evaluating the performance of the probabilistic approach.

In our case studies, within the exact prediction mode, predictive controllers are provided with the true future states of cells, as dictated by the simulations. In contrast, under the probability threshold prediction mode, these future cell states should be estimated within the prediction window. How to do so forms the focus of the next section.

ESTIMATING ENVIRONMENTAL MATRICES WITHIN THE PROBABILITY THRESHOLD MODE

This section describes how key environment matrices are estimated within the case study simulations under the probability threshold prediction mode. These estimations inform both the local decision making of individual robots and the global reasoning performed by the supervisory controllers.

Prediction of the Victim Probability Matrix: For predictive controllers, the elements of the victim probability matrix estimated at time step k for any future time step κ within the prediction window, when cell (i,j) is planned to be visited, are estimated by:

$$\hat{m}^{\text{victim}}_{(i,j)}(\kappa|k) = \begin{cases} m^{\text{victim}}_{(i,j)}(k) \\ \quad \text{if cell } (i,j) \text{ has ever been scanned} \\ c^{\text{pop}}\ell^{\text{h}}_{(i,j)}\ell^{\text{v}}_{(i,j)}\left(1-o_{(i,j)}(k)\right) \\ \quad \text{if cell } (i,j) \text{ has never been scanned} \end{cases} \tag{5.27}$$

where c^{pop} is the average population density of the disaster environment, while $\ell^{\text{h}}_{i,j}$ and $\ell^{\text{v}}_{i,j}$ denote spatial parameters of the cell. The value of $m^{\text{victim}}_{(i,j)}(k)$ used when the cell has already been scanned is obtained based on (5.2). Moreover, $o_{(i,j)}(k) \in [0,1]$ is the most recent value for the perceived proportion of cell (i,j) occupied by debris, as defined in (5.3).

Prediction of the Hazard Matrix: The elements of the hazard matrix at each future time step $\kappa > k$ within the prediction horizon are predicted at time step k by:

$$\hat{m}^{\text{hazard}}_{(i,j)}(\kappa|k) = f^{\text{hazard}}\left(\hat{M}^{\text{fire}}(\kappa|k)\right) \tag{5.28}$$

where $\hat{M}^{\text{fire}}(\kappa|k)$ denotes the matrix of predicted fire states $\hat{m}^{\text{fire}}_{(l,q)}(\kappa|k)$ at future time step κ, as predicted at time step k for all cells (l,q) within the simulation environment. The function $f^{\text{hazard}}(\cdot)$ implements the fire propagation mechanism as described by (5.23)–(5.25), following the procedure outlined in Algorithm 7 (see Appendix 5.8). Note that wind velocity and direction are required for computing the hazard influence in (5.23)–(5.25), where in accordance with Assumption **CS-A3**, these parameters are assumed to be constant and known throughout the simulations.

Table 5.2: Parameter values for FLC input membership functions illustrated in Figure 5.8.

Name	Type	Parameters
low	Triangular	[0, 0, 0.5]
medium	Triangular	[0, 0.5, 1]
high	Triangular	[0.5, 1, 1]

The fire state in cell (l,q) at time step $\kappa > k$ within the prediction horizon, is predicted at time step k via:

$$\hat{m}^{\text{fire}}_{(l,q)}(\kappa|k) = f^{\text{fire}}\left(\hat{m}^{\text{fire}}_{(\lambda,\gamma)}(\kappa|k), \hat{m}^{\text{fire}}_{(l,q)}(\kappa-1|k)\right) \tag{5.29}$$

where $f^{\text{fire}}(\cdot)$ is a function that executes (5.6), (λ,γ) is a neighbouring cell for cell (l,q), and $\hat{m}^{\text{fire}}_{(l,q)}(k|k) = m^{\text{fire}}_{(l,q)}(k)$.

5.4.5. LOCAL FLC CONTROLLERS

The time efficiency, priority, fire risk time, and scan certainty of a cell serve as inputs to the FLC systems, as described in Section 5.3.4. In the case study, these values should be adapted for decision making in accordance with the coarsened cell structure.

COMPUTING FLC INPUTS

For priority and scan certainty of a cell, the coarsened values are computed by applying the aggregation method in (5.20) to the corresponding fine-grid values.

For response efficiency, the overall travel time is computed according to (5.17), assuming that the robot moves with its maximum feasible velocity as per Assumption **A6** and the scan time (when the assigned task is "scan") is computed according to (5.9).

For fire risk time, as explained in Section 5.3.4, Algorithm 4 given in Appendix 5.8 is employed. This algorithm takes the coarsened fire matrix, computed via (5.29), and generates the fire risk time.

The values of the first three inputs are normalised to the range $[0,1]$ prior to being passed to the associated FLC controller, while the fourth is not, since using a larger range for this input with active fires has been found to be more effective.

After the cell attractions are computed by the local FLC controllers, each robot selects the target cells it should travel to and scan. Due to cell coarsening, it is necessary to track the remaining travel or scan time after every sampling time. This adjustment is required is because coarsened cells are larger, and robots may not be able to fully traverse or scan such cells within a single sampling time. The procedure for action selection and execution per robot is detailed in Algorithm 6 in Appendix 5.8.

FLC MEMBERSHIP FUNCTIONS

For local controllers, as stated in assumption **CS-A8**, TSK-FLC controllers are deployed. In the FLC controllers, all input membership functions are defined as triangular functions, uniformly distributed across the range of each input variable. The choice of triangular membership functions is motivated by the simplicity of design, the computational

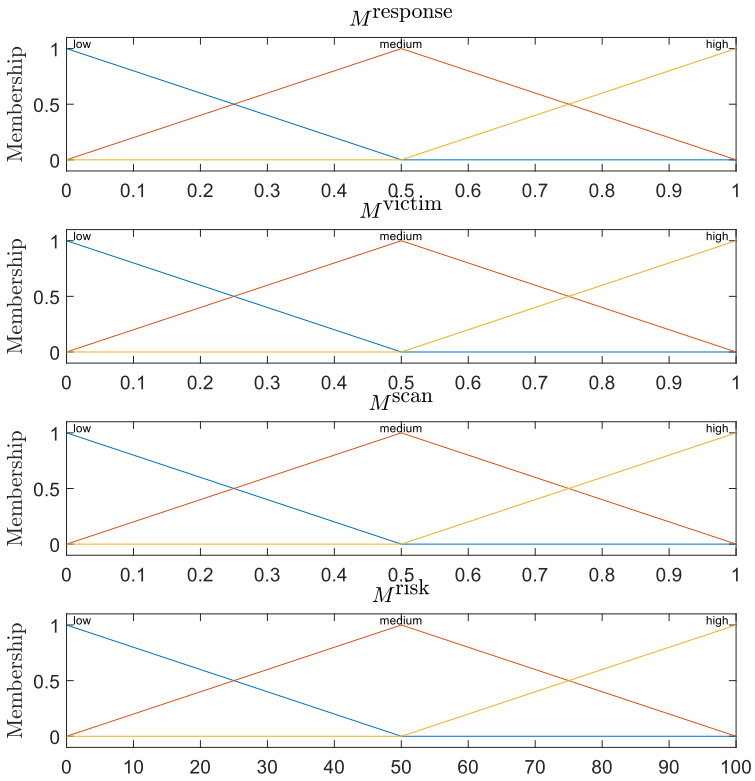


Figure 5.8: Input membership functions for local FLC systems. The first three inputs are normalised, while the fourth is not, since it has been found that using a larger range for the fourth input with active fires is more effective.

Table 5.3: Default parameter values for FLC output membership functions illustrated in Figure 5.9. As illustrated in the figure, the first four values are the coefficients of a particular input of the four inputs, respectively, while the fifth is a constant.

Name	**Type**	**Parameters (coefficients)**
low	linear	$-1,1,-1,-1,0$
medium	linear	$-1,1,-1,-1,0.5$
high	linear	$-1,1,-1,-1,1$

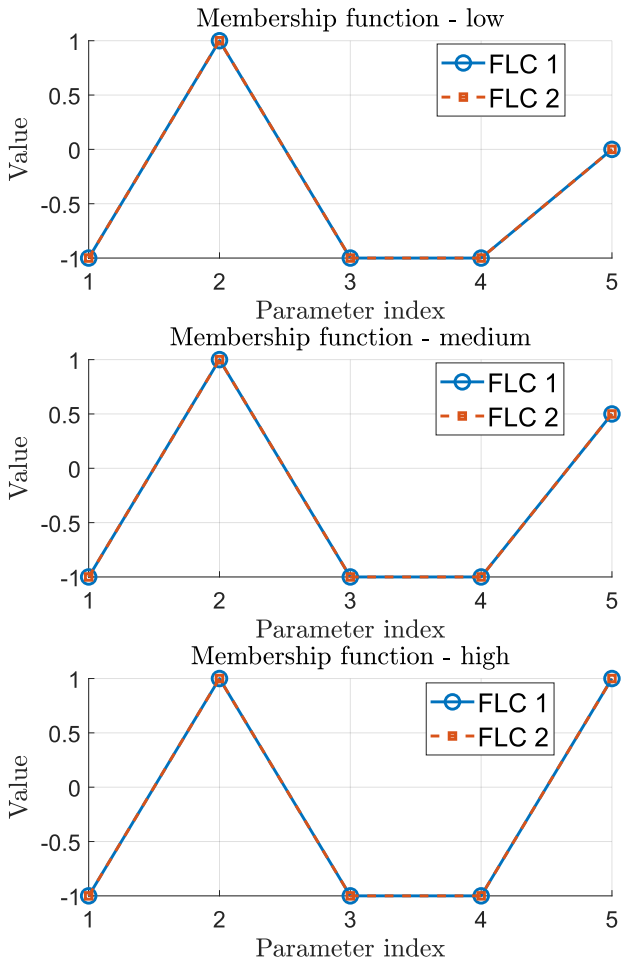


Figure 5.9: Parallel coordinate plot of default output membership functions for local FLC systems.

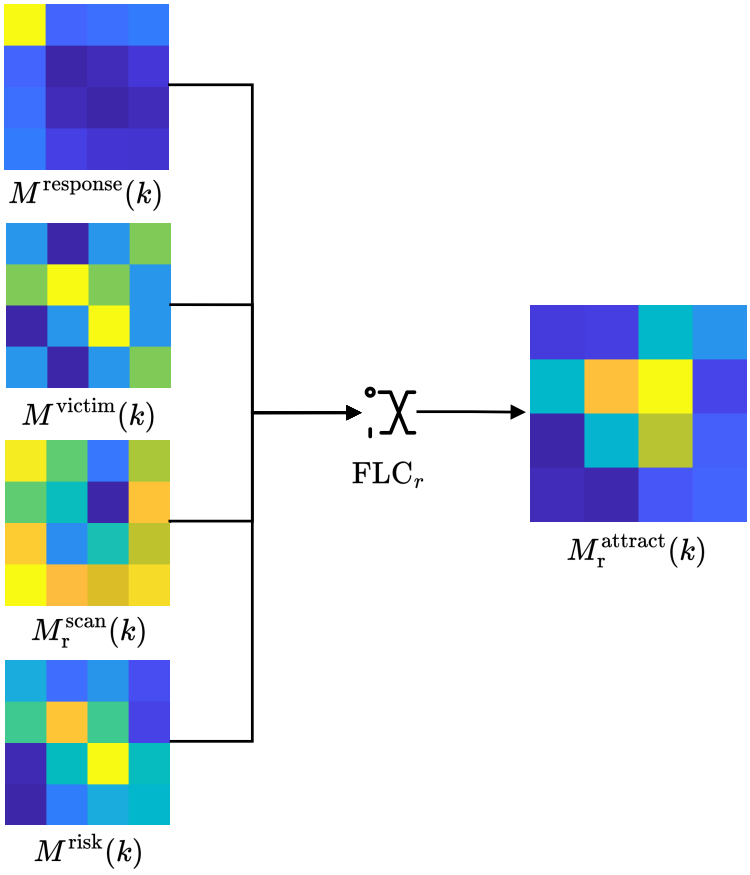


Figure 5.10: Simplified example illustrating the inputs and outputs of an FLC system: The colours from blue to yellow are based on a heat-map, representing the value of each variable for the given SaR scenario within the grid.

Table 5.4: General simulation parameters used in the case study.

Parameter	Value	Equation(s) using the parameter
Simulation sampling time (T)	15 time units	(5.5) and (5.6) and also used in Algorithms 5 and 6
Cell coarsening factor (ξ)	5	(5.20)
Objective function weight 1 (w_1)	1	(5.22)
Objective function weight 2 (w_2)	1	(5.22)

efficiency, and mostly the common usage in robotic control systems, such as for speed control [330]. These initial configurations are listed in Table 5.2, where the parameters represent the vertices of each membership function, corresponding to Figure 5.9, therefore the first four values are the coefficients of a particular input of the four inputs, respectively, while the fifth is a constant. The complete set of input membership functions is illustrated in Figure 5.8.

In contrast, the output expressions for these TSK-FLC controllers are chosen to be an affine combination of the inputs and a constant, offering a piecewise approximation that facilitates gradient-based tuning. These are initialised as shown in Table 5.3 and are visually represented using a parallel coordinate plot in Figure 5.9.

To illustrate the operation of the FLC systems, Figure 5.10 provides an example scenario for an arbitrary robot r. The disaster environment in this example is represented as a 4×4 grid. The input matrix *fire risk time*, $M^{\text{risk}}(k)$, indicates that cells in burning fire state are concentrated in the centre of the grid. The current position of the robot within cell $(1,1)$ is highlighted in the cell response efficiency matrix $M_r^{\text{response}}(k)$, since it has the colour corresponding to the lower value.

The output matrix $M_r^{\text{attract}}(k)$ reveals that the robot prioritises cell $(3,2)$ as its next target. Interestingly, this decision may appear as counter-intuitive, since cell $(3,2)$ already exhibits a high scan certainty. Such behavior reflects the flexible, adaptive decision making by the M2PFC framework. Based on the prevailing context, M2PFC may determine that the robot should contribute to increasing the weighting of the scan certainty, while reducing the emphasis on the response time — potentially leading to the selection of cell $(3,2)$ as the next target of the robot.

5.4.6. SETUP

This section outlines the setup of the simulations, hardware specifications, parameter configurations, and the performance metrics used in the case study.

Additionally, to provide further insight into the fire propagation model and robot motion under varying wind conditions, we have included illustrative simulations in Appendix 5.9.

The case study is implemented in MATLAB, with all simulations executed on a machine equipped with an AMD Ryzen 5 3500U CPU. Each simulation is configured via a dedicated set of scripts that define all relevant parameters; these scripts are publicly available in the 4TU repository [331] and the associated GitHub repository [332].

For a better match with the meaning of the objective, i.e., a measure of the risk to

Table 5.5: Environment configuration parameters used in the case study.

Parameter	Value	Equation(s) using the parameter
Environment cell lengths ($\ell^{\mathrm{h}}_{(i,j)}$ and $\ell^{\mathrm{v}}_{(i,j)}$)	10 space units	(5.9) and (5.27)
Ignition time (t^{spread})	120 time units	(5.5) and (5.6)
Burnout time ($t^{\text{burnt-out}}$)	600 time units	(5.5) and (5.6)
Max number of victims per search map cell (n^{victim})	5	(5.2)
Fire spread constant 1 (α_1)	0.2	(5.4)
Fire spread constant 2 (α_2)	0.08	(5.4)
Fire spread constant 3 (α_3)	0	(5.4)
Fire spread constant 4 (α_4)	0.1	(5.4)
Wind model constant 1 (c_1^{hazard})	0.1	(5.23)
Wind model constant 2 (c_2^{hazard})	0.1	(5.23)

Table 5.6: Robots configuration parameters used in the case study.

Parameter	Value	Equation(s) using the parameter
Maximum velocity ($v_r(k)$ for all time steps k)	$5\,\mathrm{m\,s^{-1}}$	(5.12)
Scan time per square meter (t_r^{scan})	$0.01\,\mathrm{s\,m^{-2}}$	(5.9)
Sensor accuracy (η)	0.9 [-]	(5.1)
Scan certainty loss per time step (σ)	0.01 [-]	(5.1)

the victims, we have reformatted the objective function given by (5.21) by using a minus sign, so that a lower value is considered better, therefore all optimisations become minimisations, since the purpose is to minimise the risk.

Unless stated otherwise, a standard set of parameters is applied across all simulations, where these values are given in Table 5.4. The primary exceptions are the number of robots, the size of the disaster environment, and the MPC sampling time and prediction horizon, which vary in the sensitivity analyses performed in Section 5.5.2.

ENVIRONMENT CONFIGURATION

The standard configuration of the environment model is defined by a set of parameters listed in Table 5.5. The wind velocity $m^{\text{velocity}}_{(i,j)}(k)$ is set to 1 $\frac{\mathrm{m}}{\mathrm{s}}$ in scenarios specified by "*with-wind*", and to 0 in scenarios specified by "*no-wind*".

ROBOTS CONFIGURATION

The standard configuration of the robot model is defined by the parameters listed in Table 5.6.

All robots are initially assigned task of "scan". Under the M2PFC framework, the initial target cell per robot is set to its current position, i.e., we have $\left(\tau_r^{\mathrm{h}}(k), \tau_r^{\mathrm{v}}(k)\right) =$

$\left(\rho_r^{\mathrm{h}}(k), \rho_r^{\mathrm{v}}(k)\right)$. In contrast, for the MPC-only system, since at the start of the simulations only one cell (with the robot location) is available as target, the initial target cells are set to the current position of the robot, but subsequent entries in the target cell sequence are initialised to one, that indicates a neutral or default placeholder that will be overwritten at the subsequent iterations.

LOCAL MAPPING CONFIGURATION

With regard to the matrices that robots build as local environmental maps (see Section 5.3.2), the debris occupancy matrix $M^{\text{debris}}(k)$ is initialised with all elements set to 0.5, while the structure matrix M^{struct} is initialised with all elements set to 1. For the victim probability matrix $M^{\text{victim}}(k)$, the initialisation is random, where the distribution of victims over the environment varies with the simulation seed.

For "no-wind" conditions, the fire state matrix $M^{\text{fire}}(k)$ is initialised with all elements set to 1 to ensure that there is no fire spread during the simulation.

In contrast, for "with-wind" conditions, the fire state matrix $M^{\text{fire}}(k)$ is initialised with a uniform distribution of flammable cells, and a subset of cells is assigned a burning fire state 3 (the number of such cells is selected randomly from the range 2-4).

All robots are initially positioned in a row of adjacent cells located in the bottom-left corner of the disaster environment.

Three distinct simulation scenarios are considered in the case study, namely *basic no-wind*, *basic with-wind*, and *complex with-wind*. The primary differences among these simulations lie in the absence or presence of wind (thus fire spread), and the probability distribution of matrices M^{struct} and $M^{\text{debris}}(k)$ that is closer to or farther from uniform. Further details regarding the design of these simulations are provided in Appendix 5.10.

CONFIGURATION OF DIFFERENT CONTROL SYSTEMS

This section details the configuration of control approaches considered for comparison in this case study, as explained earlier in Section 5.4.2.

In real-time optimisation-based control applications, a key constraint is the time available to compute a solution for the optimisation problem before the next control time step, as dictated by the sampling time. Accordingly, a finite iteration budget is imposed on the optimisation problems to ensure that each optimisation process completes within the allowed computation time.

Configuration of M2PFC: The M2PFC framework employs a classical *pattern search* algorithm for optimisation. This method is selected due to its robustness in solving non-smooth, nonlinear, and discontinuous optimisation problems without requiring gradient information. These characteristics make it particularly suitable for the probabilistic and complex nature of the environment model in the simulations.

Key parameters used in the optimisation toolbox of MATLAB for the M2PFC framework include: maximum number of function evaluations 100, maximum number of iterations 100, optimisation lower bounds all -1, and optimisation upper bounds all 1.

For local FLC controllers, the configuration outlined in Section 5.3.4 is deployed.

Configuration of MPC-only System: The MPC-only system uses a *genetic algorithm* for optimisation. This algorithm is preferred over *pattern search* for this controller be-

cause of its better performance in the cases of complex (non-linear, discontinuous) discrete optimisation problems.

Key parameters used for the optimisation problem include: maximum number of function evaluations 100, population size 100, optimisation lower bounds [1, 1], whereas upper bounds are set to the coarsened environment dimensions.

Configuration of Pre-tuned FLC System: The pre-tuned FLC system is configured as described in Sections 5.3.4 and 5.4.5. Its parameters remain fixed throughout the simulations, offering a baseline for evaluating the performance of simple, local rule-based controllers in the absence of global re-optimisation.

PERFORMANCE METRICS

Metrics used to evaluate the performance of various controllers in this case study are the global objective function $J^{\text{MPC}}(k)$, defined by (5.21), and the optimisation time $t^{\text{opt}}(k)$ required to solve the control problem per time step k.

Given the probabilistic nature of both the environment model and the optimisation algorithms, the performance of the multi-robot system may vary significantly across simulation runs. To capture this variability properly, we report the mean values and confidence intervals of performance metrics over multiple simulations. Considering n^{sim} total number of simulation runs, the mean of the objective function is computed by:

$$\bar{J}^{\text{MPC}}(k) = \frac{1}{n^{\text{sim}}} \sum_{s=1}^{n^{\text{sim}}} J_s^{\text{MPC}}(k) \tag{5.30}$$

Moreover, the mean optimisation time at each control time step is given by:

$$\bar{t}^{\text{opt}}(k) = \frac{1}{n^{\text{sim}}} \sum_{s=1}^{n^{\text{sim}}} t_s^{\text{opt}}(k) \tag{5.31}$$

To represent the uncertainty in these estimations, we calculate 95% confidence intervals using the standard error of the mean (we show the corresponding function by SEM$(\cdot)$). These intervals are represented via $\left(\bar{J}^{\text{MPC}}_{0.025}(k), \bar{J}^{\text{MPC}}_{0.975}(k)\right)$ for the objective function, and via $\left(\bar{t}^{\text{opt}}_{0.025}(k), \bar{t}^{\text{opt}}_{0.975}(k)\right)$ for the optimisation time. The bounds for these intervals at time step k are computed via $\bar{J}^{\text{MPC}}(k) \pm 1.96\,\text{SEM}\left(\bar{J}^{\text{MPC}}(k)\right)$ for the objective function, and via $\bar{t}^{\text{opt}}(k) \pm 1.96\,\text{SEM}\left(\bar{t}^{\text{opt}}(k)\right)$ for the optimisation time.

To ensure fair comparisons among different controllers, we define a fixed sequence of random seeds. Each controller uses the same seed per simulation run, guaranteeing that the stochastic components of the environment are consistent across all simulations.

STANDARD RESULTS FORMAT

Simulation results are visualised using curve plots for each controller and scenario. To ensure visual consistency, we define a standard curve and colour format, as shown in Figure 5.11. Results corresponding to the pre-tuned FLC system, the M2PFC framework, and the MPC-only system are shown, respectively, in green, orange, and purple. A solid curve represents an MPC-based controller embedded within a centralised architecture, while a dashed curve represents a decentralised implementation.

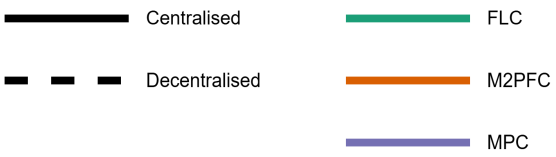


Figure 5.11: Curve styles used to show the simulation results.

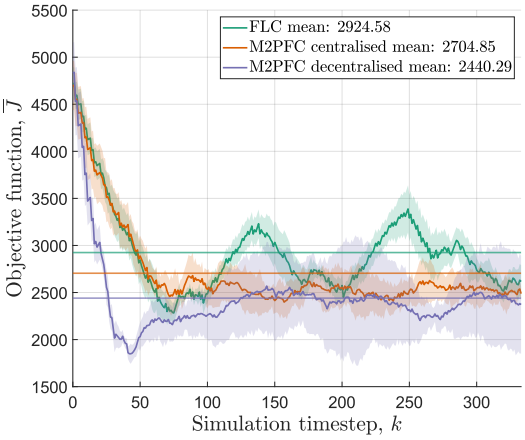


Figure 5.12: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a small static disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

5.5. RESULTS AND DISCUSSIONS

This section presents the results of the simulations in the case study, illustrated via plots that show the evolution of performance metrics over time for each controller type. The results are organised into three categories of analysis related to the M2PFC framework, including performance analysis, sensitivity analysis, and design exploration.

Following the presentation of results, a detailed discussion and interpretation of the observed patterns and outcomes for these results is provided.

5.5.1. RESULTS FOR M2PFC PERFORMANCE ANALYSIS

In the performance analysis, the M2PFC framework is assessed against MPC-only and pre-tuned FLC systems. Starting with simple simulation cases, we enable isolating the control performance from the modelling complexities. More complex simulations are introduced gradually and systematically.

Basic Scenarios: Details that describe this category of simulations are presented in Appendix 5.10.

Figure 5.12 presents the values of the mean objective function over time for centralised implementation of the M2PFC framework and the MPC-only system and decentralised implementation of the pre-tuned FLC system, for a two-robot system for a no-

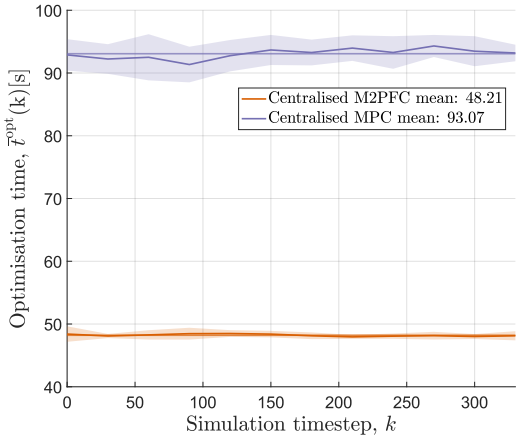


Figure 5.13: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a two-robot system in a small static disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

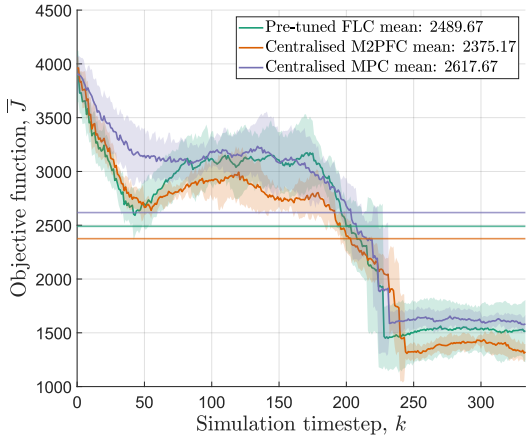


Figure 5.14: The values of the mean of the objective function $\overline{J}(k)$ over time for two-robot system in a small dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

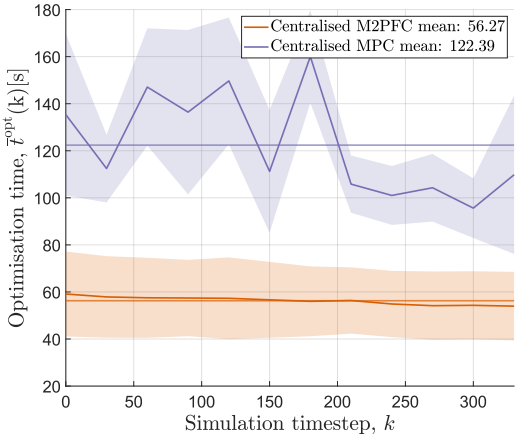


Figure 5.15: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ for a two-robot system in a small dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

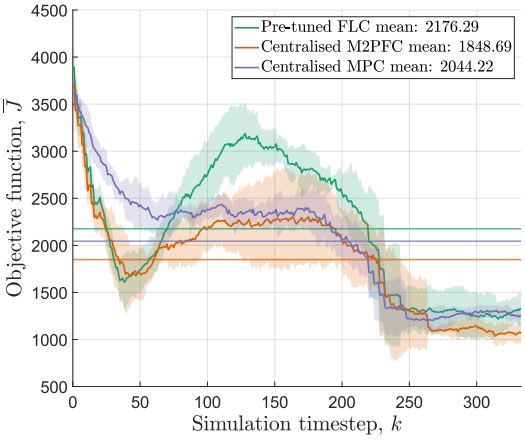


Figure 5.16: The values of the mean of the objective function $\overline{J}(k)$ for a four-robot system in a small dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

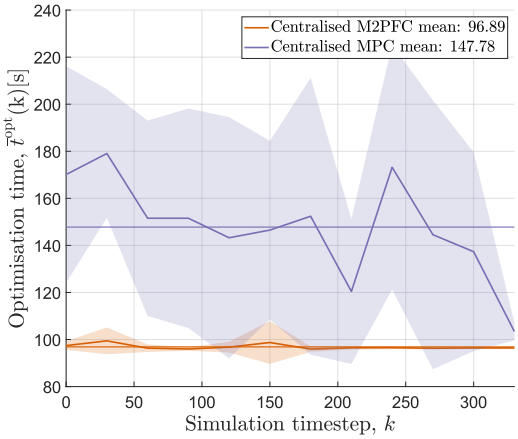


Figure 5.17: The values of the mean optimisation time $\bar{t}^{\text{opt}}(k)$ for a four-robot system in a small dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

wind condition within a grid environment of 40 × 40 identical cells (see Assumption **CS-A1**), across 5 simulation runs, each simulated over 5000 time units. Figure 5.13 shows the evolution of the corresponding mean optimisation time for both MPC-based controllers over time.

Figure 5.14 presents the evolution of the mean objective function over time for the centralised implementation of the M2PFC framework and the MPC-only system and the decentralised implementation of the pre-tuned FLC system for a two-robot system for a with-wind condition within a grid environment of 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units. Figure 5.15 shows the evolution of the corresponding mean optimisation time for both MPC-based controllers over time.

Finally, Figure 5.16 presents the results for the evolution of the mean objective function over time for similar environment, wind conditions, and controller implementation as before, but with the number of robots increased to $n^{\text{rob}} = 4$. Figure 5.17, likewise, shows the evolution of the corresponding mean optimisation time for both MPC-based controllers over time.

Decentralised vs Centralised M2PFC Architectures: Maintaining the same basic with-wind scenario, centralised and decentralised implementations of the M2PFC framework are compared next.

Figure 5.18 presents the evolution of the mean objective function over time for both centralised and decentralised architectures for a two-robot system and the basic with-wind conditions, across 5 simulation runs, each simulated over 5000 time units. Figure 5.19 shows the evolution of the corresponding mean optimisation time for both MPC-based controllers over time.

Figure 5.20 presents the results of the evolution of the mean objective function over time for the same scenario and implementation of controllers, where the number of robots is increased to $n^{\text{rob}} = 4$. Figure 5.21, likewise, shows the evolution of the corre-

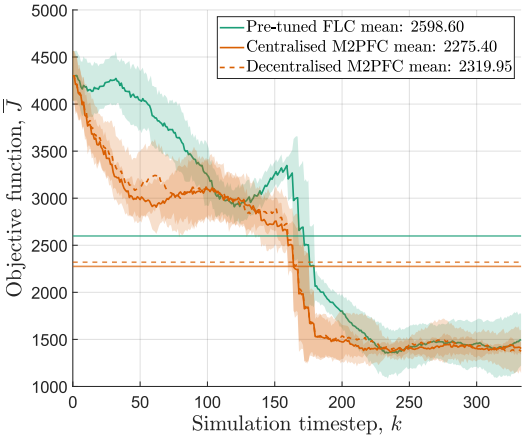


Figure 5.18: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a small dynamic disaster environment of size 40×40 cells with centralised vs decentralised M2PFC, across 5 simulation runs, each simulated over 5000 time units.

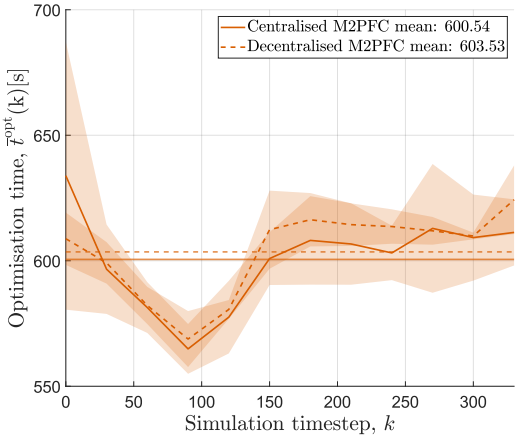


Figure 5.19: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a two-robot system in a small dynamic disaster environment of size 40×40 cells with centralised vs decentralised M2PFC, across 5 simulation runs, each simulated over 5000 time units.

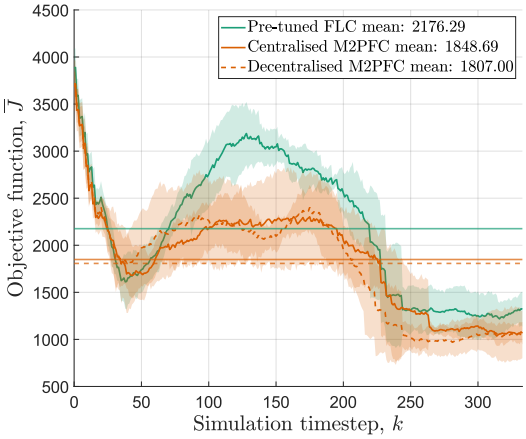


Figure 5.20: The values of the mean of the objective function $\overline{J}(k)$ over time for a four-robot system in a small dynamic disaster environment of size 40×40 cells with centralised vs decentralised M2PFC, across 5 simulation runs, each simulated over 5000 time units.

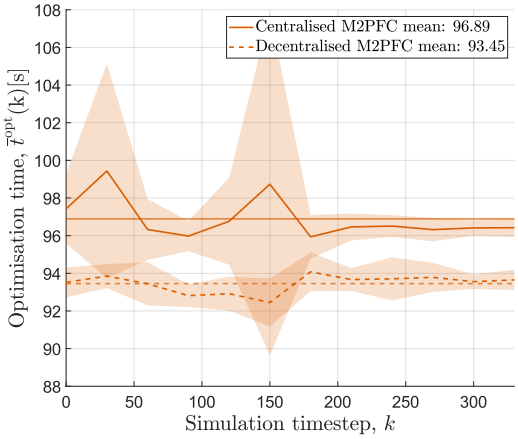


Figure 5.21: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a four-robot system in a small dynamic disaster environment of size 40×40 cells with centralised vs decentralised M2PFC, across 5 simulation runs, each simulated over 5000 time units.

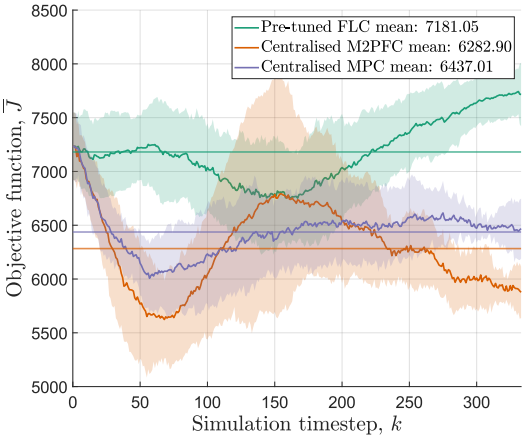


Figure 5.22: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a complex dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

sponding mean optimisation time for both MPC-based controllers over time.

Complex Scenarios: In the previous performance analysis simulations, the environment variables were simplified as much as possible to isolate controller performance in a controlled environment. In complex scenarios, which are all under with-wind conditions, we analyse controller performance under more complex conditions by introducing environmental parameters that more significantly involve the impact of wind and fire spread, as well as a non-uniform distribution of matrices M^{struct} and $M^{\text{debris}}(k)$. Further details about the design of these complex scenarios are presented in Appendix 5.10.

Figure 5.22 illustrates the evolution of the mean objective function over time for these complex scenarios, across 5 simulation runs, each simulated over 5000 time units. Figure 5.23 shows the evolution of the corresponding mean optimisation time for both MPC-based controllers over time.

5.5.2. RESULTS FOR M2PFC SENSITIVITY ANALYSIS

The objective of the sensitivity analysis is to investigate the performance of the supervisory MPC-based controller over a range of values for given design parameters.

The analysis is performed individually for four parameters — number of robots, size of the environment, sampling time of MPC, and prediction horizon of MPC — across a defined range of values, as given in Table 5.7.

Number of Robots: Figure 5.24 presents the values for the normalised mean objective function over time for decentralised and centralised implementations of the MPC-only system and the M2PFC framework — based on running 5 simulations, each for 5000 time steps — with the number of robots varying within the range $n^{\text{rob}} = \{2,3,4\}$. Varying the number of robots from 2 to 4 is enough to obtain information on the scalability of the multi-robot system, as explained in the subsequent discussion section (see Section 5.5.5). The values for the mean objective function are normalised against the results

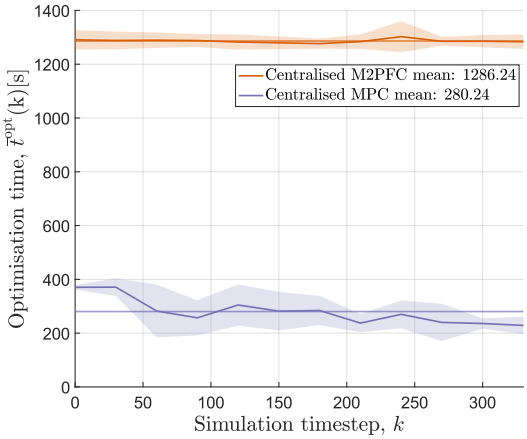


Figure 5.23: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a two-robot system in a complex dynamic disaster environment of size 40 × 40 cells, across 5 simulation runs, each simulated over 5000 time units.

5

Table 5.7: Sensitivity analysis parameters.

Parameter	**Range**
Number of robots (n^{rob})	2, 3, 4
Size of the environment (n^{h} and n^{v})	20, 40, 60
MPC sampling time (T^{ctrl})	30, 75, 225, 450, 675, 900 (given in time units)
Prediction horizon (N^{p})	30, 45, 60, 75 (given as number of time steps)

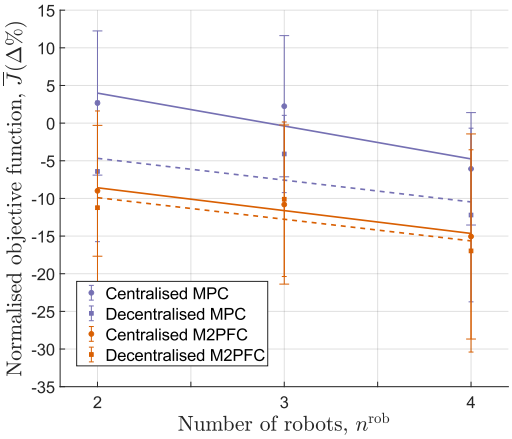


Figure 5.24: Sensitivity, represented for the normalised mean objective function $\overline{J}(k)$ over time, with respect to the number of robots, running 5 simulations per parameter value, each simulated over 5000 time units.

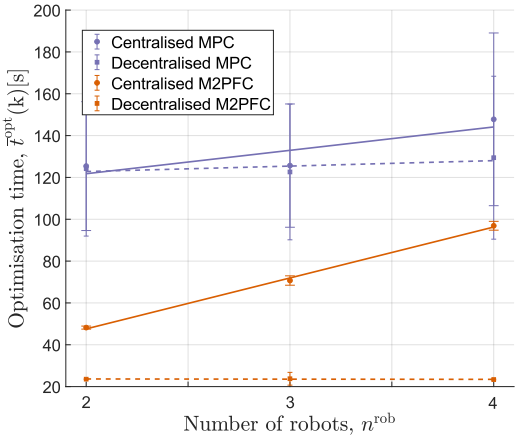


Figure 5.25: Sensitivity, represented for the mean optimisation time $\bar{t}^{\text{opt}}(k)$ over time, with respect to the number of robots, running 5 simulations per parameter value, each simulated over 5000 time units.

for the pre-tuned FLC system, given as the percentage difference from the value of the mean instantaneous objective function for the pre-tuned FLC system.

The individual data points for each set of simulations are denoted by coloured dots in Figure 5.24, following the same standard format explained in Section 5.4.6. Moreover, the 95% confidence intervals are represented by error bars. To differentiate between controller architectures in illustrations, results for the centralised implementation are displayed using circular points and wider error bar tips, while, for the decentralised implementation, results are displayed using square points and narrower error bar tips.

A first-order polynomial trend line has been fitted to the results of the sensitivity analysis per case, in order to approximately visualise the correlation with the number of robots. This is done in MATLAB, using the function `polyfit`, which performs the least square error minimisation for a polynomial of a chosen order. Note that the choice of the order of the polynomial depends on the number of data points (for orders larger than one see, for instance, Figure 5.28 for the sensitivity analysis concerning the MPC sampling time).

Finally, Figure 5.25 shows the values for the corresponding mean optimisation times over time.

Disaster Environment Size: Figure 5.26 presents the normalised values for the mean objective function over time, for centralised implementation of the MPC-only system and the M2PFC framework for varying number of cells in the disaster environment, across 5 simulation runs, each simulated over 5000 time units.

While changing the size of the environment, for all simulations only one ignition point is considered, whereas the amount of victims increases, and they are always distributed differently, because they are randomly located.

Figure 5.27 shows the values of the corresponding mean optimisation time over time.

MPC Sampling Time: The MPC sampling time, according to Assumption **CS-A9**,

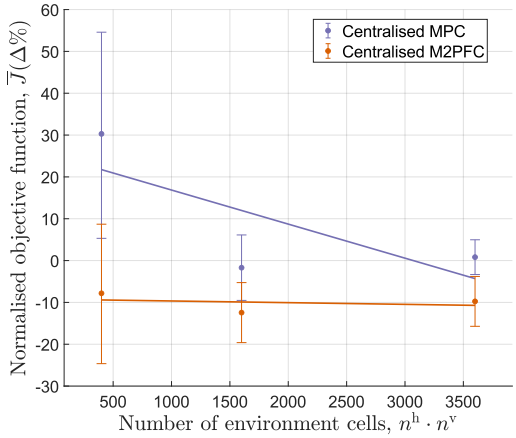


Figure 5.26: Sensitivity, represented for the normalised mean objective function $\overline{J}(k)$ over time, with respect to the environment size, running 5 simulations per parameter value, each simulated over 5000 time units.

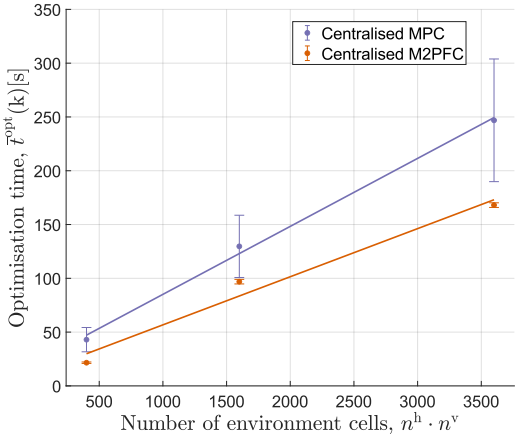


Figure 5.27: Sensitivity, represented for the mean optimisation time $\overline{t}^{\mathrm{opt}}(k)$ over time, with respect to the environment size, running 5 simulations per parameter value, each simulated over 5000 time units.

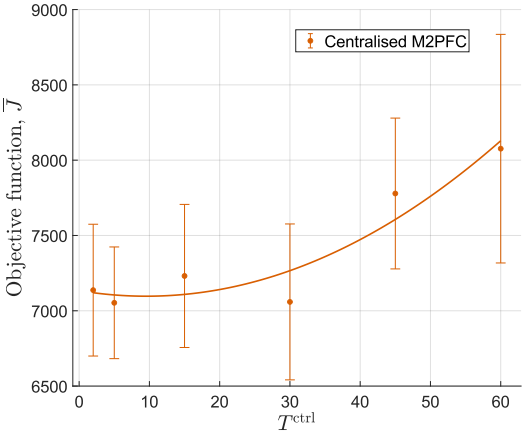


Figure 5.28: Sensitivity, represented for the normalised mean objective function $\overline{J}(k)$ over time, with respect to the MPC sampling time, running 5 simulations per parameter value, each simulated over 5000 time units.

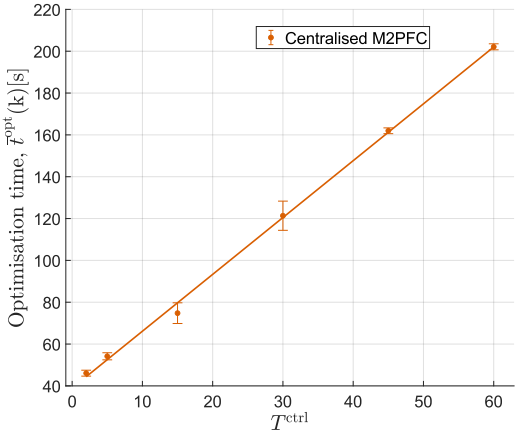


Figure 5.29: Sensitivity, represented for the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time, with respect to the MPC sampling time, running 5 simulations per parameter value, each simulated over 5000 time units.

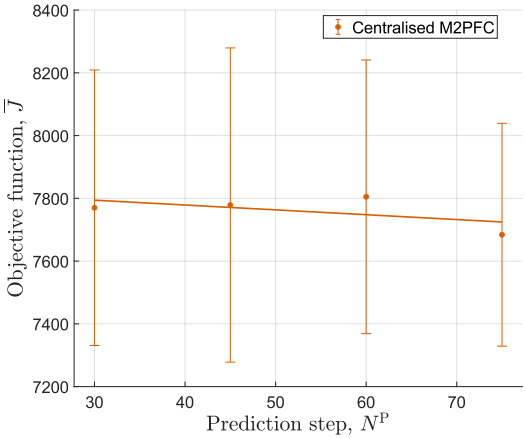


Figure 5.30: Sensitivity, represented for the normalised mean objective function $\overline{J}(k)$ over time, with respect to the MPC prediction horizon, running 5 simulations per parameter value, each simulated over 5000 time units.

determines when parameters of the FLC controllers are updated. In this section, we present the results for a sensitivity analysis when the MPC sampling time, T^{ctrl}, varies, while the prediction horizon is maintained as $N^{\mathrm{p}} = T^{\mathrm{ctrl}} + 15$ time units.

Figure 5.28 presents the mean objective function over time, across 5 simulation runs, each simulated over 5000 time units. These values have not been normalised, because the FLC has not this parameter (sampling time), therefore a normalisation is not done since otherwise it would be done against the same value for all the simulations.

Figure 5.29 shows the mean optimisation times over time for this sensitivity analysis.

MPC Prediction Horizon: In this section, the prediction horizon N^{p} varies, while the MPC sampling time remains constant, i.e., $T^{\mathrm{ctrl}} = 30$ time units.

Figure 5.30 presents the mean objective function over time, across 5 simulation runs, each simulated over 5000 time units. These values have not been normalised either for a similar reason to the sampling time sensitivity analysis.

Figure 5.31 shows the mean optimisation time over time for this sensitivity analysis.

5.5.3. RESULTS FOR M2PFC DESIGN EXPLORATION

This section explores potential design choices for M2PFC that may enhance its performance (with respect to metrics given in Section 5.4.6), computational efficiency, or our insight about pros and cons of various configuration options for the M2PFC framework.

Design factors considered include the choice of the prediction mode, FLC controllers (Type-1 vs Type-2), and mapping strategy for robots (i.e., global, computationally expensive mapping vs local, computationally efficient mapping).

Prediction Modes: As explained in Section 5.4.2, two prediction modes are incorporated into the case study, namely the probability threshold mode and the exact prediction mode.

These prediction modes are compared in this section for a two-robot system in a

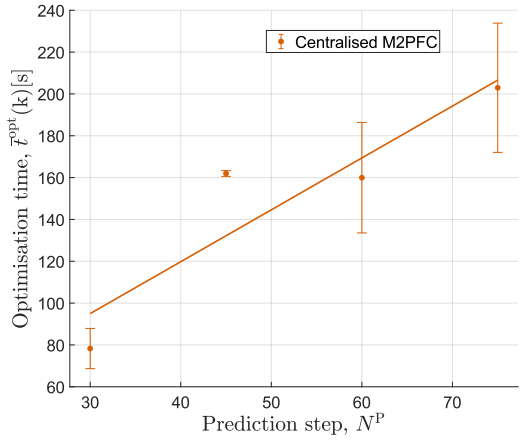


Figure 5.31: Sensitivity, represented for the mean optimisation time $\bar{t}^{\text{opt}}(k)$ over time, with respect to the MPC prediction horizon, running 5 simulations per parameter value, each simulated over 5000 time units.

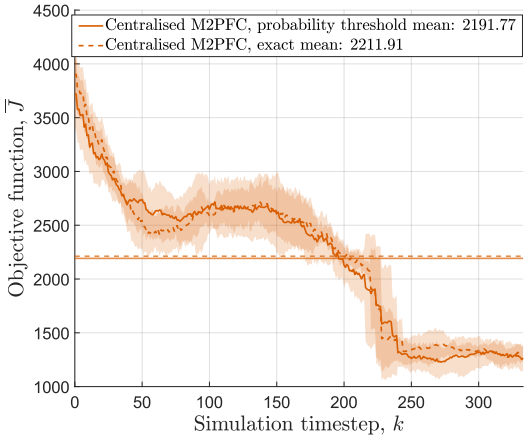


Figure 5.32: The values of the mean of the objective function $\bar{J}(k)$ over time for a two-robot system in a small dynamic disaster environment of size 40×40 cells for the prediction modes, $T^{\text{ctrl}} = 30$ time units, across 5 simulation runs, each simulated over 5000 time units.

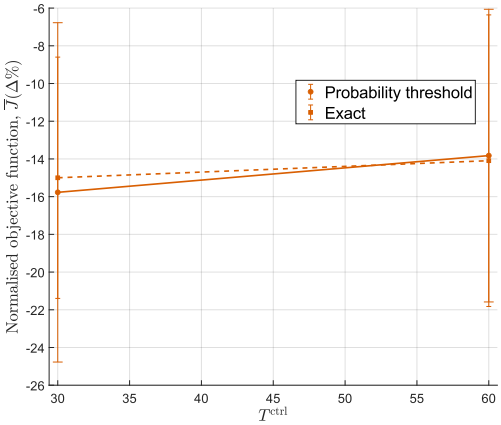


Figure 5.33: Sensitivity, represented for the normalised mean objective function $\overline{J}(k)$ over time, with respect to the MPC sampling time for the prediction modes, running 5 simulations per parameter value, each simulated over 5000 time units.

small dynamic disaster environment of 40 × 40 cells. A relatively large MPC sampling time is chosen ($T^{\text{ctrl}} = 30$ time units), resulting in a larger prediction window for a fixed prediction horizon. Such a design allows to better understand the reliability of the probability threshold prediction mode, which unlike the exact prediction mode has no access to real values of future states, and its prediction errors accumulate over time within the prediction window.

Figure 5.32 shows the evolution of the normalised mean objective function over time, across 5 simulation runs, each simulated over 5000 time units. These values have been normalised against the performance of the pre-tuned FLC system.

To understand the relative difference in controller performance between these two prediction modes, a sensitivity analysis is run against T^{ctrl} for centralised M2PFC in the range [30 time units, 60 time units], and it is presented in Figure 5.33.

Type-1 vs Type-2 FLC: In this simulation, we implement M2PFC within a grid environment of 40 × 40 cells under with-wind conditions, with local Type-2 TSK-FLC controllers, which model two — instead of one — level of fuzzy uncertainties. We compare the performance of this framework against the original design of M2PFC including local Type-1 TSK-FLC controllers.

Type-1 FLC controllers use one level of membership functions, where each input is associated with a single degree of fuzziness. This simplicity allows for efficient computations and straightforward implementations. In contrast, Type-2 FLC controllers employ a primary and a secondary membership function, i.e., each input is associated with two levels of fuzziness. While Type-1 FLC controllers consider no uncertainty about the fuzziness degree, Type-2 FLC controllers model the level of uncertainty about the fuzziness degree. This potentially enhances reliability of the system when dealing with uncertain variables, at the expense of increased computations.

Figure 5.34 presents the evolution of the normalised mean objective function over

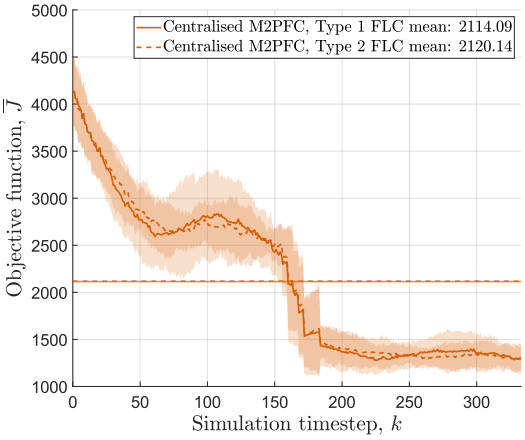


Figure 5.34: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a small dynamic disaster environment of size 40 × 40 cells for Type-1 vs Type-2 FLC in M2PFC architecture, across 5 simulation runs, each simulated over 5000 time units.

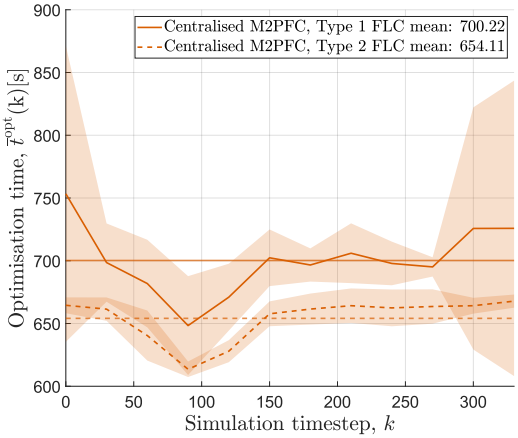


Figure 5.35: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a two-robot system in a small dynamic disaster environment of size 40 × 40 cells for Type-1 vs Type-2 FLC in M2PFC architecture, across 5 simulation runs, each simulated over 5000 time units.

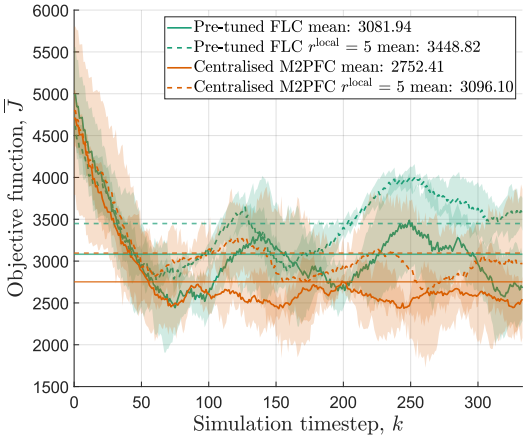


Figure 5.36: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a small static disaster environment of size 20×20 cells for neighbourhood radius r^{local}, across 3 simulation runs, each simulated over 5000 time units.

5

time, across 5 simulation runs, each simulated over 5000 time units, while Figure 5.35 shows the evolution of the mean optimisation time over time.

Local Prediction Maps: The optimisation time for M2PFC increases with the size of the environment and dimensions of corresponding maps (see Figures 5.26 and 5.27, as well as Section 5.5.5).

Due to the configuration of the control problem introduced in this case study, robots prioritise nearby cells over distant cells to reduce their response time. Therefore, local controllers may not analyse cells outside a neighbourhood of radius r^{local} of the robot.

Accordingly, to make optimisations more efficient, local map models are proposed to limit predictions to local cells within radius r^{local} of a robot. Predictions run only for the zone that is composed of all cells within distance r^{local} from the robot (see Figure 5.40 for an example, where r^{local} includes a one-cell distance). In case the cells are coarsened, the original cell size is still used for finding the zone with radius r^{local}. A local attraction map is then calculated per robot and is restored to the global map before assigning a target cell to the robot, in order to update the corresponding local information within the global map.

Considering local maps for robots, as explained above, enables to effectively decouple the computational complexity of M2PFC from the size of the environment.

First, the performance of the local map model is analysed for a two-robot system in a basic no-wind scenario with a grid environment composed of 20×20 cells. We simulate the M2PFC framework and the pre-tuned FLC system, using both global and local map models with $r^{\text{local}} = 5$ cells. Figure 5.36 presents the normalised mean objective function over time while Figure 5.37 shows the mean optimisation times over time, in both cases across 3 simulation runs, each simulated over 5000 time units.

Next, we repeat the simulations for a range of values for r^{local}, including $3, 5, 7$, within a larger environment of size 200×200 cells, under with-wind conditions. Figure 5.38

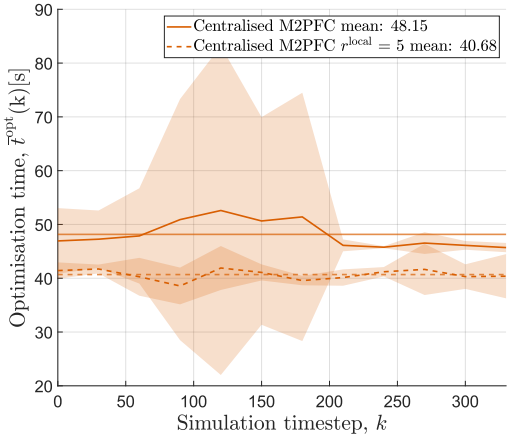


Figure 5.37: The values of the mean optimisation time $\overline{t}^{\text{opt}}(k)$ over time for a two-robot system in a small static disaster environment of size 20×20 cells for neighbourhood radius r^{local}, across 3 simulation runs, each simulated over 5000 time units.

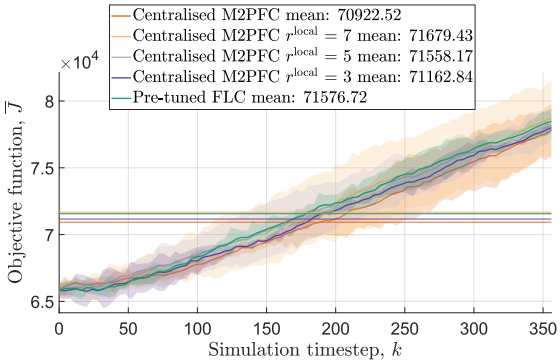


Figure 5.38: The values of the mean of the objective function $\overline{J}(k)$ over time for a two-robot system in a large dynamic disaster environment of size 200×200 cells for neighbourhood radius r^{local}, across 3 simulation runs, each simulated over 5000 time units.

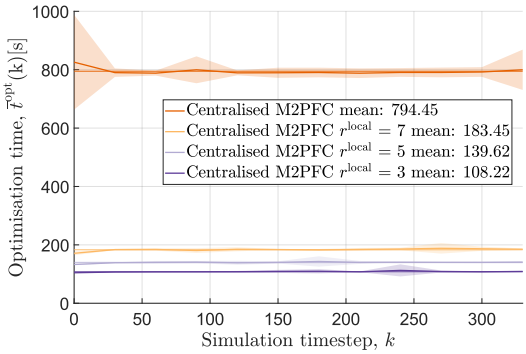


Figure 5.39: The values of the mean optimisation time $\bar{t}^{\text{opt}}(k)$ over time for a two-robot system in a large dynamic disaster environment of size 200 × 200 cells for neighbourhood radius r^{local}, across 3 simulation runs, each simulated over 5000 time units.

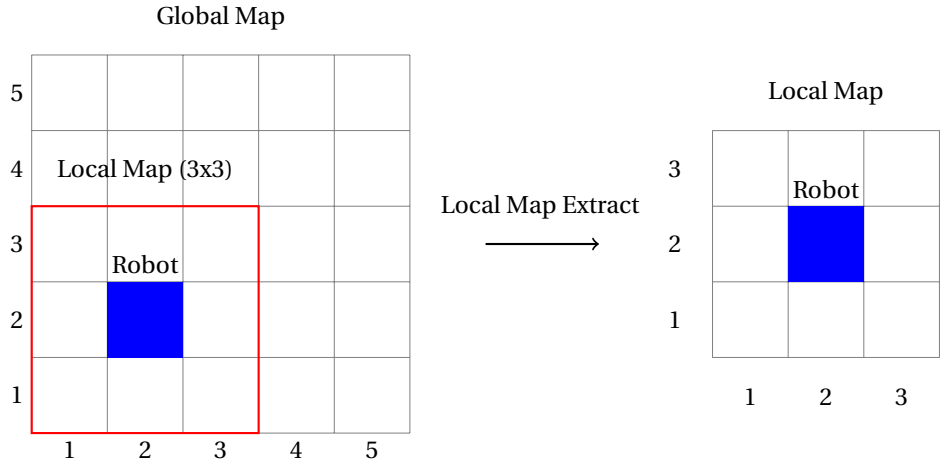


Figure 5.40: Extraction of local map from global map, where $r^{\text{local}} = 1$.

presents the normalised mean objective function over time and Figure 5.39 shows the mean optimisation times over time, in both cases across 3 simulation runs, each simulated over 5000 time units.

5.5.4. DISCUSSION OF RESULTS FOR M2PFC PERFORMANCE ANALYSIS

Next, we discuss the results of the M2PFC performance analysis.

Basic No-wind Scenarios: In these scenarios (see Figure 5.12), the objective function starts around 5000, then rapidly decreases as robots scan more cells, and stabilises once most cells have been observed. This illustrates the effect of the sensor accuracy component of $M^{\text{scan}}(k)$ (see (5.1)) in the objective function (see (5.21)). It has a significant overall contribution to the objective function at the start of the simulation as all cells begin unscanned, and reaches a stable state when there is a balance between the rate that robots scan cells and the rate of degradation of $M^{\text{scan}}(k)$ due to the scan certainty decay. The pre-tuned FLC system demonstrates the poorest mean performance at 2925, followed by the centralised M2PFC with 2705 (−7.5 %) and the centralised MPC-only system with 2440 (−16.6 %). Note that the initial performance of the pre-tuned FLC system is very close to the centralised M2PFC, suggesting that the initial tuning of the FLC is already proper. Although centralised M2PFC does not outperform centralised MPC-only system, most of the difference in performance is during the first 1000 time units of the simulation, where the centralised MPC-only system is able to optimise the initial cells, while both stabilise around a similar value of the objective function over the remainder of the simulation. Additionally, the confidence intervals for centralised M2PFC are much tighter. From Figure 5.13, centralised M2PFC has a mean of 48 time units, almost half of the mean of 93 time units corresponding to the centralised MPC-only system. The optimisation time remains stable for both control methods over the simulation.

Figure 5.14 reflects the effect of the fire propagation on the objective function, with the number of cells possessing a burning fire state reaching its peak at around 120 time units, and most cells possessing an extinguished fire state around 230 time units. In this case, the centralised MPC-only system has a mean objective function of 2618, followed by the pre-tuned FLC system with 2490 (−4.9 %) and the centralised M2PFC framework with 2375 (−9.3 %). With the introduction of the uncertain fire spread component (i.e., with respect to the previous scenario), the performance of the centralised MPC-only system has degraded relative to the pre-tuned FLC system, while the centralised M2PFC framework achieves the best performance.

From Figure 5.15, centralised M2PFC has a mean optimisation time of 56 time units, while centralised MPC-only system has as optimisation time of 122 time units (+118 %). The optimisation times for all control methods are higher due to the additional complexity of predicting the fire spread. Unlike the static case, the optimisation time is highly variable for the centralised MPC-only system, while centralised M2PFC maintains a stable optimisation time over the course of the simulation.

As shown in Figure 5.16, the pre-tuned FLC system demonstrates the poorest mean performance at 2176, followed by the centralised MPC-only system with 2044 (−6.1 %) and centralised M2PFC with 1849 (−15.0 %). By introducing more robots in the same disaster environment, the objective function values decrease. Robot dynamics becomes more closely coupled and the performance of the MPC-only system and the M2PFC

framework is improved relative to the pre-tuned FLC system, as they are able to optimise individual robot behaviours against predicted future states. Figure 5.17 shows that centralised M2PFC has a mean optimisation time of 97 time units, while centralised MPC-only system has as optimisation time of 148 time units (+53 %). Compared to the two-robot case, the optimisation time for the centralised MPC-only system is closer to that of centralised M2PFC. This is due to the larger number of optimisation variables for centralised M2PFC versus the centralised MPC-only system.

Centralised vs Decentralised M2PFC: From Figure 5.18, the pre-tuned FLC system demonstrates the poorest mean performance at 2599, followed by decentralised M2PFC with 2320 (−10.7 %), and the centralised M2PFC framework with 2275 (−12.5 %). In this case both architectures achieve a similar performance, but there is a trade-off between the number of optimisation variables due to the number of robots in the system. Figure 5.19 shows that the centralised M2PFC framework has a mean optimisation time of 601 time units, while the decentralised M2PFC framework has as optimisation time of 604 time units (+0.5 %).

As shown in Figure 5.20, the pre-tuned FLC system demonstrates the poorest mean performance at 2176, followed by the centralised M2PFC framework with 1849 (−15.0 %) and the decentralised M2PFC framework with 1807 (−17.0 %). Likewise, Figure 5.21 shows that the centralised M2PFC framework has a mean optimisation time of 97 time units, while the decentralised M2PFC framework has as optimisation time of 93 time units (−4 %). In this case, the decentralised M2PFC framework outperforms the centralised M2PFC framework in both metrics despite the fact that the behaviour of the robots is more coupled. This likely comes down to a trade-off between the degree of coupling between robot actions and the number of optimisation variables, which for the centralised M2PFC framework scales with the number of robots in the system.

Two-robot System in Complex Dynamic Disaster Environment: As shown in Figure 5.22, the pre-tuned FLC system demonstrates the poorest mean performance at 7181, followed by the centralised MPC-only system with 6437 (−10.4 %) and the decentralised M2PFC framework with 6283 (−12.5 %). The centralised MPC-only system achieves more stable performance over the entire simulation. This may be due to the MPC-only system having direct control over robot target cells, which allows it to always respond to cells with burning fire state regardless of the positions of the fires or robots, while M2PFC can only control robot actions indirectly via the output parameters of the FLC. This highlights the importance of the design of the FLC. For instance, by choosing alternative input parameters, it may be possible for M2PFC to tune robots to prioritise certain geographical areas of the disaster environment.

5.5.5. DISCUSSION OF RESULTS FOR M2PFC SENSITIVITY ANALYSIS

This section discusses the results of the sensitivity analyses performed for the M2PFC framework, as illustrated in Section 5.5.2.

Number of Robots: As shown in Figure 5.24, all predictive controller configurations outperform the pre-tuned FLC system as the number of robots (n^{rob}) increases. Consistent with previous observations, the decentralised implementation of the M2PFC framework demonstrates slightly better performance than its centralised counterpart, with a similar trend observed for decentralised and centralised MPC-only systems.

Across the tested range of n^{rob}, the centralised implementation of the M2PFC framework consistently yields approximately 10 % better performance than the centralised MPC-only system, while the decentralised M2PFC framework maintains an approximately 5 % improvement over its MPC-only counterpart.

Figure 5.25 shows that optimisation times for the decentralised architectures remain stable as the number of robots increases. However, since each robot performs a separate optimisation, the computational benefit hinges on the ability to parallelise these optimisations. The variability in optimisation times is significantly lower for M2PFC-based architectures compared to MPC-only systems, as reflected in the narrower confidence intervals.

Disaster Environment Size: Figure 5.26 indicates that M2PFC maintains a consistent performance advantage of 10 % over the pre-tuned FLC system across varying environment sizes. Interestingly, the MPC-only system appears to perform relatively better in large environments than in smaller ones. However, this observation is influenced by the normalisation process, i.e., in absolute terms the mean objective function increases with the environment size, due to the larger number of cells and increased complexity.

The performance trend for the MPC-only system does not align linearly with all data points. For example, the confidence intervals at 1600 cells are not intersected by the trend line. This suggests a potentially nonlinear relationship.

From Figure 5.27, both controller types exhibit a linear increase in optimisation time with respect to environment size, and neither controller demonstrates a clear scalability advantage over the other in this respect.

MPC Sampling Time: Figures 5.28 and 5.29 show that performance of the controller remains relatively stable within the range of 10 time units to 25 time units for T^{ctrl}. Beyond this range, the performance degrades by increasing the control sampling time. Optimisation time shows a strong linear correlation with the control sampling time, as larger values for T^{ctrl} reduce the frequency of optimisations, while increasing the complexity of each one. These results suggest that an optimal value for the control sampling time that balances computational cost and performance lies within the range of 10 time units to 25 time units. Minimising T^{ctrl} within this range may yield the best objective function outcomes while avoiding excessive computational demand.

Prediction Horizon: Figure 5.30 shows no strong correlation between the global mean objective function with the prediction horizon. Meanwhile, Figure 5.31 demonstrates a clear linear relationship between the prediction horizon of MPC and the optimisation time. This suggests that extending the prediction horizon offers very limited benefit in the current simulation context, while incurring a significant computational cost.

5.5.6. DISCUSSION OF RESULTS FOR M2PFC DESIGN EXPLORATION

This section discusses the results for the M2PFC design exploration presented in Section 5.5.3.

Prediction Modes: As shown in Figure 5.32, the predicted values for the mean objective function remain consistent across both prediction modes, with overlapping confidence intervals.

The probability threshold prediction mode performs slightly better overall, achiev-

ing a mean objective function of 2191.77, compared to 2211.91 for the exact prediction mode. The most noticeable discrepancy between the two prediction modes occurs at the beginning of the simulation, particularly within the first 60 time units of the simulation, corresponding to the first two times that MPC is called for tuning, when rapid fire spread dominates. During this initial phase, the exact prediction mode yields superior performance, likely due to its access to perfect foresight about the environmental dynamics.

This early advantage does not persist and leads to less favourable performance in subsequent stages. Figure 5.33 shows that both prediction modes achieve a similar relative reduction in the mean objective function of around 15 % at 30 time units, degrading to around 14 % at 60 time units, although wide confidence intervals suggest variability. The observed degradation reflects the accumulation of prediction errors over time.

Overall, the close correlation between the two prediction modes suggests that the probability threshold prediction mode is a viable and practical alternative to the idealised exact prediction mode.

Type-1 vs Type-2 FLC: The comparison between Type-1 FLC and Type-2 FLC controllers indicates no significant difference in performance for the selected simulation scenario. This suggests that under the current level of environmental uncertainty (stochasticity in fire spread only), the additional complexity of a Type-2 FLC controller offers limited benefit.

However, a Type-2 FLC controller may be advantageous in scenarios that involve more pronounced uncertainties or more complex relationships between fuzzy parameters and system behaviour.

Local Prediction Maps: Figure 5.36 shows that in the given small static disaster environment, M2PFC with local prediction maps achieves comparable performance to a pre-tuned FLC controller that uses global maps, while the mean optimisation time is significantly reduced from 48.2 time units to 40.7 time units (which offers a reduction of 16%).

Moreover, the optimisation times are more consistent with local maps, highlighting improved computational stability. These findings support the potential of local maps to reduce computational overhead without compromising decision quality.

In the larger dynamic environment, as shown in Figure 5.38, the mean objective function increases steadily due to the scale of the disaster environment and early fire propagation, which overwhelm the capacity of the multi-robot system. In this context, M2PFC using global maps yields the best mean objective function (70923), while variants with local maps show varying results, i.e., a mean objective function of 71163 for $r^{\text{local}} = 3$ (i.e., −0.3 %), with other configurations performing worse than the pre-tuned FLC.

Despite the lack of a clear performance trend with respect to r^{local}, all M2PFCs frameworks that use local maps achieve performance comparable to or better than the pre-tuned FLC system, while significantly reducing the optimisation time (see Figure 5.39).

These results indicate that the use of local maps is a promising approach for implementing M2PFC in a scalable way, especially in large environments where global mapping is computationally prohibitive.

5.6. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

In this chapter, we introduced Multi-agent Model-Predictive Fuzzy logic Control (M2PFC), a novel hierarchical control architecture for mission planning in multi-robot Search-and-Rescue (SaR) systems in dynamic and uncertain disaster environments. This two-layer control architecture integrates decentralised Fuzzy Logic Control (FLC) for real-time decision making with a supervisory Model Predictive Control (MPC) layer for periodic or event-triggered tuning of FLC parameters. A key innovation lies in the indirect role of MPC, which optimises the parameters of local FLC controllers, rather than injecting direct control actions into the multi-agent system. The resulting control architecture preserves decentralised autonomy and balances responsiveness, offered by the FLC layer, and scalable, context-aware coordination of the robots, provided by the MPC layer.

Beyond the control architecture, we introduced a rich and realistic modelling framework for the SaR environment, including dynamic matrices for fire propagation, victim probability, debris occupancy, and scan certainty. The fire model incorporates wind velocity and direction, combustibility scores, and a cellular automaton-based spread mechanism to enable context-aware hazard prediction. Robot dynamics is also modelled with time-scale separation, wind-influenced motion, and task execution logic. This allows for accurate simulation of traverse and scan actions under environmental constraints.

We validated M2PFC through a comprehensive case study involving both centralised and decentralised implementations, and compared it against conventional MPC-only and pre-tuned FLC systems, across diverse dynamic and uncertain scenarios. The case study included different prediction strategies and extensive sensitivity analyses across varying environmental conditions, control parameters, and architectural configurations, with or without access to global maps at local levels. Results show that M2PFC consistently achieves a favourable trade-off between autonomy, optimality, robustness, and computational efficiency. It outperforms pre-tuned FLC systems in mission performance, implying that re-tuning local parameters guarantees a close-to-optimal performance. It also outperforms MPC-only systems in computational load, where it maintains responsiveness without sacrificing scalability.

These findings demonstrate the potential of M2PFC as a flexible and robust control framework for multi-agent robotic systems. Future work may explore extension of M2PFC to learning-based tuning, integration with real-world sensor data, and deployment in other domains, e.g., as environmental monitoring or cooperative exploration.

We also aim to incorporate additional sources of uncertainty, e.g., wind velocity and direction, which may require robust or stochastic MPC formulations.

Further research can focus on enhancing the modelling and simulation of the disaster environment and robotic agents, including the development of continuous 3D models and integration of sensor models, measurement errors, environment geometry, and structural damage.

Advancements in the design of the M2PFC architecture are also envisioned, including refinement of input and output selection, membership functions, and rule bases for the FLC layer, as well as improvements to the objective function and optimisation solver, and incorporation of probabilistic prediction models within the MPC layer.

DATA AVAILABILITY

The code used to perform the simulations of the case study is publicly available in the 4TU.ResearchData repository [331].

5.7. APPENDIX: FREQUENTLY USED MATHEMATICAL NOTATIONS

In this appendix, Table 5.8 is given including the mathematical notations frequently used throughout the chapter, together with their definitions.

Moreover, Table 5.9 includes the abbreviations used in the chapter.

Variable	Description
$a_{(i,j)}(k^{\text{ctrl}})$	attraction value for cell (i,j) at k^{ctrl}
$E_r^{\text{feasible}}\left(k^{\text{rob}}\right)$	feasible part of the environment for robot r within T^{rob}
$\mathscr{E}$	2D grid discretised SaR environment
$f_{(l,q)}(k)$	spreading potential of cell (l,q) to its neighbouring cells
k	global time step counter
k^{ctrl}	control time step counter for the controller(s)
k^{rob}	local time step counter
$\mathbb{K}^{\text{rob}}(k)$	set of local time steps that occur between global time steps k and $k+1$
$\ell^{\text{h}}_{(i,j)}$	length of side x of the cell (i,j)
$\ell^{\text{v}}_{(i,j)}$	length of side y of the cell (i,j)
$m^{\text{debris}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{debris}}(k)$
$m^{\text{direction}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{direction}}(k)$
$m^{\text{fire}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{fire}}(k)$
$m^{\text{hazard}}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{hazard}}(k)$
$m^{\text{hazard,dir}}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{hazard,dir}}(k)$
$m^{\text{hazard,dist}}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{hazard,dist}}(k)$
$m^{\text{scan}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{scan}}(k)$, i.e., scan certainty state of cell (i,j) at global time step k
$m^{\text{struct}}_{(i,j)}$	$(i,j)^{\text{th}}$ element of M^{struct}
$m^{\text{velocity}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{velocity}}(k)$
$m^{\text{victim}}_{(i,j)}(k)$	$(i,j)^{\text{th}}$ element of $M^{\text{victim}}(k)$
$M^{\text{debris}}(k)$	debris occupancy matrix at global time step k
$M^{\text{direction}}(k)$	wind direction matrix at global time step k
$M^{\text{fire}}(k)$	fire state matrix at global time step k
M^{grid}	grid matrix
$M^{\text{hazard}}(k)$	hazard matrix at global time step k
$M^{\text{hazard,dir}}(k)$	hazard direction matrix at global time step k
$M^{\text{hazard,dist}}(k)$	hazard distance matrix at global time step k
$M^{\text{scan}}(k)$	scan certainty matrix at global time step k
M^{struct}	structure matrix
$M^{\text{velocity}}(k)$	wind velocity matrix at global time step k
$M^{\text{victim}}(k)$	victim probability matrix at global time step k
n^{h}	2D grid x dimension (horizontal)
n^{rob}	number of SaR robots
$n^{\text{perceive}}_{(i,j)}(k)$	perceived number of victims in cell (i,j) at global time step k
n^{v}	2D grid y dimension (vertical)

n^{victim}	maximum number of victims in each cell
$o_{(i,j)}(k)$	perceived proportion of each cell (i,j) that is occupied by debris
$t^{\text{fire}}_{(i,j)}$	global time step when cell (i,j) catches fire
$t^{\text{scan}}_r\left(k^{\text{rob}}\right)$	time required for robot r to scan a cell at time step k^{rob}
$t^{\text{travel}}_r\left(k^{\text{rob}}\right)$	time required for robot r to reach the target cell at time step k^{rob}
T	sampling time for the simulation of the SaR environment dynamics
T^{ctrl}	control sampling time for the controller(s)
T^{rob}	sampling time of the dynamic model per robot(s)
$v_r\left(k^{\text{rob}}\right)$	maximum possible actual velocity of robot r at time step k^{rob}
$v^{\text{nom}}_r\left(k^{\text{rob}}\right)$	nominal velocity of robot r at time step k^{rob}
$\bar{v}_{(i,j),(l,q)}(k)$	average magnitude of the wind velocity between cells (i,j) and (l,q) at time step k (i.e., mean of $m^{\text{velocity}}_{(l,q)}(k)$ and $m^{\text{velocity}}_{(i,j)}(k)$)
x_i	x coordinate of the cell (i,j)
y_j	y coordinate of the cell (i,j)
δ	distance metric
ζ	ignition threshold
η_r	sensor accuracy for robot r
$\boldsymbol{\theta}_r(k^{\text{ctrl}})$	vector of all tuning parameters for the FLC controller corresponding to robot r, and optimisation variables of the MPC problem
$\pi^{\text{fire}}_{(i,j),(l,q)}(k)$	probability that the fire spreads to cell (i,j) from a neighbouring cell (l,q) at global time step k
$\boldsymbol{\rho}(k)$	current robot position at global time step k
σ	scan certainty decay per global time step k
$\boldsymbol{\tau}(k)$	target cell for the robot at global time step k
$\psi_{(i,j),(l,q)}(k)$	angle between the average wind direction (i.e., mean of $m^{\text{direction}}_{(l,q)}(k)$ and $m^{\text{direction}}_{(i,j)}(k)$) and the direction of the fire propagation

Table 5.8: Table of frequently used mathematical notation.

Abbreviation	Meaning
BIBO	Bounded-Input-Bounded-Output
FL	Fuzzy Logic
FLC	Fuzzy Logic Control
MPC	Model Predictive Control
M2PFC	Multi-agent Model-Predictive Fuzzy logic Control
SaR	Search-and-Rescue
TSK	Takagi-Sugeno-Kang

Table 5.9: Table of used abbreviations.

5.8. APPENDIX: ALGORITHMS

This appendix presents core algorithms used in the case study for computing certain dynamic variables related to the fire model, robot model, and task execution.

Algorithm 4 shows how fire risk time — introduced in Section 5.3.2 and used in Section 5.3.4 — is computed. The cell fire risk time $t^{\text{risk}}_{(i,j)}(k)$ for cell (i,j) depends on its vicinity to the nearest cell (l,q) with an igniting fire state, i.e., $m^{\text{fire}}_{(l,q)} = 2$ — adopting a burning fire state in 2 minutes. This is based on the assumption that fire spreads immediately when the corresponding state becomes 3 (which in fact happens when $\pi^{\text{fire}}_{(i,j),(l,q)}(k)$ is above threshold ζ), and it is based on the wind velocity. The fire risk time $t^{\text{risk}}_{(i,j)}(k)$ is set to 100 (a high value) when the cell is non-flammable (i.e., $m^{\text{fire}}_{(i,j)} = 0$) to prevent it from falsely being influenced by burning cells.

Algorithm 4 Calculate fire risk time at time step k

Input: Fire matrix $M^{\text{fire}}(k)$, wind velocity matrix $M^{\text{velocity}}(k)$
Output: Fire risk time matrix $M^{\text{risk}}(k)$
Initialise $M^{\text{risk}}(1)$ as a matrix of size $n^{\text{h}} \times n^{\text{v}}$ with all elements being 100
for each cell (i, j) **do**
  if $m^{\text{fire}}_{(i,j)}(k) = 3$ **then**
    $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow 0$
  else if $m^{\text{fire}}_{(i,j)}(k) \neq 3$ **then**
    for $p \in \{1, 2, 3\}$ **do**
      Get neighbour cells (l, q) that are p cells distant from cell (i, j)
      Extract valid neighbours within grid bounds
      Apply fire rules:
      if $m^{\text{velocity}}_{(i,j)}(k) < 1 \frac{\text{m}}{\text{s}}$ **then**
        if any neighbour cell (l, q) that is 1 cell distant has $m^{\text{fire}}_{(l,q)}(k) = 3$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow 0$
        else if any neighbour cell (l, q) that is 1 cell distant has $m^{\text{fire}}_{(l,q)}(k) = 2$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 2 \text{ minutes}\right\}$
        else if any neighbour cell (l, q) that is 2 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 3$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 2 \text{ minutes}\right\}$
        else if any neighbour cell (l, q) that is 2 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 2$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 4 \text{ minutes}\right\}$
        else if any neighbour cell (l, q) that is 3 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 3$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 4 \text{ minutes}\right\}$
        else if any neighbour cell (l, q) that is 3 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 2$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 6 \text{ minutes}\right\}$
        end if
      else if $1 \frac{\text{m}}{\text{s}} \leq m^{\text{velocity}}_{(i,j)}(k) < 5 \frac{\text{m}}{\text{s}}$ **then**
        if any neighbour cell (l, q) that is ≤ 2 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 3$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow 0$
        else if any neighbour cell (l, q) that is ≤ 2 cells distant $m^{\text{fire}}_{(l,q)}(k) = 2$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 2 \text{ minutes}\right\}$
        end if
      else if $m^{\text{velocity}}_{(i,j)}(k) > 5 \frac{\text{m}}{\text{s}}$ **then**
        if any neighbour cell (l, q) that is ≤ 3 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 3$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow 0$
        else if any neighbour cell (l, q) that is ≤ 3 cells distant has $m^{\text{fire}}_{(l,q)}(k) = 2$ **then**
          $t^{\text{risk}}_{(i,j)}(k+1) \leftarrow \min\left\{t^{\text{risk}}_{(i,j)}(k), 2 \text{ minutes}\right\}$
        end if
      end if
    end for
  end if
end for

Algorithm 5 shows the selection and execution of tasks by robots, when the MPC-only system is used for determining target cells for the robots. In this algorithm, the binary task variable $\beta_r(k)$ determines the task that should be executed by robot r, where $\beta_r(k) = 0$ and $\beta_r(k) = 1$ correspond to "traverse" and "scan", respectively. Moreover, T represents the sampling time, q is a local variable in the algorithm that indicates the position index of a cell within the target queue cells that is generated by the MPC layer for robot r, i.e., the ordered list of all cell indices for the robot to scan across the MPC prediction horizon. Finally, $\left(\tau^{\mathrm{h}}_{r,q}(k), \tau^{\mathrm{v}}_{r,q}(k)\right)$ indicates the target cell corresponding to q in the target queue.

Algorithm 5 Robot action selection and execution at time step k when target cells are determined using MPC-only system

Require: $\left(\tau^{\mathrm{h}}_r(k), \tau^{\mathrm{v}}_r(k)\right)$, $\beta_r(k)$, $t^{\mathrm{scan}}_r(k)$, $t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right),r}(k)$, q, k

$q \leftarrow 1$
if $\beta_r(k) = 0$ and $t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right),r}(k) > 0$ **then**
    $t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right),r}(k+1) \leftarrow t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right),r}(k) - T$
else if $\beta_r(k) = 0$ and $t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right)r}(k) \leq 0$ **then**
    $\beta_r(k+1) \leftarrow 1$
    $\left(\rho^{\mathrm{h}}_r(k+1), \rho^{\mathrm{v}}_r(k+1)\right) \leftarrow \left(\tau^{\mathrm{h}}_{r,q}(k), \tau^{\mathrm{v}}_{r,q}(k)\right)$
    $q \leftarrow q+1$
    $t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_r(k),\tau^{\mathrm{v}}_r(k)\right),r}(k+1) \leftarrow t^{\mathrm{travel}}_{\left(\tau^{\mathrm{h}}_{r,q}(k),\tau^{\mathrm{v}}_{r,q}(k)\right),r}(k)$
else if $\beta_r(k) = 1$ and $t^{\mathrm{scan}}_r(k) > 0$ **then**
    $t^{\mathrm{scan}}_r(k+1) \leftarrow t^{\mathrm{scan}}_r(k) - T$
else if $\beta_r(k) = 1$ and $t^{\mathrm{scan}}_r(k) \leq 0$ **then**
    $q \leftarrow q+1$
    $\beta_r(k+1) \leftarrow 0$
    $t^{\mathrm{scan}}_r(k+1) \leftarrow t^{\mathrm{scan}}_r(k)$
end if

Similarly, Algorithm 6 shows the selection and execution of actions by robots after determining the target cells using either the M2PFC framework or the pre-tuned FLC system. In this algorithm, the simplified notation "$\arg\max\{A_r(k)\}$" is used for returning the indices of the cell with the maximum value in the attraction matrix $A_r(k)$, which contains attraction values $a_{(\tau_x,\tau_y),r}(k)$ for all cells per time step k.

Algorithm 6 Robot action selection and execution at time step k when target cells are determined using M2PFC or pre-tuned FLC system

Require: $A_r(k)$, $\left(\tau_r^{\mathrm{h}}(k), \tau_r^{\mathrm{v}}(k)\right)$, $\beta_r(k)$, $t_r^{\mathrm{scan}}(k)$, $t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k)$, k

if $\beta_r(k) = 0$ and $t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k) > 0$ **then**
  $t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k+1) \leftarrow t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k) - T$
else if $\beta_r(k) = 0$ and $t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k) \leq 0$ **then**
  $\beta_r(k+1) \leftarrow 1$
  $\left(\rho_r^{\mathrm{h}}(k+1), \rho_r^{\mathrm{v}}(k+1)\right) \leftarrow \left(\tau_r^{\mathrm{h}}(k), \tau_r^{\mathrm{v}}(k)\right)$
  $t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k+1) \leftarrow t^{\mathrm{travel}}_{\left(\tau_r^{\mathrm{h}}(k),\tau_r^{\mathrm{v}}(k)\right),r}(k)$
else if $\beta_r(k) = 1$ and $t_r^{\mathrm{scan}}(k) > 0$ **then**
  $t_r^{\mathrm{scan}}(k+1) \leftarrow t_r^{\mathrm{scan}}(k) - T$
else if $\beta_r(k) = 1$ and $t_r^{\mathrm{scan}}(k) \leq 0$ **then**
  $\beta_r(k+1) \leftarrow 0$
  $\left(\tau_r^{\mathrm{h}}(k+1), \tau_r^{\mathrm{v}}(k+1)\right) \leftarrow \arg\max\{A_r(k)\}$
  $t_r^{\mathrm{scan}}(k+1) \leftarrow t_r^{\mathrm{scan}}(k)$
end if

Algorithm 7 shows how the hazard matrix introduced in Section 5.4.4 required for computing the objective function of the MPC-based systems via (5.22)–(5.21) is computed. Here, $1_{[m\times n]}$ indicates a matrix of all ones with dimension $m \times n$.

Algorithm 7 Calculate hazard matrix at time step k

Require: $M^{\text{fire}}(k)$, n^{h}, n^{v}
Ensure: $M^{\text{hazard}}(k)$
Initialise $M^{\text{hazard}}(k)$ as a zero matrix of size $n^{\text{h}} \times n^{\text{v}}$
for $i = 1$ to n^{h} **do**
  for $j = 1$ to n^{v} **do**
    if $m^{\text{fire}}_{(i,j)} = 2$ or $m^{\text{fire}}_{(i,j)} = 3$ **then** ▷ Igniting or burning fire state
      Calculate $m^{\text{hazard,dir}}_{(i,j)}(k)$ and $m^{\text{hazard,dist}}_{(i,j)}(k)$ using Equation (5.23) and Equation (5.25)
      $m^{\text{hazard}}_{(i,j)}(k+1) = \max\left\{m^{\text{hazard}}_{(i,j)}(k), m^{\text{hazard,dir}}_{(i,j)}(k) \cdot m^{\text{hazard,dist}}_{(i,j)}(k)\right\}$
    end if
  end for
end for
$M^{\text{hazard}}(k+1) = 1_{[n^{\text{h}}\times n^{\text{v}}]} - M^{\text{hazard}}(k)$ ▷ Invert the hazard effect
return $M^{\text{hazard}}(k)$

Figure 5.41: Colour scheme used for illustrating states of the fire matrix $M^{\text{fire}}(k)$.

5.9. APPENDIX: EXAMPLE SIMULATIONS

This appendix presents example simulations for fire spread and robot motion based on the wind velocity and structural characteristics of the cells. For visualisation of the fire states, the colour scheme in Figure 5.41 is adopted.

Figure 5.42 displays the correlation between fire spread and M^{struct} within an environment of 20×20 cells, assuming that all cells have the same structure state under constant wind velocity $m^{\text{velocity}}_{(i,j)} = 0$. Cells with a burning fire state are shown in red and those with an extinguished state are shown in black. As it is observed in these plots, larger values for $m^{\text{struct}}_{(i,j)}$ significantly increase the spread of fire.

In Figure 5.43, variations in the fire spread probability $\pi^{\text{fire}}_{(i,j),(l,q)}(k)$ is shown within an environment of 9×9 cells — as a function of wind velocity $m^{\text{velocity}}_{(i,j)}$, which is constant across cells and time — illustrated for three different values of $m^{\text{velocity}}_{(i,j)}$. The central cell of the environment is initialised with fire state "burning", i.e., $m^{\text{fire}}_{(5,5)}(1) = 3$. In this example, we set $m^{\text{direction}}_{(i,j)} = \frac{\pi}{4}$ rad and $r^{\text{wind}}_{(i,j)} = 3$ cells, where $r^{\text{wind}}_{(i,j)}$ (i.e., the wind spread radius) determines the number of cells around one with a "burning" fire state that are affected by fire spread. We also observe from the plot that fire does not spread further than 3 cells from the burning cell.

Figure 5.44 demonstrates the relationship between fire spread and wind velocity $m^{\text{velocity}}_{(i,j)}$ within a grid environment of 20×20 cells. In this simulation, the fire is initiated with a "burning" state within the central cell of the environment, with the wind direction set to $m^{\text{direction}}_{(i,j)} = \frac{\pi}{4}$ rad and the structural characteristic of all cells set to $m^{\text{struct}}_{(i,j)} = 0.2$ for $i, j = 1, \ldots, 20$. It is observed that the correlation between fire spread and wind direction is more significant for higher wind velocities, whereas this behaviour may be tuned by adjusting constants $\alpha_1, \ldots, \alpha_4$ defined in (5.4). In this example, we have used $\alpha_1 = 0.1$, $\alpha_2 = 1.2$, $\alpha_3 = 0$, and $\alpha_4 = 1$.

Finally, Figure 5.45 shows an example of the variation of travel time of a robot across a grid environment of 10×10 cells, as a function of wind velocity $m^{\text{velocity}}_{(i,j)}$, which is assumed to remain constant across cells and time. In this example, the velocity of the robot is set to $v_r(k) = 5 \frac{\text{m}}{\text{s}}$ and the wind direction is $m^{\text{direction}}_{(i,j)} = \frac{\pi}{4}$ rad. The robot, represented by a red dot in the figure, is positioned in cell $(5,5)$.

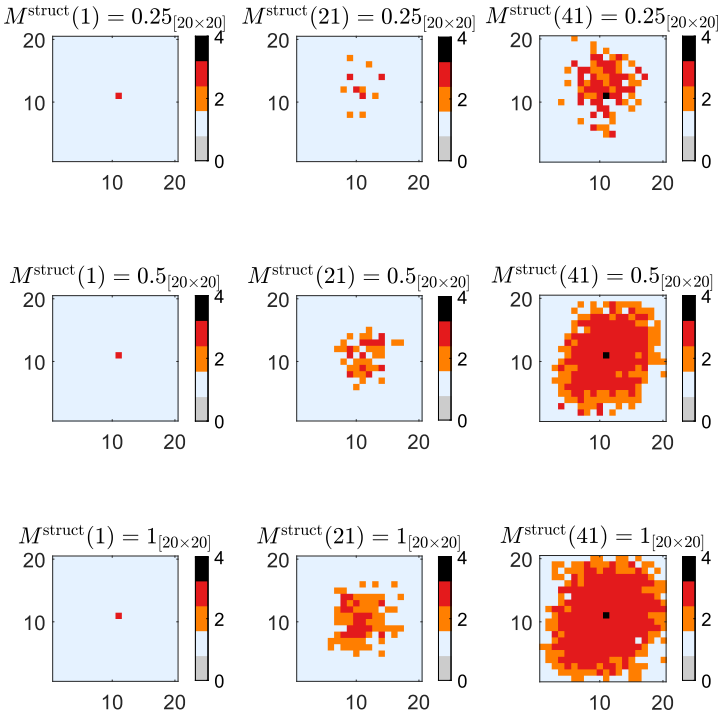


Figure 5.42: Fire spread over time within a grid environment of 20×20 cells, as a function of structural characteristics of cells, encapsulated via M^{struct}, which is constant across cells and time. Each row in the figure represents the fire spread state corresponding to a specific structural characteristic across two stages, each including 20 time steps.

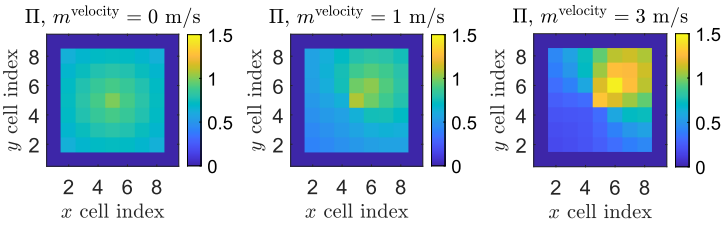


Figure 5.43: Variation of $\pi^{\text{fire}}_{(i,j),(l,q)}(k)$ — values for all cells included in matrix $\Pi(k)$ — within a grid environment of 9×9 cells, as a function of wind velocity $m^{\text{velocity}}_{(i,j)}$, which is constant across cells and time, illustrated for three different values of $m^{\text{velocity}}_{(i,j)}$.

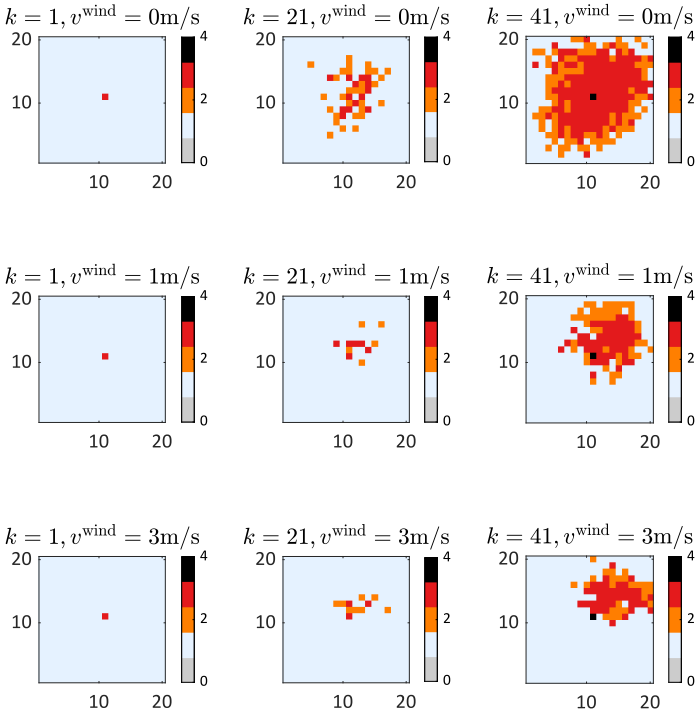


Figure 5.44: Fire spread over time within a grid environment of 20×20 cells, as a function of wind velocity $m^{\text{velocity}}_{(i,j)}$, which is constant across cells and time. Each row in the figure represents the fire spread state corresponding to a specific wind velocity across two stages, each including 20 time steps.

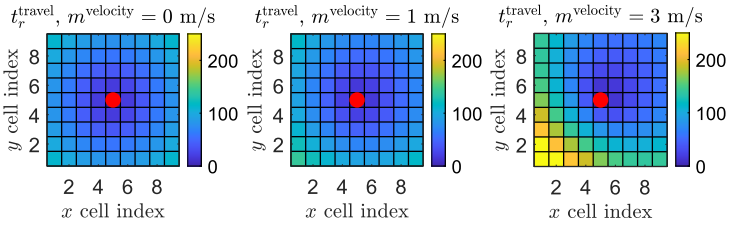


Figure 5.45: Variation of robot travel time $t_r^{\text{travel}}(k)$ to different cells within a grid environment of 9×9 cells, as a function of wind velocity $m^{\text{velocity}}_{(i,j)}$, which is constant across cells and time, illustrated for three different values of $m^{\text{velocity}}_{(i,j)}$. The red dot in the central cell shows the current location of the robot.

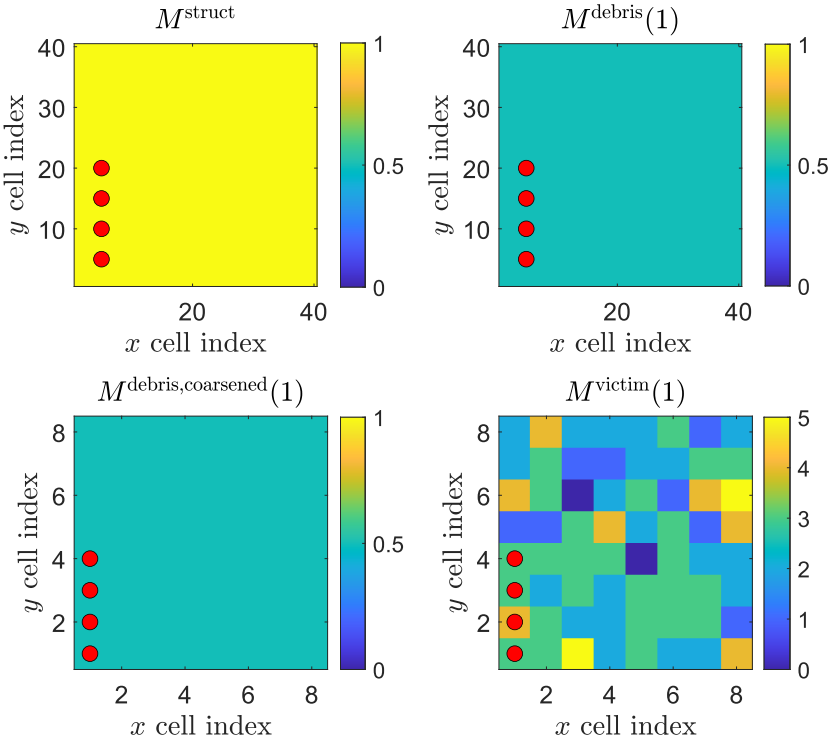


Figure 5.46: Single instance of environment setup for basic small disaster environment, with red dots indicating robot positions. Each subplot represents the value of the labelled environmental state variable for the corresponding cell, from the lower limit (dark blue) to upper limit (yellow) of the parameter range. Debris occupancy and victim probability matrices are coarsened with a factor $\xi = 5$.

5.10. APPENDIX: SCENARIOS

This appendix presents the three main categories of SaR scenarios that have been used in the case studies of this chapter.

Basic No-wind and Basic With-wind Scenarios: The first category of scenarios consists of a basic small disaster environment of 40×40 cells. Figure 5.46 shows an initialisation (at time step $k = 1$) of the environment parameters for this scenario, including four robots. As shown, uniform distributions for the structure matrix M^{struct} and the debris occupancy matrix $M^{\text{debris}}(1)$ are considered, whereas for the victim probability matrix, $M^{\text{victim}}(1)$, the distribution is random across the environment, varying per simulation seed. These matrices are coarsened by a factor $\xi = 5$. The robots, shown as red dots in the figure, are initialised in a row of adjacent coarsened cells in the bottom left corner of the disaster environment.

For basic no-wind scenarios (i.e., with wind velocity equal to zero and no fire spread),

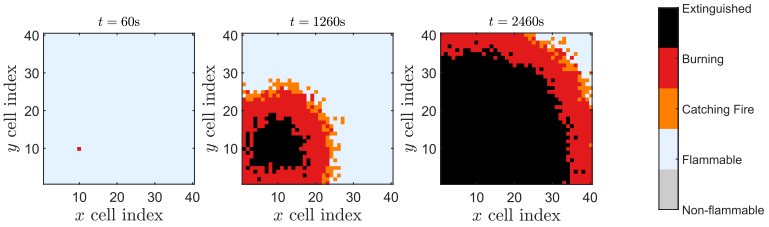


Figure 5.47: Example simulation for fire spread progression for a basic dynamic scenario.

the fire matrix $M^{\text{fire}}(k)$ is initialised with elements $m^{\text{fire}}_{(i,j)}(1) = 1$ for all cells. Since there are no cells with a burning fire state, fire does not spread.

In basic with-wind scenarios, the fire matrix $M^{\text{fire}}(k)$ is initialised using a uniform distribution for flammable cells, and one cell is initialised with a burning fire state. This setup ensures that fire keeps spreading during the simulation, rather than extinguishing prematurely (see Figure 5.47, where fire starts to propagate from cell (20,21), for which $m^{\text{fire}}_{(20,21)}(1) = 3$).

Complex With-wind Scenarios: The second category of scenarios involve an environment of 60×60 cells, with more complex parameters. Figure 5.48 shows an example of such an environment, set up for one simulation seed. The structure matrix M^{struct} is initialised using as a Perlin noise matrix to emulate clusters of buildings with different flammabilities. The debris occupancy matrix $M^{\text{debris}}(k)$ is initialised using the following three probability density functions:

$$\text{PDF}_1(i,j) = 0.5 \cdot \exp\left(-\left((i-0.2)^2 + \left(j-0.3\right)^2\right) / \left(2 \cdot 0.2^2\right)\right),$$

$$\text{PDF}_2(i,j) = 0.5 \cdot \exp\left(-\left((i-0.7)^2 + \left(j-0.4\right)^2\right) / \left(2 \cdot 0.2^2\right)\right),$$

$$\text{PDF}_3(i,j) = 0.5 \cdot \exp\left(-\left((i-0.5)^2 + \left(j-0.8\right)^2\right) / \left(2 \cdot 0.2^2\right)\right),$$

which represent various population centres (i.e., the means of the probability distribution) across the environment.

The wind velocity and direction are set to $m^{\text{velocity}}_{(i,j)} = 1 \frac{\text{m}}{\text{s}}$ and $m^{\text{direction}}_{(i,j)} = \frac{\pi}{4}$ rad, respectively. The fire model is configured to increase the influence of wind direction and velocity on fire spread and to slow down fire spread throughout the simulation. This is achieved by setting the constants of the fire and wind models to $\alpha_1 = 0.8$, $\alpha_2 = 2.25$, $\alpha_3 = -3.5$, and $\alpha_4 = 1.5$ (see (5.4) in Section 5.3.2). The fire matrix, $M^{\text{fire}}(k)$, is initialised with exactly two, randomly chosen, cells being set to the burning fire state.

Figure 5.49 shows the fire spread over time for a single simulation seed. In this case, one sees that several fire fronts form and spread more strongly with the wind direction to the Northeast, while some cells do not catch fire due to their low flammability. In this simulation, fire spread occurs over the entire simulation, and does not have a clear peak early on unlike previous simulations.

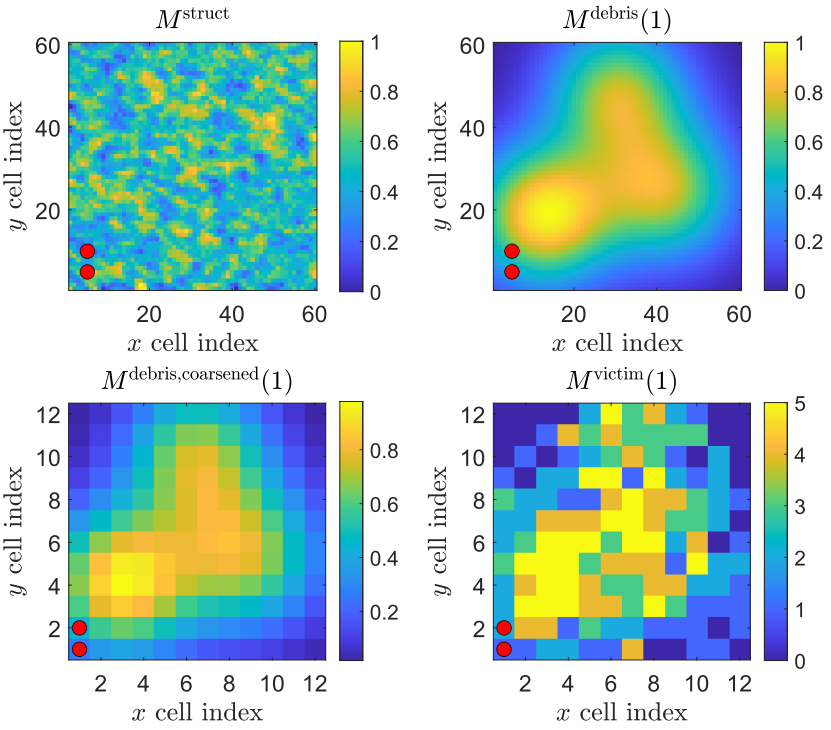


Figure 5.48: Single instance of environment setup for complex disaster environment, with red dots indicating robot positions. Each subplot represents the value of the labelled environmental state variable for the corresponding cell, from the lower limit (dark blue) to upper limit (yellow) of the parameter range. Debris occupancy and victim probability matrices are coarsened with a factor $\xi = 5$.

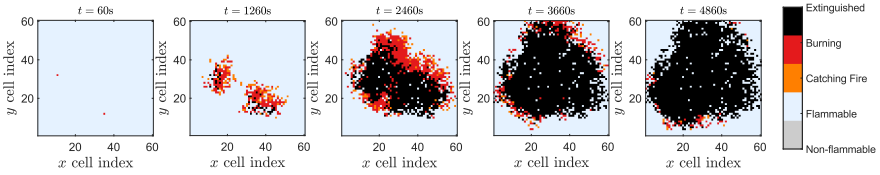


Figure 5.49: Single instance of fire spread for complex disaster environment.

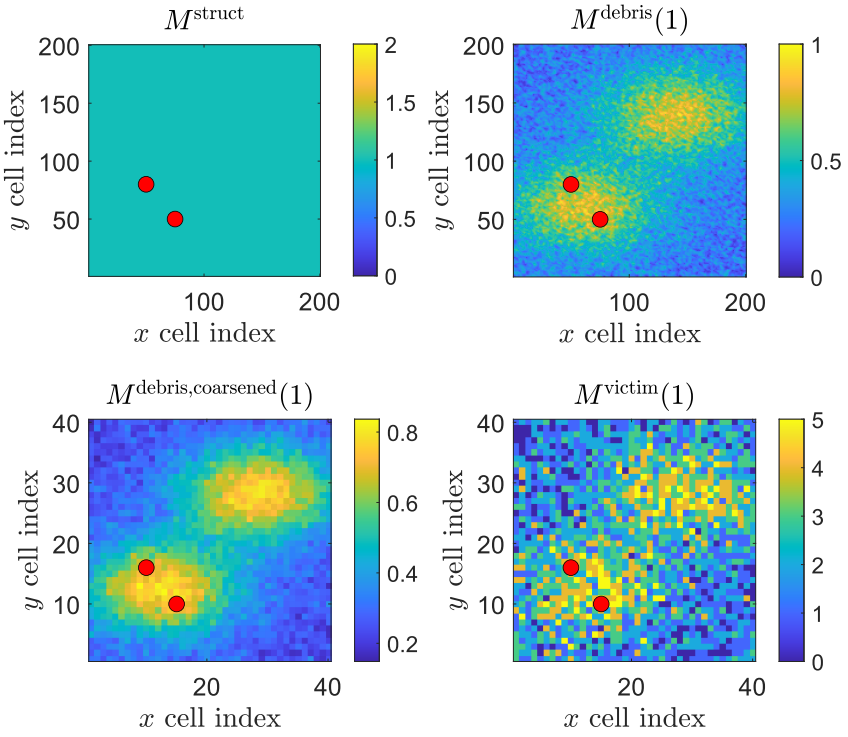


Figure 5.50: Disaster environment setup for local map simulation, red dots indicate robot positions.

The third scenario, shown in Figure 5.50, also involves two robots in a larger complex with-wind disaster environment of dimensions $n^{\text{h}} = n^{\text{v}} = 200$. The initialisation of the environment and robot states is shown in Figure 5.50, where red circles represent initial locations of the robots. The building matrix, $M^{\text{debris}}(k)$, is initialised with two population centres with Gaussian distributions; the structure matrix, M^{struct}, is fixed to ones; and the victim matrix, $M^{\text{victim}}(k)$, is initialised randomly taking into account the building distribution.

The fire matrix is initialised by setting the state of the cells in the bottom left corner to burning fire. The progression of fire spread during one of the simulations is shown in Figure 5.51. Due to the size of the disaster environment and the building matrix, the fire rapidly propagates out from the initial cell with burning fire with a wide radial burning fire front.

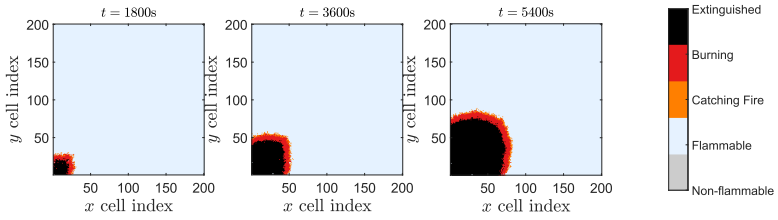


Figure 5.51: Fire spread progression for local map simulation, example simulation.

6

TASK HIERARCHICAL CONTROL FOR MULTI-UAV AUTONOMOUS COORDINATION AND REAL-TIME VISUAL SUPPORT IN DYNAMIC OCCLUDED ENVIRONMENTS

Wildfire suppression is a complex and dangerous task that poses significant risks to human operators. The deployment of robotic teams for wildfire management offers a promising alternative, enhancing safety and operational efficiency in tasks, e.g., fire detection, monitoring, and suppression. However, real-time coordination and motion planning for these robots remain challenging. We propose a task hierarchical control architecture for autonomous navigation of aerial robots, called auxiliary UAVs, that assist human operators who remotely control suppression UAVs. Auxiliary UAVs, equipped with onboard cameras, must maintain best visual coverage of the main UAV, providing the operator with an uninterrupted and occlusion-free view. We incorporate a novel line-of-sight occlusion avoidance method that dynamically determines the best viewpoints for each auxiliary UAV, ensuring continuous visibility of the main UAV. Additionally, path integral control is incorporated to generate optimal motion trajectories for the auxiliary UAVs towards their target positions. We evaluate the proposed framework through an ablation study in simulation environments that have been developed using Matlab and Unity. The results

Parts of this chapter are under review in Robotics and Autonomous Systems.
M. Baglioni contributed to the conceptualization and the methodology, performed the simulations, assessed the results, and wrote the first draft of the chapter. A. Patil contributed to the conceptualization and the methodology, implementation of the simulations, and review of the first draft. L. Sentis and A. Jamshidnejad contributed to the conceptualization and the methodology, assessed the results, edited the final draft, and performed supervision.

demonstrate that our full control framework significantly outperforms baseline configurations that exclude one or more of the components. In particular, the line-of-sight occlusion avoidance task yields up to 44.0% *improvement in maintaining visibility. Moreover, the use of path integral control improves the motion trajectory efficiency by up to* 39.2%.

6.1. INTRODUCTION

USING robots in severe wildfire is expected to reduce the risks to firefighters' lives and to accelerate the firefighting process [19]. Robots can contribute to rapid suppression of the fire by accessing areas that are inaccessible to humans, while enhancing safety, time-efficiency, and affordability. Among various types of robots, flying robots (also referred to as *unmanned aerial vehicles* (UAVs)) are particularly promising for wildfire suppression due to their high maneuverability, ability to provide aerial views, and capacity to cover large areas quickly [14], [15], [333]. These robots can be teleoperated by humans without requiring the operator to approach the fire scene closely. This, however, necessitates an autonomous system that continuously provides the best viewpoints of the suppression robots for the human operator. To support effective wildfire suppression, a team of auxiliary UAVs equipped with cameras may be deployed to follow the main suppression UAV, ensuring continuous, occlusion-free visual coverage of the suppression UAV. This continuous visibility is crucial, as the suppression UAV—steered by a human operator—should promptly and safely reach and position itself above the fire.

Ensuring real-time visual support in such dynamic environments poses significant challenges. The auxiliary UAVs should autonomously coordinate in real time, adapting their motion in order to maintain occlusion-free views of the suppression UAV while avoiding collisions and navigating efficiently [334], [335]. Addressing these challenges is key to enabling robust, autonomous assistance for fire suppression missions.

To this end, we introduce a control architecture that integrates optimization and reactive control strategies to fully automate the behavior of auxiliary UAVs [25], [161], [162], [336]. Specifically we employ *task hierarchical control* (THC) [66] to execute multiple concurrent tasks with different priority levels, including maintaining occlusion-free views and ensuring safe navigation. For tasks requiring optimal trajectory tracking, e.g., closely following the suppression UAV, we use *path integral control* (PIC) [21], a stochastic optimal control approach well suited for real-time applications. Additionally, to determine optimal viewpoints of the suppression UAV in cluttered wildfire environments, we adopt the state-of-the-art best viewpoints method from [83]. The main contributions of this chapter are the following:

- We achieve a consistently occlusion-free field of view (FoV) centered on the suppression UAV over extended periods by introducing a novel line-of-sight (LoS) obstacle avoidance method. Unlike traditional approaches focused solely on collision avoidance, our method actively maintains LoS to track the suppression UAV, while ensuring clear visibility and safety.

- We determine optimal viewpoints for auxiliary UAVs by integrating a state-of-the-art best viewpoints method for obstacle-prone environments with our approach, which synergizes THC and PIC. This fusion creates a novel architecture that continuously enhances both the visibility of the main UAV and the coordination of the auxiliary UAVs.

- We apply the resulting THC-PIC-Best-Viewpoint architecture in a pioneering case study, marking the first use of fully autonomous UAVs working alongside human operators in fire monitoring and suppression.

In Section 6.2 we discuss the related background; Section 6.4 explains the proposed control architecture and methods; in Section 6.5 we present the results of a case study that are analyzed and discussed in Section 6.6; Section 6.7 concludes the chapter and suggests topics for future work. In Appendix 6.7, Table 6.7 gives the mathematical notations frequently used in the chapter.

6.2. BACKGROUND

In multi-UAV wildfire suppression, UAVs should simultaneously handle collision avoidance, reach target viewpoints, and navigate safely. By prioritizing costs, UAVs should efficiently optimize multi-objective tracking performance. This highlights the relevance of hierarchical control and active perception for this chapter. Below, we provide the necessary background discussions.

6.2.1. TASK HIERARCHICAL CONTROL (THC)

Task Hierarchical Control (THC) [66] is well-suited for high-dimensional systems to optimize multi-task tracking based on prioritized objectives. The method proposed in [337] handles both strict and non-strict priorities of an arbitrary number of tasks, simultaneously achieving multiple priority re-arrangements. In [338], a passivity-based hierarchical tracking control system with strict priorities has been introduced to handle an arbitrary number of conflicting tasks, proposing a new control objective for autonomous tracking of locally optimal trajectories. These methods, however, leave gaps in computational efficiency, applicability to robots with multiple degrees of freedom, and handling singular configurations. THC is employed by De Benedittis et al. [339] to manage task priorities in multi-robot systems facing multiple conflicting objectives. By assigning a separate Model Predictive Control (MPC) system to each task, the framework is enhanced with task-specific optimality, while enforcing a hierarchical structure, managed by the THC framework, to resolve conflicts. The proposed method is validated through computer-based simulations and is compared to a state-of-the-art baseline approach that handles all tasks simultaneously without prioritizing them. The results demonstrate improved task coordination, but only in static environments. However, leveraging the proposed hybrid framework for dynamic environments with moving obstacles remains challenging, because the computation time of MPC is high even when just one static obstacle is considered in the constraints.

THC has been successfully applied to various robotics domains, e.g., industrial manipulators [337], [340], legged robots on uneven terrains [341], UAV formations [342], and motion planning of space robots [343]. However, maintaining an occlusion-free LoS for UAVs is often excluded from the task hierarchy when controlling these complex robotic systems. Moreover, optimality of the trajectories remains unaddressed, as THC typically relies on proportional-integral-derivative (PID) controllers or their individual components, whereas replacing PID with path-integral-control-based systems is expected to enhance optimality.

6.2.2. PATH INTEGRAL CONTROL (PIC)

Path integral control (PIC) [21] is a sampling-based stochastic optimal control method. PIC has been extended to enhance smoothness of the generated control inputs, which traditionally are prone to significant chattering due to the stochastic nature of PIC. It has also been effectively used to robustly handle uncertainties in the dynamic models of controlled systems [344], [345]. These methods, however, can be computationally intensive and require simplified system dynamics.

PIC has been applied across diverse robotics domains, e.g., in path following via mobile robots [346], for jumping of quadruped robots, for opening a door by humanoid robots [347], for robotic arm manipulations [345], and in autonomous path tracking with obstacle avoidance [348]. Despite optimality, these approaches face challenges in avoiding local minima. Alternative learning-based methods also struggle with learning efficiently. Furthermore, while PIC has been applied to relatively simple robots, extending it to complex systems with high degrees of freedom, e.g., multi-UAV suppression systems, remains an open challenge.

6.2.3. ACTIVE SENSING AND BEST VIEWPOINTS APPROACH

In wildfire suppression missions, a group of auxiliary UAVs monitors a fire-suppression UAV and provides occlusion-free visual frames to assist a human operator. This setup introduces an *active sensing* challenge that bridges control and perception, requiring UAVs to adapt their behavior to actively gather relevant information [3], [349]. Such adaptability is crucial for firefighting UAVs to effectively explore disaster scenes and to plan appropriate actions [25], [72].

A fundamental factor in addressing this challenge is the precise determination of the viewpoints of the auxiliary UAVs, which significantly affects the overall performance of the multi-UAV system. State-of-the-art viewpoint selection includes the method in [83], where 31 expert operators were tasked with guiding a robot through various navigation scenarios from different viewpoints. The highest-scored viewpoints were identified as the “best viewpoints”. They, however, did not account for environments with obstacles—a limitation addressed by our proposed control architecture. Additionally, existing methods have not been applied to the specific problem of wildfire suppression, a gap this chapter fills.

6.2.4. LINE-OF-SIGHT (LOS) OBSTACLE AVOIDANCE

Existing methods for line-of-sight (LoS) obstacle avoidance include [350], which relies solely on the UAV-obstacle LoS angle as feedback, providing collision avoidance in situations where the position and velocity of the obstacle cannot be measured. In [351] and [352], LoS-based guidance is used for obstacle avoidance deploying, respectively, a fuzzy-logic-based reactive controller for autonomous underwater vehicles and a PID control strategy based on the distance of the vehicle from the obstacle [352].

LoS can also be leveraged for predicting collisions. In [353], LoS is used for probabilistic prediction of collision risks in autonomous vehicles, whereas [354] proposes an algorithm to detect and prevent potential collisions beyond the LoS. These approaches, however, focus on generating warnings for possible collisions rather than proactively controlling the system to avoid them. In summary, a crucial gap in literature is that LoS

is typically used for obstacle avoidance or for collision predictions, but not for tracking targets. To the best of our knowledge, our method is the first to leverage the distance between an obstacle and the LoS to a moving target to guarantee both collision avoidance and occlusion-free visual tracking, *while simultaneously maintaining the target (i.e., the suppression UAV) within the field of view (FoV) of the system (i.e., auxiliary UAV)*. This enables a more complex and integrated task, i.e., target tracking with guaranteed collision avoidance.

6.2.5. UAVS FOR WILDFIRE SUPPRESSION

UAVs are increasingly being used to support wildfire management missions, which typically involve three main phases: detection [355], monitoring [356], [357] and suppression [358]. Most existing works focus on a specific phase. For instance, [356] presents a multi-UAV coordination strategy for reducing uncertainty in fire boundary estimation during the monitoring phase. In [359], a scheduling approach for monitoring and suppression tasks is introduced, addressing the dynamic nature of wildfire environments to improve fire containment and area coverage. For the suppression phase, [358] proposes an optimal controller that enables a UAV to perform risky maneuvers beyond its nominal flight envelope, in order to drop fire extinguishing agents onto the fire. In [67], the authors introduce an architecture that combines a single detection UAV with multiple suppression UAVs. Reinforcement learning is employed to optimize the strategy of the detection UAV for informing the suppression UAVs about fire locations.

All these approaches, however, assume that UAVs operate above the trees, where obstacle avoidance is not required. Moreover, none of them considers auxiliary UAVs that track and support the extinguishing UAV during its operation, an essential component that has been introduced in this chapter.

6.3. PROBLEM STATEMENT

The modeling and control paradigms of this chapter are developed for a 3D continuous space within a continuous time framework with time variable t. We consider a multi-UAV system composed of one *main UAV*—used for fire suppression—and n^{aux} *auxiliary UAVs*. Each auxiliary UAV is equipped with a camera characterized by a conic FoV with view angle θ, and is assumed to have perfect perception. The main UAV is teleoperated by a human, flying over the fire to drop fire suppression materials. The auxiliary UAVs autonomously monitor the main UAV, ensuring that it remains within the FoVs of their cameras. The images captured by these cameras are relayed to the human operator to support effective teleoperation.

Let $\boldsymbol{q}^{\text{main}}(t) \in \mathbb{R}^3$ denote the configuration (i.e., 3D position) of the main UAV, and $\boldsymbol{q}_a^{\text{aux}}(t) \in \mathbb{R}^3$ denote the configuration of auxiliary UAV indexed a, for $a = 1, \dots, n^{\text{aux}}$.

We also consider n^{obs} number of static, convex-shaped obstacles modeled as spheres. Each obstacle o is described by its center $\boldsymbol{q}_o^{\text{obs}} \in \mathbb{R}^3$ and radius $r_o^{\text{obs}} \in \mathbb{R}$, with $o = 1, \dots, n^{\text{obs}}$.

The dynamics of each auxiliary UAV is governed by:

$$\dot{\boldsymbol{q}}_a^{\text{aux}}(t) = f\left(\boldsymbol{q}_a^{\text{aux}}(t)\right) + g\left(\boldsymbol{q}_a^{\text{aux}}(t)\right) \boldsymbol{u}_a^{\text{aux}}(t) \tag{6.1}$$

where $\boldsymbol{q}_a^{\text{aux}}(t)$ denotes the state vector (i.e., the configuration) of auxiliary UAV a at time t, $\boldsymbol{u}_a^{\text{aux}}(t)$ is its control input (i.e., the acceleration of the auxiliary UAV), and $f(\cdot)$ and

$g(\cdot)$ represent the nonlinear state and input functions of the system, respectively. In fact, (6.1) describes how the position of the auxiliary UAV evolves over time, where its velocity is influenced by the acceleration, i.e., the control input of the dynamic equation. We assume that in reality the auxiliary UAVs are guided via velocity commands, thus each auxiliary UAV a receives the value of $\dot{\boldsymbol{q}}_a^{\text{aux}}(t)$ to move accordingly.

Remark 12. *We assume that each auxiliary UAV has perfect knowledge of its own state, as well as access to the state of the main UAV with bounded uncertainty. This information allows to estimate the desired distance ρ^{main} between auxiliary UAVs and the main UAV, knowing the radius (uncertainty in the main UAV state) of the cone base. The desired distance should guarantee that the visibility of the main UAV remains within the FoV of the auxiliary UAV, even under the presence of estimation uncertainties.*

6.4. PROPOSED METHODOLOGIES

Next, in Section 6.4.1, we elaborate our proposed approaches. Then in Section 6.4.5, we present a theorem that guarantees our novel LoS occlusion avoidance method ensures continuous, non-occluded visibility of the main UAV by the auxiliary UAVs.

6.4.1. AUTONOMOUS CONTROL OF AUXILIARY UAVS

In the fire suppression mission, each auxiliary UAV should simultaneously pursue multiple objectives—including navigating to effective viewpoint locations, avoiding collision with obstacles and other UAVs, and maintaining a safe, yet sufficiently close, distance from the main UAV. To this end, the auxiliary UAV follows a trajectory determined by a PIC controller that steers the UAV towards designated locations, referred to as *waypoints*. These waypoints are generated in response to the trajectory of the main UAV, as influenced by the human operator.

At the current waypoint, the autonomous control system performs the following 3 key actions:

Action 1 **Determining the next waypoints** for all auxiliary UAVs using state-of-the-art heuristic best viewpoints method

Action 2 **Executing prioritized tasks using THC** to enable the auxiliary UAVs to simultaneously manage multiple objectives

Action 3 **Localizing the fire and the main UAV** based on perceived data, ensuring accurate positioning and tracking by auxiliary UAVs.

In the following sections, we explain how each of these actions is performed.

6.4.2. ACTION 1: DETERMINING THE NEXT WAYPOINTS

To position themselves at the best viewpoints, auxiliary UAVs should account for the task that is currently being performed by the main UAV. Two tasks are considered for the main UAV:

- *Reachability*—when the main UAV is navigating towards its target position above the fire

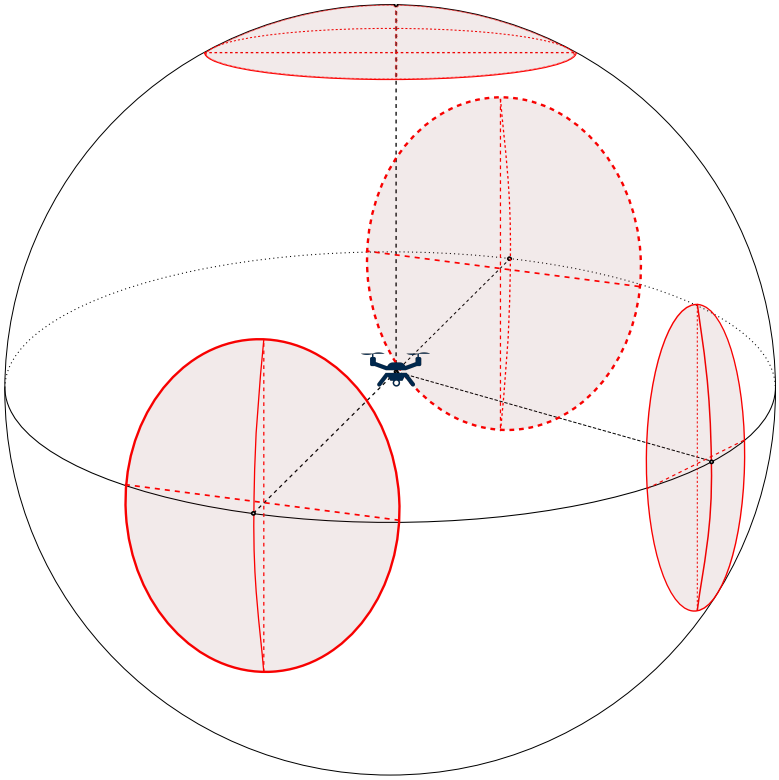


Figure 6.1: Illustration of approximate areas that include the best viewpoints for the auxiliary UAVs, based on the front, top, left, or right views from the main UAV.

- *Manipulability*—when the main UAV is performing the fire suppression action by releasing fire extinguishing material

Based on the findings in [83], the most informative viewpoints vary depending on these tasks. For reachability, the front and top views allow the human operator to effectively perceive the approach and trajectory of the main UAV. For manipulability, the right and top views offer the clearest visual access to the suppression actions (see Figure 6.1 for an approximate illustration). In symmetric settings, left views may equally be effective for manipulability.

These viewpoint preferences are deployed to guide the auxiliary UAVs in real time. To this end, the following simple algorithm is utilized to determine waypoint $\boldsymbol{p}_a^{\mathrm{wp}}(t)$ for auxiliary UAV a, for all $a \in \{1, \ldots, n^{\mathrm{aux}}\}$ and starting from $a = 1$:

Step 1 Determine areas that provide the best viewpoints with respect to the main UAV, using the heuristic approach based on [83]

Step 2 Select one area that has not yet been selected and assign it to "current area"

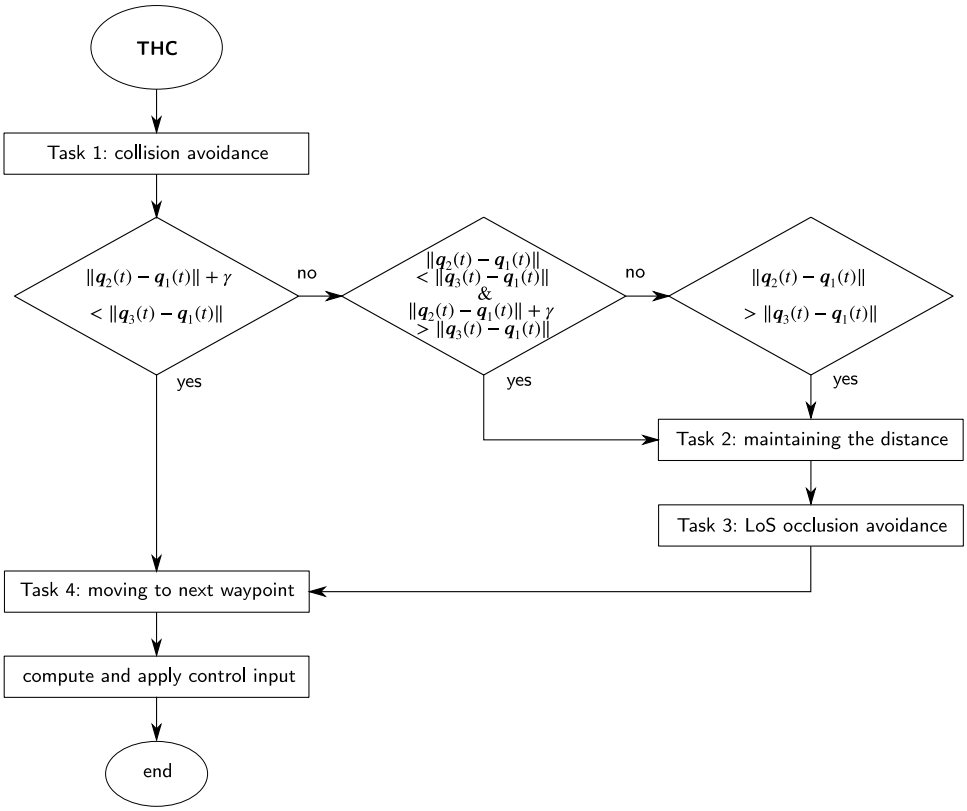


Figure 6.2: Flow chart of the THC framework, for all the UAVs at a given time t.

Step 3 Determine the centroid of "current area" and assign it to variable $\boldsymbol{p}_a^{\mathrm{wp}}(t)$

Step 4 If $a \neq n^{\mathrm{aux}}$, then go to **Step 2**

6.4.3. ACTION 2: EXECUTING PRIORITIZED TASKS USING THC

High-dimensional or redundant systems can effectively be controlled using THC [66], enabling them to simultaneously perform multiple tasks by prioritizing objectives. Using the null space projection method [360], tasks are arranged based on a hierarchy. Higher-priority tasks utilize major capabilities of the controlled system, whereas lower-priority tasks are executed within the remaining degrees of freedom of the controlled system, ensuring that they will not interfere with the execution of higher-priority tasks. Nevertheless, all tasks are performed in parallel, to the extent that is possible by the redundancy of the controlled system.

In our framework, THC automates a team of auxiliary UAVs for tracking and monitoring a fire suppression UAV. We design a four-level hierarchy of tasks for the auxiliary UAVs, executed in descending order of priority (see Figure 6.2):

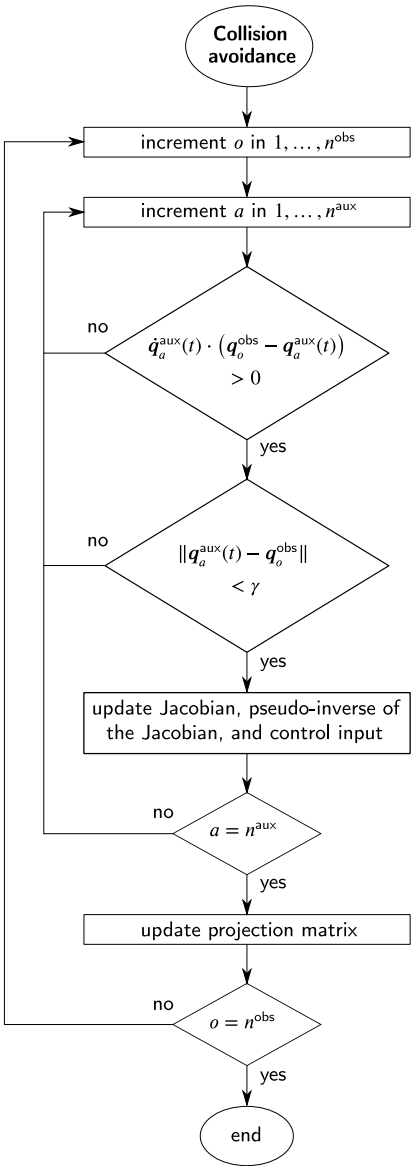


Figure 6.3: Flow chart of **Task 1** (collision avoidance) within the THC framework. The symbol · represents the scalar product.

Task 1 Avoiding collisions with obstacles, e.g., trees

Task 2 Maintaining a constant distance (ρ^{main}) from the main UAV and a safe distance ($\geq \rho^{\text{aux}}$) from other auxiliary UAVs

Task 3 Avoiding occlusion along the LoS that connects the auxiliary and main UAVs

Task 4 Moving towards the the next waypoint

These tasks are explained in the following sections.

TASK 1. COLLISION AVOIDANCE

Task 1 focuses on maintaining a safe distance from all obstacles. This task is triggered whenever the auxiliary UAV approaches an obstacle closer than a defined threshold γ, and its velocity vector points in the direction of that obstacle (see Figure 6.3).

The control input $\dot{\boldsymbol{q}}_{1,a}^{\text{aux}}(t)$ (i.e., the 3D velocity vector) of auxiliary UAV a corresponding to **Task 1** at time t is given by:

$$\dot{\boldsymbol{q}}_{1,a}^{\text{aux}}(t) = w_{1,a} J^{+}\left(\mathscr{Q}_{1,a}\left(\boldsymbol{q}_{a}^{\text{aux}}(t)\right)\right) e_{1,a}(t) \tag{6.2}$$

where $\mathscr{Q}_{1,a}\left(\boldsymbol{q}_{a}^{\text{aux}}(t)\right)$ denotes the forward kinematics function that maps the configuration space into the task space for **Task 1**, and is formulated by:

$$\mathscr{Q}_{1,a}\left(\boldsymbol{q}_{a}^{\text{aux}}(t)\right) = \| \boldsymbol{q}_{a}^{\text{aux}}(t) - \boldsymbol{q}^{\text{obs}} \| \tag{6.3}$$

Moreover, $J^{+}(\cdot)$ is the pseudo-inverse of the Jacobian matrix function with respect to $\boldsymbol{q}_{a}^{\text{aux}}(t)$, and $w_{1,a}$ is a tunable positive weight guaranteeing that the task error $e_{1,a}(t)$ for auxiliary UAV a will converge to zero. This task error is defined by:

$$e_{1,a}(t) = r^{\text{obs}} + \gamma - \mathscr{Q}_{1,a}\left(\boldsymbol{q}_{a}^{\text{aux}}(t)\right) \tag{6.4}$$

TASK 2. MAINTAINING A CONSTANT DISTANCE FROM THE MAIN UAV AND A SAFE DISTANCE FROM OTHER AUXILIARY UAVS

Task 2 ensures that a constant distance $\rho^{\text{main}} > 0$ is maintained between each auxiliary UAV and the main UAV (see Remark 12 for the criteria on choosing this costant value). In other words, all auxiliary UAVs move along the surface of a spherical shell of radius ρ^{main}, centered at the configuration $\boldsymbol{q}^{\text{main}}(t)$ of the main UAV (see Figure 6.4). Moreover, auxiliary UAVs are constrained to remain outside spherical caps (also illustrated in Figure 6.4), each centered on another auxiliary UAV and with radius ρ^{aux}. These constraints guarantee that, whenever **Task 2** is active, auxiliary UAVs maintain safe separation and avoid collisions with one another.

Note that remaining close to the main UAV must be handled entirely by the autonomous control system of the auxiliary UAVs through the THC framework, without assistance from the human operator in adapting the position of the main UAV. The corresponding control input $\dot{\boldsymbol{q}}_{2,a}^{\text{aux}}(t)$ (i.e., the contribution of **Task 2** to the 3D velocity vector of auxiliary UAV a) is given by:

$$\dot{\boldsymbol{q}}_{2,a}^{\text{aux}}(t) = w_{2,a} J^{+}\left(\mathscr{Q}_{2,a}\left(\boldsymbol{q}_{a}^{\text{aux}}(t)\right)\right) e_{2,a}(t) \tag{6.5}$$

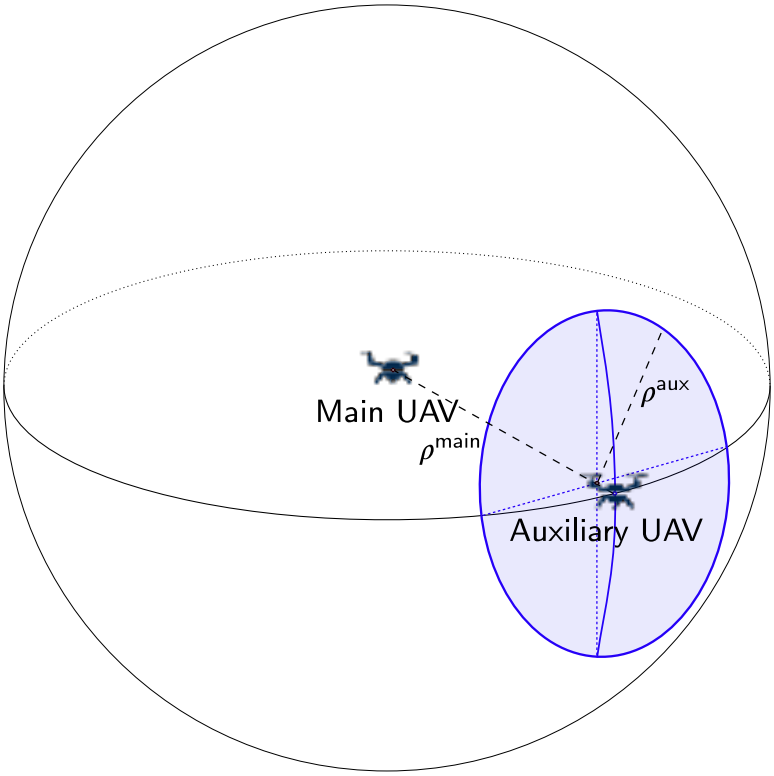


Figure 6.4: Illustration of a spherical cap of radius ρ^{aux} and centered on an auxiliary UAV, where this cap specifies a forbidden area for other auxiliary UAVs at the given time.

with $\mathscr{Q}_{2,a}\left(\boldsymbol{q}_a^{\text{aux}}(t)\right)$ the forward kinematics function mapping the configuration space into the task space for **Task 2**, formulated via:

$$\mathscr{Q}_{2,a}\left(\boldsymbol{q}_a^{\text{aux}}(t)\right) = \|\boldsymbol{q}_a^{\text{aux}}(t) - \boldsymbol{q}^{\text{main}}(t)\| \tag{6.6}$$

The corresponding task error for auxiliary UAV a given by:

$$e_{2,a}(t) = \rho^{\text{main}} - \mathscr{Q}_{2,a}\left(\boldsymbol{q}_a^{\text{aux}}(t)\right) \tag{6.7}$$

When $w_{2,a}$, which is a tunable gain used in (6.5), is chosen to be positive the task error is ensured to asymptotically converge to zero.

TASK 3. AVOIDING OCCLUSION ALONG THE LOS

The activation of the higher-priority tasks, particularly **Task 2**, ensures that auxiliary UAVs do not occlude each other's FoVs, as they maintain separation while flying on the same spherical shell around the main UAV.

After positioning themselves on a spherical shell around the main UAV, **Task 3** guarantees that the LoS—defined as the line segment that connects points $\boldsymbol{p}_a^{\text{aux}}(t)$ and $\boldsymbol{p}^{\text{main}}(t)$

(i.e., the points representing 3D position of, respectively, auxiliary UAV a and the main UAV at time t)—also remains free of occlusions potentially caused by obstacles.

For this, a minimum clearance distance γ should be sustained between the conic FoV of the auxiliary UAV and any obstacles. This is enforced by maintaining a safe separation between the following points: Point $\boldsymbol{p}_2(t)$ is one where the line perpendicular to the LoS, passing through the center $\boldsymbol{p}^{\text{obs}}$ of the obstacle, intersects the surface of the conic FoV of the auxiliary UAV, whereas point $\boldsymbol{p}_3(t)$ is the corresponding intersection point with the surface of the obstacle (see Figure 6.5 for a visual representation of the points).

The control input $\dot{\boldsymbol{q}}^{\text{aux}}_{3,a}(t)$, which includes the 3D velocity vector of auxiliary UAV a corresponding to **Task 3** at time t, is formulated through:

$$\dot{\boldsymbol{q}}^{\text{aux}}_{3,a}(t) = w_{3,a} J^{+}\left(\mathcal{Q}_{3,a}\left(\boldsymbol{q}^{\text{aux}}_{a}(t)\right)\right) e_{3,a}(t) \tag{6.8}$$

The expressions for the corresponding forward kinematics function $\mathcal{Q}_{3,a}\left(\boldsymbol{q}^{\text{aux}}_{a}(t)\right)$ and the task error $e_{3,a}(t)$, as well as conditions for tunable gain $w_{3,a}$ for ensuring the asymptotic convergence of the error to zero are discussed in detail in Theorem 1, Lemma 1, and Lemma 2.

TASK 4. MOVING TOWARDS THE NEXT WAYPOINT

In **Task 4**, each auxiliary UAV navigates towards its next designated waypoint (which has been determined as explained in Section 6.4.2) following a trajectory that is generated using PIC [21]. PIC operates over a receding time horizon and exploits a Monte Carlo sampling strategy to estimate an optimal control input trajectory through a cost-weighted averaging of perturbed trajectories.

The core idea is to induce stochastic perturbations into candidate control inputs and to evaluate their impact on the cost function. This allows to approximate the gradient of the cost function numerically, and to iteratively steer the system to lower-cost solutions. Unlike many classical gradient-based approaches, PIC does not require linearization and is thus suited for nonlinear systems. Moreover, thanks to its parallelizable nature, PIC can be executed in real time using available GPU resources.

The PIC problem for auxiliary UAV a, which at time t should move towards a waypoint of configuration $\boldsymbol{q}^{\text{wp}}_{a}(t)$, is formally stated as:

$$\begin{aligned}\min_{\tilde{\boldsymbol{u}}^{\text{aux}}_{4,a}(t)} \mathbb{E}\Bigg[\int_{t}^{t+T} \Big(&\left(\boldsymbol{q}^{\text{aux}}_{a}(\tau) - \boldsymbol{q}^{\text{wp}}_{a}(t)\right)^{\top} Q \left(\boldsymbol{q}^{\text{aux}}_{a}(\tau) - \boldsymbol{q}^{\text{wp}}_{a}(t)\right) \\ &+ \boldsymbol{u}^{\text{aux}\top}_{4,a}(t) R \boldsymbol{u}^{\text{aux}}_{4,a}(t) \Big) \mathrm{d}\tau \\ &+ \left(\boldsymbol{q}^{\text{aux}}_{a}(T) - \boldsymbol{q}^{\text{wp}}_{a}(t)\right)^{\top} P \left(\boldsymbol{q}^{\text{aux}}_{a}(T) - \boldsymbol{q}^{\text{wp}}_{a}(t)\right) \Bigg]\end{aligned} \tag{6.9}$$

In (6.9), T, which denotes the time horizon, is given and represents the desired duration for auxiliary UAV a to reach its assigned waypoint. The trajectory planning is guided by a stage cost comprising state and input penalties, weighted by matrices Q and R, respectively. These costs ensure that the UAV steadily reduces its distance to the waypoint, while also regulating fluctuations in the injected control inputs. Furthermore, a terminal cost weighted by matrix P promotes proximity to the waypoint at the end of the given

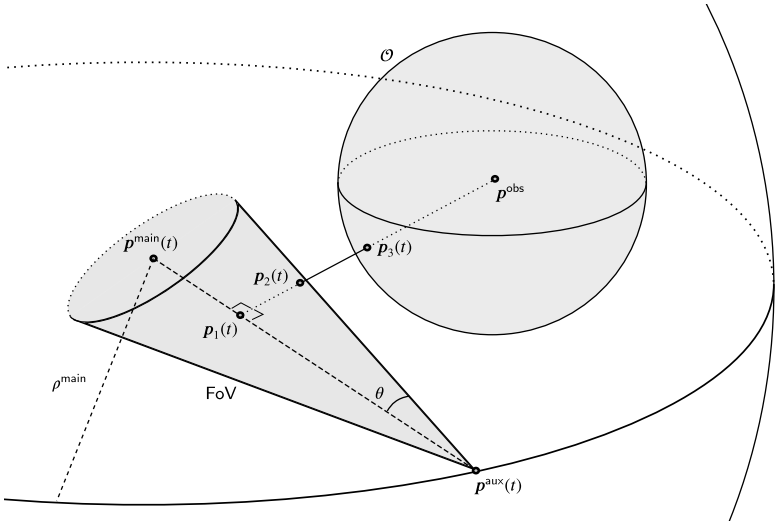


Figure 6.5: Illustration of occlusion avoidance along the LoS: In this figure, $\boldsymbol{p}^{\text{main}}(t)$ and $\boldsymbol{p}^{\text{aux}}(t)$ denote the 3D points representing the positions of, respectively, the main and the auxiliary UAVs at time t. Point $\boldsymbol{p}^{\text{obs}}$ is the center of obstacle $\mathcal{O}$. The imposed distance between the auxiliary and main UAVs is ρ^{main}, and θ represents the FoV angle of the camera of the auxiliary UAV. Points $\boldsymbol{p}_1(t)$, $\boldsymbol{p}_2(t)$, and $\boldsymbol{p}_3(t)$ are intersection points used in the geometric reasoning for the LoS occlusion avoidance.

time horizon. The control inputs are subject to stochastic perturbations by a Brownian random motion process. Consequently, the control input is perturbed from the nominal input $\boldsymbol{u}^{\text{aux}}_{4,a}(t)\,\mathrm{d}t$ by an additive noise. Due to this stochastic nature, the expected value operator $\mathbb{E}[\cdot]$ is used in estimation of the expected cumulative cost over the time horizon. Note that a tilde symbol is used for the optimization variable in (6.9) to indicate that the PIC optimization generates the trajectory of control input $\boldsymbol{u}^{\text{aux}}_{4,a}(t)$ corresponding to **Task 3** during $[t, t+T]$.

To compute (6.9), the PIC framework should predict the evolution of the state trajectory over the interval $\tau \in [t, t+T]$, which is done via:

$$\mathrm{d}\boldsymbol{q}^{\text{aux}}_{a}(\tau) = f\left(\boldsymbol{q}^{\text{aux}}_{a}(\tau)\right)\mathrm{d}\tau + g\left(\boldsymbol{q}^{\text{aux}}_{a}(\tau)\right)\cdot\left(\boldsymbol{u}^{\text{aux}}_{4,a}(\tau)\,\mathrm{d}\tau + s\,\mathrm{d}\boldsymbol{w}(\tau)\right) \tag{6.10}$$

where the functions $f(\cdot)$ and $g(\cdot)$ correspond to, respectively, the system state dynamics and control input mapping, as defined in (6.1). $w(t)$ is the Brownian random motion process, with diffusion coefficient $s \in \mathbb{R}$. $s\,\mathrm{d}\boldsymbol{w}(t)$ is the additive noise that perturbs the control input. The optimal control input trajectory determined by (6.9) is then used in (6.1) to compute the corresponding velocity trajectory $\tilde{\boldsymbol{q}}^{\text{aux}}_{4,a}(t)$ of the auxiliary UAV that is given to the UAV as the input to its actuators.

THC COMPOSITION USING PROJECTION MATRICES

The control input (i.e., 3D velocity vector) of each auxiliary UAV a at time t is determined via the THC framework, based on the four components $\dot{\boldsymbol{q}}^{\text{aux}}_{1,a}(t)$, $\dot{\boldsymbol{q}}^{\text{aux}}_{2,a}(t)$, $\dot{\boldsymbol{q}}^{\text{aux}}_{3,a}(t)$, and

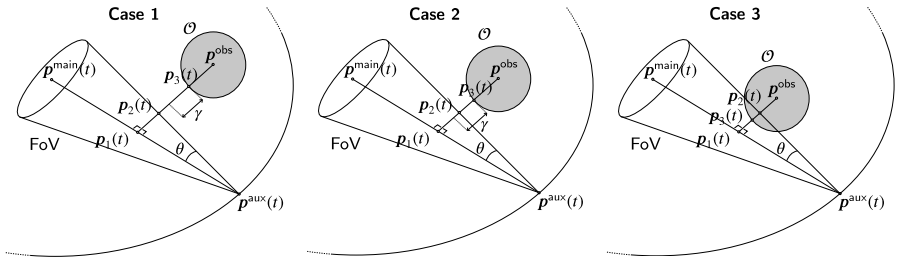


Figure 6.6: When an auxiliary UAV and the main UAV are in the proximity of an obstacle, three potential cases—illustrated in this figure—exist regarding the relative position of the FoV of the auxiliary UAV and the obstacle (specified via $\mathcal{O}$). These cases depend on the distance between the central axis of the conic FoV and the obstacle, compared to a threshold γ. Points $\boldsymbol{p}^{\text{main}}(t)$ and $\boldsymbol{p}^{\text{aux}}(t)$ in the figure correspond to, respectively, the main and the auxiliary UAVs, whereas point $\boldsymbol{p}^{\text{obs}}$ is the center of obstacle $\mathcal{O}$. The angle of FoV is θ and $\boldsymbol{p}_1(t)$, $\boldsymbol{p}_2(t)$, and $\boldsymbol{p}_3(t)$ are points where a perpendicular line segment to the axis of the conic FoV that crosses through $\boldsymbol{p}^{\text{obs}}$ intersects with, respectively, the axis, the surface of the conic FoV, and the obstacle.

$\dot{\boldsymbol{q}}^{\text{aux}}_{4,a}(t)$ computed via, respectively, (6.2), (6.5), (6.8), and plugging the solution of (6.9) into (6.1). To do so, the null spaces (i.e., the set of joint-space motions that do not affect the task) for the Jacobian matrices of all higher priority tasks must be determined. In the THC framework, projection matrices project the commands corresponding to lower-priority tasks to the null space of higher-priority tasks, ensuring that the lower-priority tasks never interfere with higher-priority tasks. The projection matrix into the null space of a higher priority task $i+1$ is defined via:

$$\begin{aligned}\mathscr{P}\left(\mathscr{Q}_{i,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right) = \mathscr{P}\left(\mathscr{Q}_{i-1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right) - \\ \left(J\left(\mathscr{Q}_{i-1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\mathscr{P}\left(\mathscr{Q}_{i-1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\right)^{+} \cdot \\ \left(J\left(\mathscr{Q}_{i-1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\mathscr{P}\left(\mathscr{Q}_{i-1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\right)\end{aligned} \tag{6.11}$$

where $\mathscr{P}(\cdot)$ is the projection operator. In our case with 4 tasks, we have $i \in \{2,3,4\}$. For $i=1$, i.e., for the highest-priority task within the THC framework, we have $\mathscr{P}\left(\mathscr{Q}_{1,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right) = I$, with I the identity matrix. Finally, the overall control input (i.e., overall 3D velocity vector) of auxiliary UAV a dictated by the THC framework at time t is computed by:

$$\begin{aligned}\dot{\boldsymbol{q}}^{\text{aux}}_a(t) = \dot{\boldsymbol{q}}^{\text{aux}}_{1,a}(t) + \mathscr{P}\left(\mathscr{Q}_{2,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\cdot\dot{\boldsymbol{q}}^{\text{aux}}_{2,a}(t) + \\ \mathscr{P}\left(\mathscr{Q}_{3,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\cdot\dot{\boldsymbol{q}}^{\text{aux}}_{3,a}(t) + \\ \mathscr{P}\left(\mathscr{Q}_{4,a}\left(\boldsymbol{q}^{\text{aux}}_a(t)\right)\right)\cdot\dot{\boldsymbol{q}}^{\text{aux}}_{4,a}(t)\end{aligned} \tag{6.12}$$

6.4.4. ACTION 3: LOCALIZING THE FIRE AND THE MAIN UAV

The positions of the fire and the main UAV can be estimated using computer vision techniques applied to the images that are captured by the auxiliary UAVs. In this chapter, we do not focus on this technical aspect and instead assume that these positions are already available to the auxiliary UAVs.

6.4.5. THEOREM: LOS OCCLUSION AVOIDANCE

In this section, we present a theorem demonstrating that, with the proposed control method, a single auxiliary UAV always maintain an occlusion-free view of the main UAV within their FoVs.

Note. *For notational simplicity, since the theorem concerns a single auxiliary UAV, we omit index a in the following statements (e.g., we use* $\boldsymbol{q}^{\text{aux}}(t)$ *instead of* $\boldsymbol{q}_a^{\text{aux}}(t)$*).*

While the theorem is stated for a single auxiliary UAV, it generalizes to multi-UAV systems. This is because **Task 1** (collision avoidance) has the highest priority in the THC architecture, and **Task 2** ensures that all auxiliary UAVs maintain a safe distance from each other. Consequently, when LoS occlusion avoidance is triggered via **Task 3**, each auxiliary UAV independently satisfies the required distance constraints with respect to obstacles.

Let $\boldsymbol{q}^{\text{aux}}(t), \boldsymbol{q}^{\text{main}}(t) \in \mathbb{R}^3$ denote the configurations of, respectively, the auxiliary and the main UAVs at time t, and $\boldsymbol{p}^{\text{aux}}(t)$ and $\boldsymbol{p}^{\text{main}}(t)$ be their corresponding points in 3D space. The line segment $\overline{\boldsymbol{p}^{\text{aux}}(t)\boldsymbol{p}^{\text{main}}(t)}$ (where $\overline{\boldsymbol{p}_{\text{a}}\boldsymbol{p}_{\text{b}}}$ generally denotes the segment connecting arbitrary points $\boldsymbol{p}_{\text{a}}$ and $\boldsymbol{p}_{\text{b}}$) represents the LoS, which also serves as the principal axis of the conic FoV of the auxiliary UAV, with vertex at $\boldsymbol{p}^{\text{aux}}(t)$ oriented towards the main UAV (see Figure 6.5). Moreover, let $\boldsymbol{p}^{\text{obs}}$ denote the center of the spherical obstacle $\mathcal{O}$. we define $\mathcal{V}^{\text{obs}}$ as the (continuous) set of all points comprising the outer surface and interior of the obstacle, and $\mathcal{V}^{\text{FoV}}(t)$ as the corresponding set for the FoV cone of the auxiliary UAV.

Theorem 1. *By incorporating **Task 3**, i.e., "avoiding obstacles on the LoS", into the THC framework, the FoV of the auxiliary UAV remains occlusion-free at all times. In other words, the FoV does not intersect with any obstacles, i.e.,* $\mathcal{V}^{\text{FoV}}(t) \cap \mathcal{V}^{\text{obs}} = \emptyset$*, and a minimum distance of at least* γ *is maintained between the FoV and the obstacle.*

Proof. Let $\boldsymbol{p}_1(t)$ be the closest point on the line segment $\overline{\boldsymbol{p}^{\text{aux}}(t)\boldsymbol{p}^{\text{main}}(t)}$ to the obstacle center $\boldsymbol{p}^{\text{obs}}$. Define $\overline{\boldsymbol{p}_1(t)\boldsymbol{p}^{\text{obs}}}$ as the line segment perpendicular to $\overline{\boldsymbol{p}^{\text{aux}}(t)\boldsymbol{p}^{\text{main}}(t)}$ that intersects both $\boldsymbol{p}_1(t)$ and $\boldsymbol{p}^{\text{obs}}$. Let $\boldsymbol{p}_2(t) \in \mathcal{V}^{\text{FoV}}(t)$ and $\boldsymbol{p}_3(t) \in \mathcal{V}^{\text{obs}}$ be the closest points to each other on $\overline{\boldsymbol{p}_1(t)\boldsymbol{p}^{\text{obs}}}$ (see Figure 6.5). Since obstacle $\mathcal{O}$ is convex-shaped (also see Remark 14), its intersection point $\boldsymbol{p}_3(t)$ with $\overline{\boldsymbol{p}_1(t)\boldsymbol{p}^{\text{obs}}}$ is unique.

Let $\boldsymbol{q}_1(t), \boldsymbol{q}_2(t), \boldsymbol{q}_3(t)$ be the configurations corresponding to the points $\boldsymbol{p}_1(t), \boldsymbol{p}_2(t), \boldsymbol{p}_3(t)$, respectively. Given a threshold value $\gamma > 0$, three distinct cases can arise (illustrated in Figures 6.6 and explained via Figure 6.2):

Case 1 $\|\boldsymbol{q}_2(t) - \boldsymbol{q}_1(t)\| + \gamma < \|\boldsymbol{q}_3(t) - \boldsymbol{q}_1(t)\|$

Case 2 $\|\boldsymbol{q}_2(t) - \boldsymbol{q}_1(t)\| < \|\boldsymbol{q}_3(t) - \boldsymbol{q}_1(t)\|$ and $\|\boldsymbol{q}_2(t) - \boldsymbol{q}_1(t)\| + \gamma > \|\boldsymbol{q}_3(t) - \boldsymbol{q}_1(t)\|$

Case 3 $\|\boldsymbol{q}_2(t) - \boldsymbol{q}_1(t)\| > \|\boldsymbol{q}_3(t) - \boldsymbol{q}_1(t)\|$

In Case 1, it is clear that $\mathcal{V}^{\text{FoV}}(t) \cap \mathcal{V}^{\text{obs}} = \emptyset$ and $\|\boldsymbol{q}_3(t) - \boldsymbol{q}_2(t)\| > \gamma$. Therefore, Theorem 1 holds.

In Case 2, while $\|\boldsymbol{q}_2(t)-\boldsymbol{q}_1(t)\| < \|\boldsymbol{q}_3(t)-\boldsymbol{q}_1(t)\|$ ensures that the FoV and the obstacle volumes are not intersecting (i.e., $\mathcal{V}^{\text{FoV}}(t) \cap \mathcal{V}^{\text{obs}} = \emptyset$), the safety margin condition is violated, since we have $\|\boldsymbol{q}_3(t)-\boldsymbol{q}_2(t)\| < \gamma$ (also see the middle plot in Figure 6.6).

In Case 3, neither condition holds, i.e., $\|\boldsymbol{q}_2(t)-\boldsymbol{q}_1(t)\| > \|\boldsymbol{q}_3(t)-\boldsymbol{q}_1(t)\|$ implies that $\mathcal{V}^{\text{FoV}}(t) \cap \mathcal{V}^{\text{obs}} \neq \emptyset$, meaning that $\boldsymbol{p}_2(t) \in \mathcal{V}^{\text{obs}}$ (see the third plot in Figure 6.6).

In both Case 2 and Case 3, **Task 3** (i.e., LoS occlusion avoidance) is activated. Due to the prioritized nature of THC, this implies that both **Task 1** (i.e., collision avoidance) and **Task 2** (i.e., maintaining distance from the main UAV) are already in effect.
Define the task error for **Task 3**, "avoiding occlusion along the LoS", by:

$$e_3(t) = \gamma - \|\boldsymbol{q}_3(t)-\boldsymbol{q}_2(t)\| \tag{6.13}$$

We will show in Lemma 1 that the following control input vector (including the 3D velocity vector of the auxiliary drone) will guarantee the convergence of the task error, i.e., $e_3(t) \to 0$. In other words it ensures that $\|\boldsymbol{q}_3(t)-\boldsymbol{q}_2(t)\| \to \gamma$, and thus the achievement of both occlusion-free visibility and the minimum safety distance γ:

$$\dot{\boldsymbol{q}}_3^{\text{aux}}(t) = w_3 J^{+}\left(\|\boldsymbol{q}_3(t)-\boldsymbol{q}_2(t)\|\right) e_3(t) \tag{6.14}$$

where $J^{+}(\cdot)$ is the pseudo-inverse of the Jacobian of the given function with respect to $\boldsymbol{q}^{\text{aux}}(t)$, and w_3 is a tunable positive weight. □

Remark 13. *Theorem 1 is stated for a single obstacle. However, our control architecture also handles scenarios with multiple obstacles: The contributions of multiple obstacles are addressed in the THC framework by incorporating sub-tasks of **Task 1** and **Task 3** into an extended projection matrix. In this way, each obstacle is treated independently, but all are avoided simultaneously through coordinated task prioritization.*

Remark 14. *Theorem 1 holds for convex-shaped obstacles. Nevertheless, in the case of non-convex obstacles, their convex hulls can be computed and used as approximations. Our THC approach can then be applied to these convex hulls in the same manner. Therefore, to ensure the validity of Theorem 1, obstacles should be represented by their convex hulls.*

Next we give Lemma 1, which provides the control input vector that ensures an occlusion-free LoS for all n^{aux} auxiliary UAVs in vicinity of a static obstacle.

Lemma 1. ***Control inputs for Task 3 "avoiding occlusion along the LoS", guaranteeing convergence of the task error to zero:*** *Let the configuration of the entire system at time t, including all auxiliary UAVs indexed $a = 1,\ldots,n^{\text{aux}}$, the main UAV, and an obstacle $\mathcal{O}$ that is currently in the vicinity of the main UAV, be defined as:*

$$\boldsymbol{q}^{\text{tot}}(t) = \left[\boldsymbol{q}^{\text{main}\top}(t), \boldsymbol{q}_1^{\text{aux}\top}(t), \ldots, \boldsymbol{q}_{n^{\text{aux}}}^{\text{aux}\top}(t), \boldsymbol{q}^{\text{obs}\top}\right]^{\top}$$

Moreover, let $\mathcal{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)$ denote the forward kinematics function that maps this configuration space into the task space via:

$$\mathcal{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right) = \left[\|\boldsymbol{q}_{3,1}(t)-\boldsymbol{q}_{2,1}(t)\|, \ldots, \|\boldsymbol{q}_{3,n^{\text{aux}}}(t)-\boldsymbol{q}_{2,n^{\text{aux}}}(t)\|\right]^{\top} \tag{6.15}$$

where $\boldsymbol{q}_{2,a}(t)$ and $\boldsymbol{q}_{3,a}(t)$ generalize the notations for the points $\boldsymbol{q}_3(t)$ and $\boldsymbol{q}_2(t)$ introduced in Figure 6.5, and the second subscript 'a' corresponds to each auxiliary UAV.

*The total task error $\boldsymbol{e}_3^{\text{tot}}(t) \in \mathbb{R}^{n^{\text{aux}}}$ for **Task 3** (avoiding occlusion along the LoS) is defined as the difference between the desired task vector $\boldsymbol{\Gamma} \in \mathbb{R}^{n^{\text{aux}}}$, with all elements equal to the threshold distance γ, and the current task-space vector $\mathscr{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right) \in \mathbb{R}^{n^{\text{aux}}}$. Thus, we have:*

$$\boldsymbol{e}_3^{\text{tot}}(t) = \boldsymbol{\Gamma} - \mathscr{Q}\left(\boldsymbol{q}^{\text{tot}}(t)\right) \tag{6.16}$$

*To guarantee that a minimum distance of at least γ is maintained between the FoV of each auxiliary UAV and obstacle $\mathscr{O}$, we define the total control input vector $\boldsymbol{u}_3^{\text{tot}}(t) \in \mathbb{R}^{3n^{\text{aux}}}$, corresponding to the velocities of all auxiliary UAVs for **Task 3** via:*

$$\dot{\boldsymbol{q}}_3^{\text{tot}}(t) = J^{+}\left(\mathscr{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)\right) W_3 \boldsymbol{e}_3^{\text{tot}}(t) \tag{6.17}$$

where $J^{+}(\cdot)$ is the pseudo-inverse of the Jacobian of the forward kinematics function, and W_3 is a tunable gain matrix of size $n^{\text{aux}} \times n^{\text{aux}}$. To ensure convergence of the task error, matrix W_3 must be chosen such that all its eigenvalues have positive real parts. In practical applications, this condition can be satisfied by selecting W_3 as a diagonal matrix with strictly positive diagonal elements.

Proof. We begin by deriving the dynamics of the total task error $\boldsymbol{e}_3^{\text{tot}}(t)$. Using the chain rule, we have:

$$\begin{aligned}
\dot{\boldsymbol{e}}_3^{\text{tot}}(t) &= -\dot{\mathscr{Q}}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right) \\
&= -J\left(\mathscr{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)\right) \dot{\boldsymbol{q}}_3^{\text{tot}}(t) \\
&= -J\left(\mathscr{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)\right) J^{+}\left(\mathscr{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)\right) W_3 \boldsymbol{e}_3^{\text{tot}}(t) \\
&= -W_3 \boldsymbol{e}_3^{\text{tot}}(t)
\end{aligned} \tag{6.18}$$

Since $-W_3$ is a tunable gain matrix that can be designed to be stable (i.e., all its eigenvalues have negative real parts, e.g., when designed as a diagonal matrix with strictly positive diagonal elements), the above linear error dynamics is exponentially stable. Therefore, the task error $\boldsymbol{e}_3^{\text{tot}}(t)$ asymptotically converges to zero, ensuring that a minimum distance of at least γ is maintained between the FoV of each auxiliary UAV and obstacle $\mathscr{O}$.

Note. *Defining a positive definite candidate Lyapunov function via:*

$$V\left(\boldsymbol{e}_3^{\text{tot}}(t)\right) = \boldsymbol{e}_3^{\text{tot}}(t)^{\top} L \boldsymbol{e}_3^{\text{tot}}(t) \tag{6.19}$$

with 'L' a positive definite matrix, and taking the derivative of this function along the trajectories of the total error dynamics (replacing $\dot{\boldsymbol{e}}_3^{\text{tot}}(t)$ from (6.18)) yields:

$$\dot{V}\left(\boldsymbol{e}_3^{\text{tot}}(t)\right) = -2\boldsymbol{e}_3^{\text{tot}}(t)^{\top} L W_3 \boldsymbol{e}_3^{\text{tot}}(t) \tag{6.20}$$

One can show that for a negative definite Lyapunov derivative function, the following must hold:

$$LW_3 + W_3^{\top} L \succ 0 \tag{6.21}$$

which is true if $-W_3$ is a stable matrix, i.e., all eigenvalues of gain matrix W_3 must only have real positive parts.

□

Note. *Since the expressions that define the control inputs for tasks 1, 2, 3 (see (6.2) and (6.5) and (6.8)) follow the same rationale, the convergence of the corresponding task errors to zero can be ensured in a similar way.*

Next, we present Lemma 2, which provides the expression of the forward kinematics function $\mathcal{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)$, which in Lemma 1 was given as an expression based on virtual point $\boldsymbol{p}_2(t)$ and $\boldsymbol{p}_3(t)$.

Lemma 2. ***Forward kinematics function for Task 3 as a function of the total configuration of the system:*** *Function $\mathcal{Q}_3\left(\boldsymbol{q}^{\text{tot}}(t)\right)$, which has been expressed in (6.15) based on virtual points $\boldsymbol{p}_{2,a}(t)$ and $\boldsymbol{p}_{3,a}(t)$ with non-measurable configurations $\boldsymbol{q}_{1,a}(t)$ and $\boldsymbol{q}_{2,a}(t)$ for auxiliary UAVs $a \in \{1,\ldots,n^{\text{aux}}\}$, can be obtained based on measurable configurations (i.e., total configuration $\boldsymbol{q}^{\text{tot}}(t)$ of the system). For $a = 1,\ldots,n^{\text{aux}}$ we have:*

$$\boldsymbol{q}_{2,a}(t) = \boldsymbol{q}^{\text{obs}} + \lambda_a(t)\vec{\boldsymbol{u}}_a(t) \tag{6.22}$$

$$\boldsymbol{q}_{3,a}(t) = \boldsymbol{q}^{\text{obs}} + r^{\text{obs}}\vec{\boldsymbol{u}}_a(t) \tag{6.23}$$

where $\lambda_a(t)$, which is the distance between points $\boldsymbol{p}^{\text{obs}}$ and $\boldsymbol{p}_{2,a}(t)$, is given by:

$$\lambda_a(t) = \|\boldsymbol{q}_{1,a}(t) - \boldsymbol{q}^{\text{obs}}\| - \|\boldsymbol{q}_{1,a}(t) - \boldsymbol{q}_a^{\text{aux}}(t)\|\tan\theta \tag{6.24}$$

The coordinates $\boldsymbol{q}_{1,a}(t)$ of point $\boldsymbol{p}_{1,a}(t)$, i.e., the foot point of the perpendicular line from $\boldsymbol{p}^{\text{obs}}$ to line segment $\overline{\boldsymbol{p}_a^{\text{aux}}(t)\boldsymbol{p}^{\text{main}}(t)}$ (see Figure 6.5), is obtained using the following scalar projection function:

$$\sigma_a(t) = \frac{\eta_a(t)}{\|\boldsymbol{q}_a^{\text{aux}}(t) - \boldsymbol{q}^{\text{main}}(t)\|^2} \tag{6.25}$$

where

$$\begin{aligned}\eta_a(t) = &\left(x^{\text{obs}} - x^{\text{main}}(t)\right)\left(x_a^{\text{aux}}(t) - x^{\text{main}}(t)\right) + \\ &\left(y^{\text{obs}} - y^{\text{main}}(t)\right)\left(y_a^{\text{aux}}(t) - y^{\text{main}}(t)\right) + \\ &\left(z^{\text{obs}} - z^{\text{main}}(t)\right)\left(z_a^{\text{aux}}(t) - z^{\text{main}}(t)\right)\end{aligned} \tag{6.26}$$

Then we have:

$$\boldsymbol{q}_{1,a}(t) = \boldsymbol{q}_a^{\text{aux}}(t) + \sigma_a(t)\left(\boldsymbol{q}_a^{\text{aux}}(t) - \boldsymbol{q}^{\text{main}}(t)\right) \tag{6.27}$$

Note that for the coordinates of points $\boldsymbol{p}_{2,a}(t)$ and $\boldsymbol{p}_{3,a}(t)$, which are located at distances, respectively, $\lambda_a(t)$ and r^{obs} beyond point $\boldsymbol{p}^{\text{obs}}$ in the direction from $\boldsymbol{p}^{\text{obs}}$ to $\boldsymbol{p}_{1,a}(t)$ (see Figure 6.5), the corresponding unit vector $\vec{\boldsymbol{u}}_a(t)$ is given by:

$$\vec{\boldsymbol{u}}_a(t) = \frac{\boldsymbol{q}_{1,a}(t) - \boldsymbol{q}^{\text{obs}}}{\sqrt{\|\boldsymbol{q}_{1,a}(t) - \boldsymbol{q}^{\text{obs}}\|^2}} \tag{6.28}$$

Proof. The proof of Lemma 2 follows directly from basic geometric constructions and trigonometric relationships among the points and line segments. □

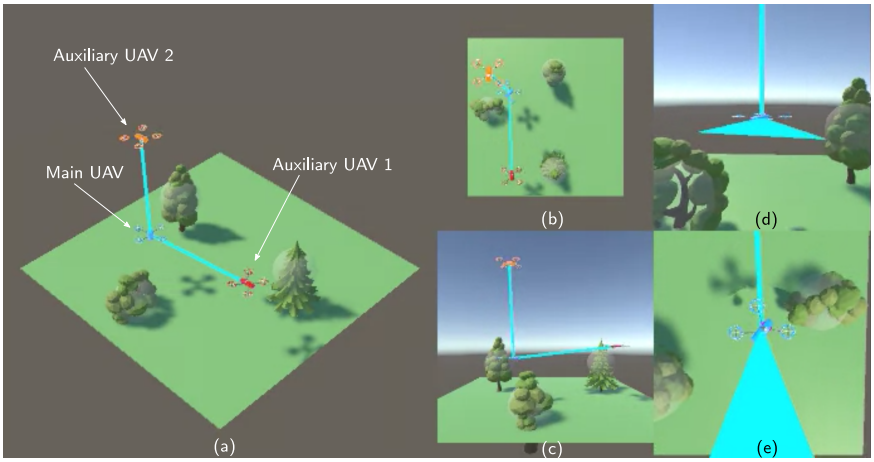


Figure 6.7: Illustration of the scenario simulated in Unity: From left to right, the 3D view of the simulated outdoor environment (a), the top view (b) and the side view (c) of the environment, and the views of the cameras of the auxiliary UAV 1 (d) and the auxiliary UAV 2 (e). The main UAV has cyan color, the auxiliary UAVs have red and orange colors.

6.5. SIMULATIONS AND RESULTS

This section presents a case study conducted to evaluate the proposed approach described in Section 6.4, detailing the implementation of each task in the THC framework and discussing the corresponding results.

6.5.1. SETUP

All algorithms, including the THC controller used in this case study, were implemented in MATLAB (R2021a version), and 3D simulations of the UAVs operating in outdoor environments were developed in Unity (2021 version). An overview of the simulated environments is provided in Figure 6.7. All experiments were run on a PC equipped with an Intel Core i7 processor (4 cores, 1.80–2.30 GHz) and 16 GB of RAM.

The simulations were conducted within discrete time with a sampling time of 0.1 s setup involved 2 auxiliary UAVs operating in a cubic outdoor environment of dimensions $22 \times 22 \times 22$ m^3, containing between 1 and 3 spherical obstacles. The obstacles represented trees or other UAVs and had radii ranging from 1 m to 2 m. The scenarios included 3 to 9 waypoints, where the time for the UAVs to move from one waypoint to the next one was 10 s. The trajectories of the main UAV were provided to the auxiliary UAVs directly in the computer simulations. However, in real-life deployments, this information is typically obtained through real-time image processing.

The angle of the FoV of the onboard cameras of the auxiliary UAVs was set within the range $\frac{\pi}{9}$ rad to $\frac{\pi}{12}$ rad. The desired distance between each auxiliary UAV and the main UAV (i.e., ρ^{main}) was set to 8 m. The UAVs were modeled as double integrators, with velocity used as the input injected to their actuators.

Comparisons were conducted across 6 different scenarios:

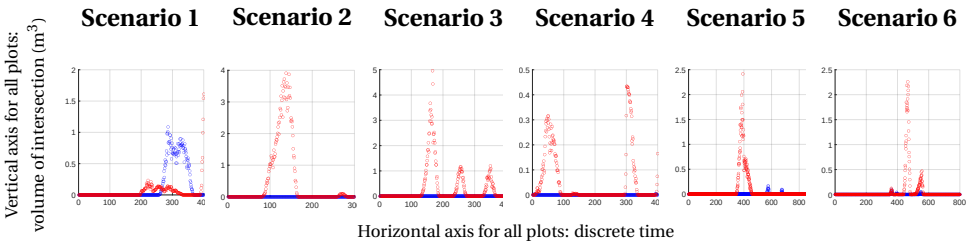


Figure 6.8: Comparison of the evolution of the intersecting volume between the FoVs and any obstacles, corresponding to **Comparison 1**, across all scenarios. The plots for when **Task 3** is activated versus not activated are shown in, respectively, blue and red.

Scenario #	**Task 3** inactive	**Task 3** active	Improvement
Scenario 1	40.25%	**30.25%**	-10.00%
Scenario 2	44.00%	**0.00**	-44.00%
Scenario 3	41.50%	**1.50%**	-40.00%
Scenario 4	44.75%	**12.75%**	-32.00%
Scenario 5	13.56%	**8.56%**	-5.00%
Scenario 6	17.62%	**7.75%**	-9.87%

Table 6.1: Comparing the percentage of the total mission time during which the FoV of at least one auxiliary UAV was occluded by obstacles, across all scenarios and corresponding to **Comparison 1**—when **Task 3** for LoS occlusion avoidance was versus was not active. The bold values indicate the best outcome, i.e., smaller interval with LoS occlusion.

Scenario 1 Static tree obstacles: Trees were modeled as stationary spherical obstacles for auxiliary UAVs.

Scenario 2 Flying obstacles: Other UAVs flying in the shared airspace were modeled as dynamic obstacles.

Scenario 3 Tall ground obstacles: Trees or structures taller than the altitude of the UAV and thus closer to it compared to **Scenario 1** were included.

Scenario 4 Task switch for main UAV: The main UAV dynamically switched between reachability to manipulability tasks, requiring high adaptation from the auxiliary UAVs.

Scenario 5 Figure-eight maneuver: The main UAV moved in a figure-eight trajectory around a tree, posing a challenge for continuous visibility.

Scenario 6 Slalom maneuver: The main UAV performed a slalom between trees, requiring precise and responsive behavior from the auxiliary UAVs to maintain an occlusion-free LoS.

6.5.2. SIMULATION RESULTS

To evaluate the performance of our proposed THC framework, we performed an ablation study comparing an implementation of the entire THC framework against versions

where specific components of the THC framework were disabled. We additionally compared our optimal control strategy based on PIC against a widely used conventional control approach based on PID control.

Our evaluation was performed across three categories of comparisons, for each of the six scenarios explained above (i.e., **Scenario 1**–**Scenario 6**):

Comparison 1 Activating versus deactivating **Task 3**, i.e., LoS occlusion avoidance

Comparison 2 Activating versus deactivating **Task 2**, i.e., distance maintenance among different UAVs

Comparison 3 Activating versus deactivating the use of PIC in **Task 4**, i.e., waypoint-reaching task

We next present the results of the case study.

Table 6.1 shows the percentage of the total mission time during which the FoVs of the auxiliary UAVs were occluded by obstacles, corresponding to **Comparison 1**, across all scenarios.

Figure 6.8 illustrates how the intersecting volume between the FoVs and any obstacles evolves over time, corresponding to **Comparison 1**, across all scenarios.

Figure 6.9 presents the distances between the main UAV and each auxiliary UAV over time, corresponding to **Comparison 2**, across all scenarios.

Tables 6.2 and 6.3 show the maximum and average deviation, respectively, from the desired distance between the main and the auxiliary UAVs, for each auxiliary UAV corresponding to **Comparison 2**, across all scenarios.

Table 6.4 compares the total path length traveled by each auxiliary UAV, corresponding to **Comparison 3**, across all scenarios.

Lastly, Table 6.5 the percentage of the total mission time during which the FoVs of the auxiliary UAVs were occluded by obstacles, corresponding to **Comparison 3**, across all scenarios.

6.6. DISCUSSION OF THE RESULTS

In this section, we analyze and compare the performance of the proposed THC framework, based on the results presented in Section 6.5.2.

From Table 6.1, it is evident that the proposed THC framework maintains an unobstructed view of the main UAV for a larger portion of the mission time. Note that violation of the unoccluded view still occurs when **Task 3** is activated because of the nature of THC: when the capabilities of the system only allow to execute the higher priority tasks, i.e., **Task 1** and **Task 2**, the view can be occluded. Activating the LoS occlusion avoidance task leads to improvements across all scenarios, with gains reaching up to 44.00%. This improvement is particularly significant in **Scenario 2** and **Scenario 3**, where the obstacles are more critical due to their proximity to the trajectories of the UAVs.

Likewise, Figure 6.8 shows that significant volumes of intersection between the FoV and obstacles occur in **Scenario 1** when **Task 3**, the LoS occlusion avoidance, is active (see the blue curves), due to a wider configuration of the obstacles. In the remaining five scenarios, the intersection volume remains negligible. Without the LoS occlusion

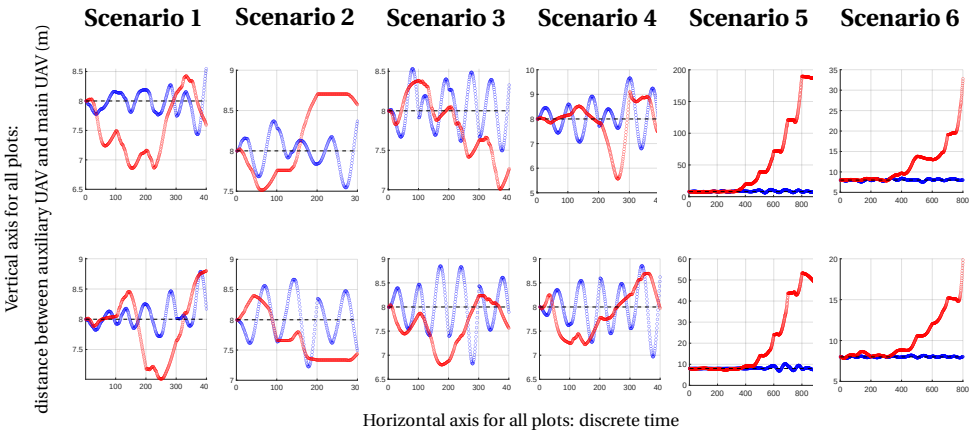


Figure 6.9: Comparison of the evolution of the distances between the main UAV and auxiliary UAV 1 (top plots) and between the main UAV and auxiliary UAV 2 (bottom plots), corresponding to **Comparison 2**, across all scenarios. The plots for when **Task 2** is activated versus not activated are shown in, respectively, blue and red. The black dashed lines indicate the desired distance ρ^{main} between the main UAV and each auxiliary UAV.

avoidance task (see the red curves), the intersection volume consistently reaches higher values in all six scenarios, reinforcing the importance of **Task 3**.

Regarding the distance maintenance task, i.e., **Task 2**, Figure 6.9 demonstrates that activating the task generally improves the ability of auxiliary UAVs to maintain the desired distance of ρ^{main} from the main UAV. This effect is especially pronounced in **Scenario 5** and **Scenario 6**, where the distances of the UAVs tend to diverge. Without the distance maintenance task, auxiliary UAVs focus solely on preserving an unobstructed view and tend to drift away, with no mechanism encouraging them to return to a closer position relative to the main UAV.

Supporting this observation, Table 6.2 shows that the maximum distance errors are typically lower when the distance maintenance task is activated. Exceptions are seen in only two cases, where higher peak errors occur due to oscillatory behavior induced by the control input responsible for distance regulation. This is also visible in Figure 6.9) with more noticeable oscillations. Table 6.3 confirm that the average distance errors are consistently lower when the distance maintenance task is included. This implies an overall improved tracking performance due to **Task 2**.

Finally, we evaluate the impact of using PIC versus a traditional PID controller for the task of reaching the next waypoint (**Task 4**). Table 6.4 shows that, in general, PIC generates (close-to) optimal trajectories (i.e., shorter paths to target waypoints), achieving path length reduction of up to 39.58% compared to PID. In a few exceptions, e.g., in **Scenario 3** and **Scenario 4**, PIC does not outperform PID. This is because in those scenarios, the second auxiliary UAV encounters fewer obstacles along its trajectory, where a more sophisticated control strategy, such as PIC, does not provide added values (as both controllers yield relatively short paths). In **Scenario 5**, the more challenging trajectory of the main UAV, closely turning around obstacles, forces the auxiliary UAVs to make larger movements to maintain visibility for the main UAV. This results in longer paths

Scenario #	Auxiliary UAV #	**Task 2** inactive	**Task 2** active	Improvement
Scenario 1	Auxiliary UAV 1	1.14 m	**0.57 m**	-50.00%
	Auxiliary UAV 2	0.99 m	**0.79 m**	-20.20%
Scenario 2	Auxiliary UAV 1	0.71 m	**0.46 m**	-35.21%
	Auxiliary UAV 2	**0.67 m**	0.78 m	+16.42%
Scenario 3	Auxiliary UAV 1	0.99 m	**0.53 m**	-46.46%
	Auxiliary UAV 2	1.20 m	**1.18 m**	-1.67%
Scenario 4	Auxiliary UAV 1	2.43 m	**1.66 m**	-31.69%
	Auxiliary UAV 2	**0.78 m**	1.03 m	+32.05%
Scenario 5	Auxiliary UAV 1	181.50 m	**3.24 m**	-98.21%
	Auxiliary UAV 2	45.18 m	**2.24 m**	-95.04%
Scenario 6	Auxiliary UAV 1	24.77 m	**0.50 m**	-97.98%
	Auxiliary UAV 2	11.86 m	**0.19 m**	-98.40%

Table 6.2: Comparing the maximum deviation in meter from the desired distance between the main and the auxiliary UAVs, per auxiliary UAV and across all scenarios, corresponding to **Comparison 2**—when **Task 2** for distance maintenance among different UAVs was versus was not active. The bold values indicate the smaller maximum deviations.

and highlights the need to for a balanced performance trade-off between maintaining visibility and minimizing the path length in complex scenarios.

Lastly, Table 6.5 further illustrates that PIC improves performance by reducing the time during which the FoV intersects with obstacles. An exception is found in **Scenario 1**, where the total intersection time is slightly smaller with PID. A breakdown reveals that with PID, the two auxiliary UAVs experience 73 and 88 time steps of intersection, respectively, while with PIC the values are 5 and 121, yielding a lower total number of intersections, as one can see from Table 6.6. This suggests that PIC more suitably distributes the effort of maintaining visibility across the auxiliary UAVs.

6.7. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

In this chapter, we proposed a task hierarchical control (THC) framework for a multi-UAV system. This framework integrates the best viewpoints for consistent monitoring of a main UAV, line-of-sight (LoS) occlusion avoidance guaranteeing a consistently occlusion-free view of the main UAV, and path integral control (PIC) for optimal trajectory planning of auxiliary UAVs. The results of a case study performed via computer simulations with one main and two auxiliary UAVs demonstrated that the LoS occlusion avoidance can bring up to 44.0% improvement in maintaining visibility, while with PIC the generated trajectory length improves by up to 39.2%.

Although the work is motivated by the XPRIZE Wildfire Challenge [361], which focuses on fire suppression using UAV systems, our framework is adaptable to a wide range of real-life multi-agent monitoring and coordination applications.

To enable real-time performance, future work will focus on implementing GPU acceleration for the PIC component of our THC framework.

Alternatively, the trajectory optimization task in THC can be implemented and compared to its current structure, using model predictive control (which allows for system-

Scenario #	Auxiliary UAV #	**Task 2** inactive	**Task 2** active	Improvement
Scenario 1	Auxiliary UAV 1	0.55 m	**0.16 m**	-70.91%
	Auxiliary UAV 2	0.40 m	**0.17 m**	-57.50%
Scenario 2	Auxiliary UAV 1	0.42 m	**0.16 m**	-61.90%
	Auxiliary UAV 2	0.43 m	**0.31 m**	-27.91%
Scenario 3	Auxiliary UAV 1	0.33 m	**0.23 m**	-30.30%
	Auxiliary UAV 2	0.44 m	**0.39 m**	-11.36%
Scenario 4	Auxiliary UAV 1	0.56 m	**0.53 m**	-5.36%
	Auxiliary UAV 2	0.40 m	**0.35 m**	-12.50%
Scenario 5	Auxiliary UAV 1	48.09 m	**0.74 m**	-98.46%
	Auxiliary UAV 2	12.38 m	**0.31 m**	-97.50%
Scenario 6	Auxiliary UAV 1	3.76 m	**0.16 m**	-95.74%
	Auxiliary UAV 2	2.23 m	**0.06 m**	-97.31%

Table 6.3: Comparing the average deviation in meter from the desired distance between the main and the auxiliary UAVs, per auxiliary UAV and across all scenarios, corresponding to **Comparison 2**—when **Task 2** for distance maintenance among different UAVs was versus was not active. The bold values indicate the smaller average deviations.

atic incorporation of constraints), as explored in prior work [1], [41].

We also plan to extend our work to real-world deployments, validating the effectiveness of the proposed THC framework beyond computer-based simulations. In scenarios that involve non-spherical obstacles, complex object geometries can be approximated using multiple spheres to represent their 3D volume (see [193] for inspiration).

Finally, future research will address challenges in handling partially or completely known obstacle information, including the position, size, and number of obstacles.

APPENDIX: TABLE OF NOTATIONS

This appendix contains the table with most used and important mathematical notations.

Scenario #	Auxiliary UAV #	Task 4 using PID	Task 4 using PIC	Improvement
Scenario 1	Auxiliary UAV 1	25.71 m	**20.57 m**	-19.99%
	Auxiliary UAV 2	32.76 m	**22.06 m**	-32.66%
Scenario 2	Auxiliary UAV 1	17.76 m	**16.31 m**	-8.16%
	Auxiliary UAV 2	23.55 m	**18.89 m**	-19.79%
Scenario 3	Auxiliary UAV 1	27.31 m	**22.44 m**	-17.83%
	Auxiliary UAV 2	**22.58 m**	24.98 m	+10.63%
Scenario 4	Auxiliary UAV 1	48.16 m	**29.29 m**	-39.18%
	Auxiliary UAV 2	**23.07 m**	27.32 m	+18.42%
Scenario 5	Auxiliary UAV 1	**89.06 m**	105.11 m	+18.02%
	Auxiliary UAV 2	**55.20 m**	68.57 m	+24.22%
Scenario 6	Auxiliary UAV 1	70.07 m	**63.35 m**	-9.59%
	Auxiliary UAV 2	52.52 m	**46.45 m**	-11.56%

Table 6.4: Comparing the total path length in meter traveled by each auxiliary UAV, per auxiliary UAV and across all scenarios, corresponding to **Comparison 3**—when **Task 4** for using PIC for waypoint-reaching was active versus was not active. The bold values indicate the shorter paths.

Scenario #	Task 4 using PID	Task 4 using PIC	Improvement
Scenario 1	**27.00%**	30.25%	+3.25%
Scenario 2	4.67%	**0.00**	-4.67%
Scenario 3	5.25%	**1.50%**	-3.75%
Scenario 4	34.50%	**12.75%**	-21.75%
Scenario 5	10.80%	**8.56%**	-2.24%
Scenario 6	10.75%	**7.75%**	-3.00%

Table 6.5: Comparing the percentage of the total mission time during which the FoV of at least one auxiliary UAV was occluded by obstacles, across all scenarios and corresponding to **Comparison 3**—when **Task 4** for using PIC for waypoint-reaching was active versus was not active. The bold values indicate the best outcome, i.e., smaller interval with LoS occlusion.

Scenario #	Task 4 using PID	Task 4 using PIC	Improvement
Scenario 1	13.25%	**1.25%**	-12.00%
Scenario 2	**0.00**	**0.00**	0.00
Scenario 3	**0.00**	**0.00**	0.00
Scenario 4	**0.00**	**0.00**	0.00
Scenario 5	**0.00**	1.67%	+1.67%
Scenario 6	**0.00**	**0.00**	0.00

Table 6.6: Comparing the percentage of the total mission time during which the FoVs of both auxiliary UAVs were occluded by obstacles, across all scenarios and corresponding to **Comparison 3**—when **Task 4** for using PIC for waypoint-reaching was active versus was not active. The bold values indicate the best outcome, i.e., smaller interval with LoS occlusion.

Table 6.7: Table of frequently used mathematical notations

Notation	Description
e_i	Error of task i of THC
$\boldsymbol{e}_3^{\text{tot}}$	Total error of task 3 of THC
$\dot{\boldsymbol{e}}_3^{\text{tot}}$	Time derivative of $\boldsymbol{e}_3^{\text{tot}}$
J	Jacobian matrix function
J^+	Pseudo-inverse of J
n^{aux}	Number of auxiliary UAVs
n^{obs}	Number of obstacles
$\mathscr{O}$	Generic obstacle
$\boldsymbol{p}$	A point in the 3D space
$\boldsymbol{p}_a^{\text{aux}}(t)$	Position of auxiliary UAV a at time t
$\boldsymbol{p}^{\text{main}}(t)$	Position of main UAV at time t
$\boldsymbol{p}^{\text{obs}}$	Position of the center of the obstacle
$\boldsymbol{p}_a^{\text{wp}}(t)$	Position of waypoint for auxiliary UAV a at time t
$\mathscr{P}$	Projection operator
$\overline{\boldsymbol{p}_{\text{a}}\boldsymbol{p}_{\text{b}}}$	Line segment connecting points $\boldsymbol{p}_{\text{a}}$ and $\boldsymbol{p}_{\text{b}}$
$\boldsymbol{q}_a^{\text{aux}}(t)$	Configuration of auxiliary UAV a at time t
$\dot{\boldsymbol{q}}_{i,a}^{\text{aux}}$	Velocity of auxiliary UAV a at time t corresponding to task i of THC
$\boldsymbol{q}^{\text{main}}(t)$	Configuration of main UAV at time t
$\boldsymbol{q}^{\text{obs}}$	Configuration of the obstacle
$\boldsymbol{q}^{\text{tot}}(t)$	Total configuration of main UAV, auxiliary UAVs and obstacle at time t
$\boldsymbol{q}_a^{\text{wp}}(t)$	Configuration of waypoint for auxiliary UAV a at time t
$\mathscr{Q}i,a$	Forward kinematics function for auxiliary UAV a corresponding to task i of THC
r^{obs}	Radius of the obstacle
t	Continuous time variable
$\boldsymbol{u}_{i,a}^{\text{aux}}(t)$	Control input of auxiliary UAV a at time t corresponding to task i of THC
$\boldsymbol{u}_3^{\text{tot}}$	Total control input vector of task 3 of THC
$\mathscr{V}$	Volume of a solid object
$x_a^{\text{aux}}(t)$	x coordinate of $\boldsymbol{p}_a^{\text{aux}}(t)$
$x^{\text{main}}(t)$	x coordinate of $\boldsymbol{p}^{\text{main}}(t)$
x^{obs}	x coordinate of $\boldsymbol{p}^{\text{obs}}$
$y_a^{\text{aux}}(t)$	y coordinate of $\boldsymbol{p}_a^{\text{aux}}(t)$
$y^{\text{main}}(t)$	y coordinate of $\boldsymbol{p}^{\text{main}}(t)$
y^{obs}	y coordinate of $\boldsymbol{p}^{\text{obs}}$
$z_a^{\text{aux}}(t)$	z coordinate of $\boldsymbol{p}_a^{\text{aux}}(t)$
$z^{\text{main}}(t)$	z coordinate of $\boldsymbol{p}^{\text{main}}(t)$
z^{obs}	z coordinate of $\boldsymbol{p}^{\text{obs}}$
ρ^{main}	Desired distance between each auxiliary UAV and the main UAV
ρ^{aux}	Desired distance between various auxiliary UAVs

7

Fuzzy Logic and Explainable AI for Trustworthy Human-Robot Collaboration during Firefighting

In firefighting scenarios, robots are used to substitute humans in dangerous tasks but are often tele-operated. Therefore, approaches to increasing robot autonomy while maintaining meaningful human control and situational awareness are urgently needed. In this chapter, we apply the socio-cognitive engineering methodology to propose such an approach that integrates fuzzy logic control for robot decision making on autonomous navigation with explainable artificial intelligence for robot communication. This approach ensures meaningful human control and situational awareness by keeping firefighters in the loop. We first elicited foundational design requirements and preferences during discussions with firefighters from the Rotterdam Fire Brigade (The Netherlands). This enhanced our approach by specifying a control system that receives inputs from both the human and the environment and determines the best possible target destinations for robot navigation. The human then selects the preferred target destination using a graphical user interface that visualizes crucial information about the possible destinations and the inputs used to determine them. This approach should be evaluated through 1) user studies assessing human situational awareness, workload, and trust and 2) comparisons with state-of-the-art robot navigation methods. Finally, our approach should be iteratively refined by repeating

This chapter outlines the core idea developed through a research collaboration with Ruben Verhagen, conducted within the research line of the AI*MAN lab, which is part of the TU Delft AI Labs programme.
M. Baglioni and R. Verhagen contributed to the conceptualization and the methodology, performed the experiments, assessed the results, and wrote the first draft of the chapter. A. Jamshidnejad, M. Tielman and M. Neerincx contributed to the methodology, assessed the results, edited the final draft, and performed supervision.

the foundation, specification, and evaluation phases of the socio-cognitive engineering methodology.

7.1. INTRODUCTION

HUMANS often need to collaborate in teams to execute complex tasks such as firefighting. However, human-only teamwork is sometimes limited or even impossible, for example, when it is too dangerous for firefighters to enter burning buildings. Explore and extinguish robots, deployed in human-robot teams, can substitute firefighters in life-threatening tasks [362]. These robots are currently often teleoperated by a firefighter from outside the building. This is challenging because the robot is beyond the perception field of the firefighter, mainly in poorly visible conditions. In addition, teleoperating the robot adds another burden to the already substantial workload of firefighters. Therefore, enhancing robot autonomy is highly desired to improve human-robot collaboration efficiency and reduce firefighter workload [362].

Such an increase in autonomy will raise important challenges for trustworthy human-robot collaboration during firefighting. For example, how to ensure meaningful human control over these increasingly autonomous robots [363], what to communicate with humans to maintain sufficient human situational awareness, and how to balance robot autonomy and human desires to build trust in these high-risk scenarios. This chapter presents our approach that addresses these challenges by combining fuzzy logic control (FLC) and explainable artificial intelligence (AI) for trustworthy human-robot collaboration during firefighting. We follow the socio-cognitive engineering methodology to first document firefighter requirements before working towards an initial prototype of our approach [364]. This approach incorporates FLC to determine temporary navigation destinations for the robot and explainable AI to support human-in-the-loop decision making. Overall, the main contributions of this chapter are the following:

1. Integrate FLC with explainable AI for human-robot collaboration.

2. Propose a human-in-the-loop approach that increases robot autonomy in firefighting scenarios.

3. Advance the development of trustworthy human-robot collaboration in high-risk environments.

4. Establish a foundation for future evaluations in real-world scenarios.

7.2. BACKGROUND

7.2.1. HUMAN-ROBOT TEAMWORK FOR FIREFIGHTING

Firefighting generally involves four possible deployment tactics on two axes: inside versus outside and offensive versus defensive. The defensive outside deployment aims to limit (environmental) damage by preventing the fire from spreading to adjacent areas and limiting the effects of smoke. The offensive outside deployment is used when it is unsafe for firefighters to enter a building. This tactic aims to improve the survival conditions of potential victims, enable a safe entry for firefighters, prevent fire spread, and extinguish the fire. The defensive inside deployment aims to provide an evacuation opportunity, prevent fire and smoke spread, and limit damage. Finally, the offensive inside deployment aims to fight the fire and rescue people. Introducing explore and ex-

tinguish robots into human firefighting teams allows for an offensive inside deployment even when conditions are unsafe for firefighters.

These explore and extinguish robots are usually provided with several sensors (e.g., temperature, LiDAR, explosion danger, RGB cameras, and thermal imaging) and other equipment (e.g., fire hose and water shield protection) to enable detection, localization, mapping, navigation, protection, and extinguishing [365]. These sensors provide crucial information that helps firefighters decide the deployment tactic or localize the fire source. The collaboration between firefighters and explore and extinguish robots demonstrates how human-robot teams can potentially combine the strengths of both parties to execute tasks that neither can execute alone [366].

On the one hand, (semi-)autonomous robot behavior is desirable to reduce the workload of firefighters by eliminating the need for continuous teleoperation [362]. On the other hand, it is important to actively keep firefighters involved in enhancing the robot's decisions during the mission [362]. While recent works have enhanced robot autonomy during urban search and rescue operations [367]–[369], results show that human operators used (semi-)autonomous robot behavior less than expected. In fact, they often took control over the robot due to a lack of transparency on its autonomous decision making, which is necessary for understanding robot behavior, situational awareness, and trust in the robot [367].

7.2.2. SITUATIONAL AWARENESS

Situational awareness (SA) is defined as the perception of elements in a situation, the comprehension of their meaning, and the projection of their status in the future [370]. The quality of this awareness in humans can be influenced by their training, capabilities, experience, and workload [371]. SA is a critical prerequisite for human decision making and action execution, making it essential for effective teamwork [371]. This importance even led to developing teamwork-centered SA concepts such as *team SA* and *shared SA* [372], [373].

Bi-directional communication is crucial in human-robot teams to develop and maintain SA [374]. This communication of relevant information can create a shared mental model between team members [375]. Graphical user interfaces can effectively facilitate such communication between humans and robots [367]. Human perception, comprehension, and projection can be achieved through transparent and explainable AI [374], [376]–[378]. However, this transparency and explainability should not reduce SA through information overload [371], [379], [380].

Human-robot communication can be facilitated using button-clicking behavior in graphical user interfaces to provide human input to robots [374]. However, it is difficult to achieve robot SA for all three levels (i.e., perception, comprehension, and projection) based on human communication only. Robots also need learning and reasoning capabilities to ensure comprehension and projection of situational elements. Finally, robots require proper sensors to perceive and map environments accurately.

Building and maintaining sufficient SA is not only essential for effective human-robot teamwork, but also for ensuring meaningful human control during the collaboration [365], [381]. Without proper SA, humans cannot effectively understand, predict, correct, or change the actions and behavior of robot teammates [365], [381]. However,

SA is insufficient for meaningful human control as it also requires human understanding of their moral responsibility for outcomes resulting from robot behavior and teamwork.

7.2.3. MEANINGFUL HUMAN CONTROL

Meaningful human control implies that humans should ultimately remain in control of, and thus morally responsible for, the behavior of (semi-)autonomous robots [363]. In addition to transparency and SA challenges, increasing robot autonomy during firefighting also raises challenges for meaningful human control during human-robot collaboration [363]. Since firefighting involves morally sensitive situations and decisions, meaningful human control is crucial to ensure humans can be held accountable for (semi-)autonomous robot behavior [382]. This, however, requires that humans are both aware and equipped to act upon their moral responsibility [381].

Recent works have proposed several actionable solutions for ensuring meaningful human control in human-robot teams. For example, value-sensitive design to respect human values and norms [381], [383], team design patterns to shape meaningful human control [381], [384], and having humans-in-the-loop to allow human control and responsibility [379], [385]. Value-sensitive design is a design approach for technological systems that accounts for human values by identifying and eliciting them from stakeholders such as firefighters [383]. Meaningful human control can be ensured if these technological systems respect those values. Team design patterns are reusable solutions to recurring problems during human-robot collaboration [384]. They are used to design how tasks are distributed among team members over time and in response to situational task requirements [386]. Team design patterns have also been developed for teams operating in morally sensitive situations to ensure meaningful human control, for example, by allocating all moral decisions to humans [365], [387]. Finally, having humans-in-the-loop requires informed human decision making about robot behavior, for example, human decisions on robot navigation during firefighting [365], [388]. This active human involvement allows explicit human control and responsibility.

7.2.4. DECISION MAKING WITH HUMAN-IN-THE-LOOP

Autonomous robots can decide how to navigate environments using systematic control techniques [389]. While these techniques mostly aim to remove human intervention, control approaches in which humans intervene in decision making also exist. These approaches can incorporate human feedback in the control architecture for robot navigation in several ways. For example, a unified control system called *shared autonomy* or *shared control* is required when robots and humans co-exist and co-work. Such a system can integrate the perceptions and actions of humans and robots within a framework that allows the transition from shared autonomy to complete robot autonomy [390]. Another way is by incorporating haptic feedback in the control architecture [391], such as wearable devices to slow down social robots navigating alongside humans [392]. Human feedback can also be used as a reward for deep reinforcement learning to teach social robots to navigate by determining their velocity [393]. Another example based on deep learning is combining human feedback with large language models to enhance robot navigation and interaction with the environment in uncertain situations [394]. Finally, human feedback can also be incorporated using brain-computer interfaces, for example,

sensory information about obstacles not detected by robots [395], [396].

An ongoing challenge regarding these approaches is enriching human-robot interaction and allowing generalization to different scenarios. We believe this can be achieved using more communication modalities to ensure richer interactions, such as visual information displayed on a graphical user interface. Another challenge of these approaches is that they are computationally expensive and not applicable in real-time/onboard, especially those based on optimization. This challenge could be addressed by introducing efficient AI-based control techniques, such as FLC.

7.2.5. FUZZY LOGIC CONTROL

Fuzzy logic control (FLC) is a heuristic control technique that allows the integration of human knowledge into a control system. This can be achieved by incorporating rules provided by human experts into the control system, especially when no data or explicit physical knowledge of the system is available or complicated to obtain. A major advantage of FLC is that it is an efficient real-time decision making approach with low computational costs [45]. Thanks to this, FLC has allowed robots to navigate autonomously in various environments, including avoiding obstacles [397]–[400]. These FLC-based systems can use different inputs, such as the distance between robot and targets, the difference between robot orientation and robot angle towards targets, or the distance from and angle between robot and obstacles. The output determined by these systems often concerns the velocity of the robot.

Some variants of traditional FLC exist, including approaches that use Z-numbers [401] or probability theory [402] to address uncertainties and unknown conditions in robot navigation. FLC for navigation has also been combined with other (AI-based) technologies, such as neural networks [403], [404] to exploit previous information and deep reinforcement learning [405] to deal with unknown environments. In addition, FLC has been combined with proportional-integral-derivative controllers to improve motion tracking performance [406], particle swarm optimization to create paths of optimal length [404], and genetic algorithms to determine an optimal sequence of target points [322]. A main advantage of FLC-based approaches is that they usually require less computational time than other control approaches. In contrast, a crucial challenge of most FLC-based approaches is the effective involvement and incorporation of human operator decisions during control processes.

7.3. PROPOSED APPROACH

7.3.1. FOUNDATION

We applied the socio-cognitive engineering methodology [364] to develop our (semi-) autonomous robot navigation and communication approach for firefighting in human-robot teams. We first elicited foundational knowledge regarding operational demands, technology, and human factors based on discussions with firefighters from the Rotterdam Fire Brigade as direct stakeholders. These discussions provided us with preferences and requirements regarding robots and human-robot collaboration that we integrated into our proposed approach. For example, the firefighters specified that enhanced autonomy in navigation is their most significant challenge and desire. They also told us

that other robot functionalities, such as computer vision to interpret the environment and detect objects, have lower priority and can be done by them.

Regarding the navigation destinations of the robot, the firefighters emphasized the importance of identifying and entering doors when exploring burning buildings to search and rescue victims. Therefore, our approach selects doors as important temporary destinations for the robot. Furthermore, the firefighters told us that the final destination of a firefighting mission is initially unknown since it depends on initially unavailable information, such as the location of victims and fire. Therefore, the robot should consider and select intermediate target points during the mission, which we call *temporary destinations*.

Concerning robot mapping of the environment, the firefighters explained that this should not be a separate initial phase but instead be performed in parallel with navigating and exploring the environment. This way, the firefighting mission is not delayed. In addition, buildings usually change when a disaster strikes due to destruction, debris, and fire. Therefore, the robot should update a map of the building (if available) using its sensors.

For human-robot collaboration, the firefighters mentioned their preference for staying actively in the control loop. Robots that autonomously navigate and do not allow human intervention can create issues when their autonomous decisions are not optimal or suited for the situation, for example, because of environmental or measurement uncertainties. Therefore, changing the robot's decisions, restarting the decision making process, or simply stopping the autonomous controller allows explicit human control.

Lastly, concerning decision making, the firefighters indicated a preference for using the robot's controller as a decision-support system. Multiple objectives can be pursued during a firefighting mission, such as exploring the unknown environment, looking for trapped victims, and moving to the fire source to extinguish it. Therefore, the firefighters would like the controller to propose multiple target points so they can select their preferred one according to their perception of current priorities in the mission. Finally, the firefighters mentioned a preference for visual robot communication of these target points rather than textual or verbal communication.

7.3.2. SPECIFICATION

Based on this foundational knowledge, we specified our (semi-)autonomous robot navigation and communication approach for firefighting. This approach mainly combines FLC with explainable AI (see Figure 7.1).

FEEDBACK-BASED FUZZY LOGIC CONTROLLER

The core of the robot control system is based on FLC. The controller receives two types of input: inputs from sensing the environment and inputs from communicating with the human. The inputs from sensing the environment include the time to reach a destination and the clutter on a path towards a destination. The robot control system determines these inputs without any human intervention. The inputs from human communication include the location of victims, doors, and obstacles. The human communicates these inputs to the robot control system using a graphical user interface, representing the human *feedback* to the controller.

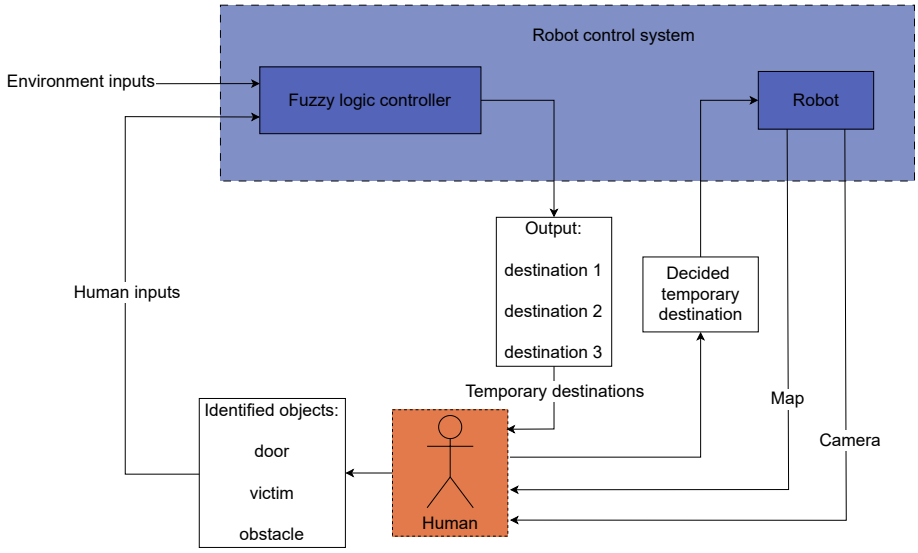


Figure 7.1: Architecture of the proposed control approach. The human is shown in the orange block, and the robot control system is shown in the blue block. The fuzzy logic controller receives inputs from the environment and from the human. The fuzzy logic controller then provides three temporary destinations for the robot, and the human selects one of these as the robot's real temporary destination. The robot communicates map and camera images of the environment to the human, which the human uses to identify the objects that the fuzzy logic controller uses as inputs.

Based on these inputs, the controller selects the best temporary destinations by determining the *attraction* value of each possible target point within the perception field of the robot. These attraction values are the output of the controller and reflect the relative importance of reaching that destination. They are computed using a set of fuzzy rules designed based on expert knowledge elicited during discussions with the firefighters. For example, target points close to victims and doors, far from obstacles, and quick and easy to reach have high attraction values.

In our approach, human feedback simplifies the complexity of the robotic system, the control system architecture, and the algorithms the robot uses to implement its tasks. A completely autonomous system may require additional robots and sensors, for example, tiny drones equipped with cameras to identify victims, doors, and obstacles by employing computer vision algorithms that perform object recognition [3]. However, there are challenges associated with using computer vision in firefighting scenarios. On the one hand, computer vision cannot recognize objects in color images due to smoke that may occlude the view. On the other hand, thermal images may be used to see shapes in smoke, but such images may not be reliable for computer vision algorithms. Instead, the graphical user interface in our approach displays images to the human-in-the-loop in real time, who then identifies crucial environmental elements that are communicated back to the system. This eliminates the need for more complex and expensive systems and algorithms. Moreover, the human's capability to identify objects based on informa-

tion from different sensor sources in the graphical user interface can address the challenges associated with employing computer vision and sensor fusion methods. This way, having the human-in-the-loop also facilitates meaningful human control during the mission as the human bears direct responsibility for object identification.

ROBOT TRANSPARENCY AND EXPLAINABILITY

The output of the controller consists of the three possible temporary destinations within the robot's perception field with the highest attraction values. The system will then communicate these possible target points to the human and ask them to select their preferred one. Given that the robot's perception field consists of numerous possible destination targets with attraction values, it is important to intuitively communicate the top three options without unnecessarily overloading humans. We believe a heatmap that overlays all attraction values on the 2D map of the environment will ensure SA and facilitate human decision making about the temporary destinations (see Figure 7.2). Moreover, such a plot excellently aligns with the preferences of firefighters for visual communication with robots over textual or verbal communication.

However, we hypothesize that merely communicating the top three temporary destinations to the human may not provide the sufficient behavioral transparency required by the human to understand the robot's decision making, achieve SA, and build trust. In addition, the system could explain why it selected these three temporary destinations based on the controller inputs. Contrasting these three against the target point with the lowest attraction value is hypothesized to further enhance the humans' predictability and understanding of the robot's reasoning process [407]. Given the firefighters' preference for visual communication with robots, these explanations should be communicated using data visualization techniques, such as a radar chart facilitating rapid comparison. As it is crucial to avoid human information overload, making these explanations more abstract over time or enabling them to appear on demand are important design considerations.

7.3.3. EVALUATION

The final phase of the socio-cognitive engineering methodology is evaluating the specified prototype. Our main contribution is this prototype approach, but future work should focus on evaluating it. Properly evaluating our approach should involve user studies that compare various levels of robot autonomy in our architecture. For example, our approach can be compared to one where the robot is fully teleoperated without FLC in its decision making and another where the robot operates autonomously based solely on the FLC system and with humans acting only as supervisors. Proper evaluation should also include studies that compare our approach against other state-of-the-art approaches for autonomous robot navigation on relevant metrics such as completion time, number of detected victims, and percentage of area covered.

User studies on the effectiveness of robot transparency and explainability are also required to evaluate and iteratively refine our approach. For example, whether or not a heatmap overlayed on the basic 2D map of the environment indeed improves human understanding, SA, and trust. These assumptions and hypotheses could first be evaluated using non-expert participants and then further validated by expert firefighters. The

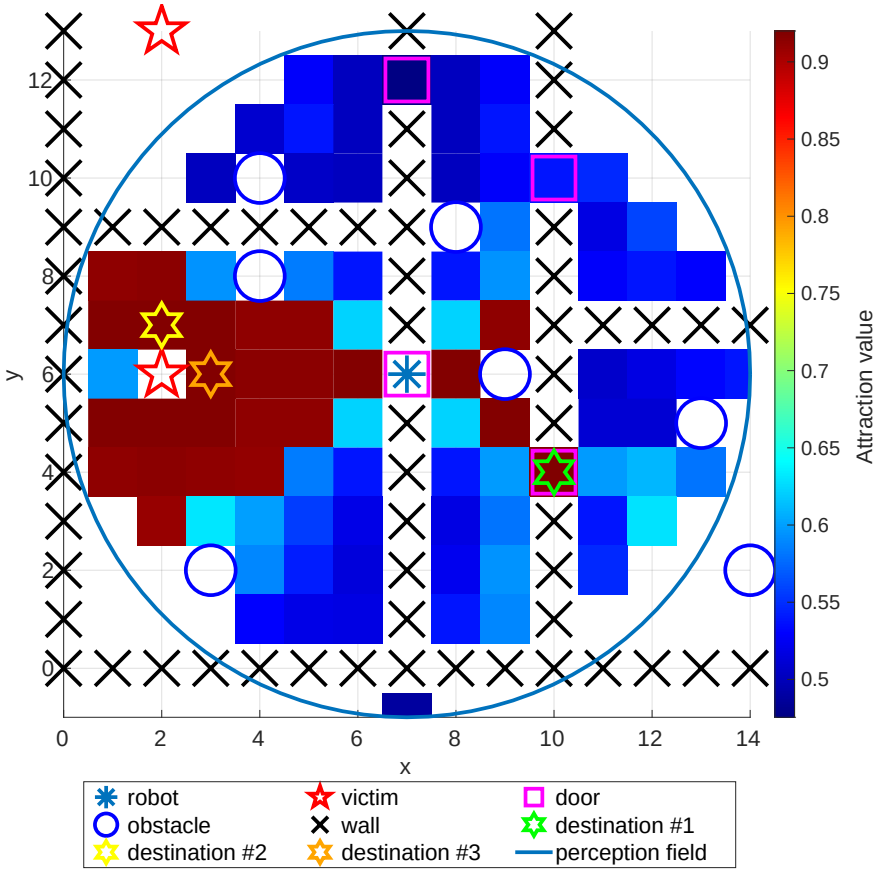


Figure 7.2: Basic example of a heatmap with attraction values overlayed on a recreated 2D environment map. The top three temporary navigation destinations are also added to the plot. Higher attraction values indicate a higher relative importance of reaching that destination.

resulting insights could then be used to refine our approach iteratively.

7.4. DISCUSSIONS

7.4.1. ADVANTAGES OF OUR APPROACH

Our proposed (semi-)autonomous robot navigation and communication approach has several advantages. First, the human-centered design, elicitation of expert knowledge using fuzzy logic, and integration of explainable AI can facilitate trustworthy human-robot collaboration. In addition, having a human-in-the-loop can ensure meaningful human control in these morally sensitive situations. Including the human in the control loop of our approach also offers the added benefit of reducing system complexity. This applies to both the number and complexity of robots required for scanning, mapping, and detecting key elements via sensors and the complexity of the algorithms required to

process sensed data.

The proposed FLC-based approach also offers advantages over traditional control approaches. In particular, the computational efficiency of FLC allows real-time deployment, while its inclusion of human expert knowledge enables it to mimic the human decision making process. At the same time, the modularity of our approach also allows substituting the fuzzy logic controller with alternative methods whenever the design requirements differ from ours.

Finally, our approach can also be generalized and translated to different contexts. For robots working under human supervision, we believe our approach can be applied to leakage detection in buildings inaccessible to humans, such as in a power plant or oil platform. Another example could be surveillance or patrolling robots that provide views of the surroundings of a site more efficiently than humans. For robots working alongside humans, our approach could be applied to (flying) robots that check if parts of old buildings or church artworks require maintenance. Another application could be space exploration, where rovers explore new lands to cover as much area as possible or to find and extract terrain samples while humans explore other directions. Moreover, our approach to configuring the inputs to the controller is modular for any possible new application. Specifically, it allows the design of different inputs derived from humans or the environment that fit the context of that particular application.

7.4.2. LIMITATIONS OF OUR APPROACH

We acknowledge a few limitations of our approach. First, the human-in-the-loop who decides the temporary navigation destinations while being supported by robot transparency and explainability might have a workload that is too high. For our approach to be effective, robot transparency and explainability should not cause information overload for the human. The human workload can be decreased using on-demand or adaptive explanations. However, the workload can also be decreased by increasing robot autonomy, for example, by implementing reliable computer vision to detect doors, victims, and obstacles in smoke-filled situations.

Another limitation of our approach concerns the lack of learning in our controller. In relation to SA, our approach can perceive the environment through the robot's sensors and human feedback. In contrast, comprehension of the situation is only partially achieved by the rule base of the controller to reason about the best temporary destinations given the environmental information. However, the controller does not learn any environmental features that can be used to give specific meaning to sensed data. This limitation can be overcome by modifying our controller by employing techniques that combine FLC with machine learning approaches such as neuro-fuzzy systems [408]. However, a drawback of such approaches is the lack of guarantees for stability and safety in critical scenarios.

In addition, our FLC-based approach is inherently reactive. It can be used to navigate in real-time by gathering new information while the mission is proceeding, but it does not have prediction capabilities to be included in decision making. Such capabilities are suitable when dealing with dynamic environments with moving obstacles or targets, for example, because of fire spread [150] or victim motion [212]. To handle these dynamic elements, the control system requires a model of their dynamics to predict their

behavior and to anticipate and counteract the motions that may create unsafety, which would be possible with approaches such as model predictive control [1], [20]. However, a drawback of employing such techniques is their higher computational complexity.

7.4.3. FUTURE WORK

Future work should involve evaluating our approach through studies like the ones described in Section 7.3.3. The three phases of the iterative socio-cognitive engineering methodology (foundation, specification, and evaluation) should then be repeated by including knowledge obtained after completing the evaluation phase. For example, an updated prototype of our approach could employ different control approaches, such as neuro-fuzzy control or model predictive control, if the evaluation phase reveals practical limitations of the current control approach.

Further work might also require tuning the control approach for scenarios in different contexts, such as humans and robots working alongside each other. For example, the control approach might require modifications when humans and robots influence each other's decisions by sharing the same space in the environment. In such cases, the controller could incorporate dynamic safety constraints or human intention recognition to adjust the robot's path and avoid interference.

7.4.4. CONCLUSIONS

In this chapter, we addressed the challenge of increasing autonomy in robot navigation while ensuring meaningful human control and situational awareness during human-robot collaboration for firefighting. We proposed a novel human-in-the-loop approach that integrates fuzzy logic control for robot decision making on autonomous navigation with explainable artificial intelligence for robot communication. This approach increases robot autonomy in firefighting scenarios, enabling firefighters to transition from teleoperating their robots to collaborating with them. Moreover, our approach leverages robot transparency and explainability to foster the human understanding and trust necessary for delegating control to the robot. Overall, this work contributes to the development of trustworthy human-robot collaboration in high-risk environments and lays the foundation for future evaluations of its effectiveness in real-world scenarios.

8

Conclusions and Future Research Directions

In this chapter, we present the concluding remarks of the PhD thesis. We first synthesize the main contributions and highlight the most significant results and findings derived from this research. We then revisit the research questions posed in Chapter 1 of the thesis and address them based on the outcomes of this PhD research. Afterwards, we discuss the broader implications of this research, both within the scientific community and in practical societal contexts. We then examine the key limitations of the work and their implications. Finally, we outline recommendations and promising directions for future research building upon the foundations that have been established in this PhD thesis.

REFLECTING ON CORE CONTRIBUTIONS

MOTIVATED by the need to improve the autonomy and efficiency of Search-and-Rescue (SaR) operations, this PhD thesis has advanced the state-of-the-art in robot mission planning, control, and perception through the integration of model-based, optimal, and human-inspired approaches. Specifically, we have:

- Advanced constrained optimal control for SaR mission planning by introducing Model Predictive Control (MPC)-based formulations that jointly address competing objectives of target tracking and area coverage under uncertainty — without providing reference trajectories — and have validated these formulations in computer-based simulations and real-world scenarios;
- Developed robust motion planning architectures that, by integrating heuristic planning and Robust Tube-based Model Predictive Control (robust TMPC), enable autonomous safe navigation in dynamic, cluttered environments;
- Enhanced perception and mapping capabilities of autonomous robots through vision-based and fusion techniques that leverage affordable sensors, including methods for victim localization, motion tracking, and terrain elevation estimation in unknown environments, and have validated them experimentally;
- Proposed scalable and responsive multi-robot hierarchical control frameworks that integrate predictive optimal control and human-inspired Fuzzy Logic Control (FLC) to achieve efficient and coordinated mapping in complex SaR settings, supported by extensive comparative simulations and systematic analyses of parameters sensitivity and design configurations;
- Enabled hierarchically structured and optimal control-theoretic coordination for Unmanned Aerial Vehicles (UAVs) to enable autonomous wildfire suppression, including novel Line-of-Sight (LoS) occlusion avoidance and best-viewpoint tracking;
- Proposed trustworthy human-in-the-loop control by integrating Explainable Artificial Intelligence (XAI) into FLC-based decision making architectures, fostering autonomy of robots while preserving meaningful human oversight.

KEY FINDINGS

This PhD thesis has advanced the state-of-the-art in autonomous perception and control of robots, as well as in human-robot collaboration, with a particular focus on SaR. Through an integrated sequence of methodological developments and experimental validations, it has addressed critical challenges faced by autonomous robotic systems operating in disaster scenarios. The core findings of the thesis collectively demonstrate how predictive optimal control, human-inspired decision making, and vision-based perception can be systematically integrated to enhance autonomy, robustness, and coordinated behavior in uncertain, dynamic disaster environments, thereby addressing key gaps in current SaR robotics research.

- In Chapter 2, we demonstrated (i) the ability of MPC — even when not supported by reference trajectory planners — to balance victim-focused tracking with exploratory area coverage, outperforming both purely target-oriented and coverage-oriented baselines in terms of victim detection, mission time, and coverage efficiency; and (ii) the potential of achieving systematic area exploration through predictive control, by adapting and embedding force-based pedestrian motion models — explicitly capturing motion uncertainties and their evolution — directly into the MPC formulation.

  These findings show how MPC offers a principled mechanism for integrating dynamic target tracking and area coverage in environments with incomplete or evolving information, achieving up to 45% higher victim detection and 29% increased area coverage compared to methods focused purely on one objective.

- In Chapter 3, we extended this line of work to enable autonomous navigation of robots in dynamic, obstacle-rich environments by embedding obstacle motion models with uncertainty handling into a robust TMPC formulation. Compared against heuristic (Artificial Potential Function (APF)) and sampling-based (Rapidly exploring Random Tree (RRT*)) planners, our approach achieved (i) safe, time-efficient, and collision-free navigation despite nonlinear obstacle motion; and (ii) on average, 11% faster mission handling and 8% shorter paths, demonstrating the superiority of predictive, optimization-based planning over heuristic or sampling-driven alternatives.

  Collectively, Chapters 2 and 3 firmly establish MPC and robust TMPC as foundational tools for rigorous, real-time decision making in highly uncertain and dynamic SaR settings, consistently outperforming traditional approaches in both safety and mission efficiency. Results published in these chapters highlight the unique capability of MPC to principally unify exploration, target pursuit, and safety into a single, constraint-aware decision making framework, moving beyond ad-hoc or sequential planning strategies.

- In Chapter 4, we demonstrated how affordable vision sensors, fused across ground and aerial robots, can support environment mapping, victim localization, and motion tracking in challenging, occluded settings.

  In particular, (i) extensions of pose estimation based on an existing object detection algorithm, YOLO (You Only Look Once), enabled robust inference of occluded victim positions with errors below 10%; and (ii) collaborative multi-robot fusion significantly improved tracking accuracy and terrain mapping, reducing trajectory deviation by 28% and terrain elevation error by over 22% compared to single-robot sensing.

  These findings illustrate how multi-modal perception and sensor fusion are capable of transforming limited onboard sensing into actionable, high-fidelity situational awareness for SaR tasks.

- In Chapter 5, we developed a hierarchical control architecture integrating local FLC with supervisory MPC for flying robots engaged in mapping uncertain SaR environments.

This framework delivered (i) the ability to adaptively tune local FLC controllers via supervisory MPC, preventing performance degradation, enhancing multi-robot coordination, and improving optimality (up to 17%) over time; and (ii) computational efficiency gains (up to 68%) over centralized MPC, achieved by delegating fast, local planning to lightweight decentralized FLC systems, while retaining near-optimal performance through supervisory MPC oversight.

- In Chapter 6, we addressed multi-UAV wildfire suppression, introducing an integrated control framework bridging Task Hierarchical Control (THC) for multi-task allocation, Path Integral Control (PIC) for trajectory optimality, and best viewpoint planning enhanced by developing a novel LoS-based occlusion avoidance mechanism.

  This architecture yielded (i) a 44% improvement in obstacle-free visual coverage through proactive occlusion avoidance; and (ii) a 39% reduction in path length for UAVs via PIC-driven trajectory optimization during coordinated suppression missions.

  Collectively, Chapters 5 and 6 establish hierarchical and multi-robot control frameworks that systematically integrate predictive optimal control, human-inspired local decision making, and task coordination strategies to achieve scalable, efficient autonomy in complex SaR and disaster response scenarios.

- In Chapter 7, we proposed a framework that integrates FLC-based robot autonomy with XAI interfaces for firefighting scenarios. This chapter serves as a forward-looking contribution, laying the groundwork for trustworthy human-robot collaboration and meaningful shared control in safety-critical missions.

  While not yet validated in real-world settings, it establishes a principled foundation and design paradigm for future empirical studies and deployment in operational disaster environments.

8

ANSWERS TO THE RESEARCH QUESTIONS

Building on the results from Chapters 2–7, the research questions posed in Chapter 1 are addressed as follows:

1. **How can optimal control approaches, particularly, MPC, be extended to enable mission planning strategies that balance the competing objectives of target tracking and area coverage in uncertain SaR environments?**

   Chapters 2 and 3 demonstrated that MPC can achieve this balance by embedding explicit models of victim motion, search area evolution, and uncertainty within a unified control framework.

   Chapter 2 introduced a multi-objective MPC formulation that systematically integrates target-tracking and coverage-expansion objectives, showing how control-theoretic optimization can formalize what are often treated as heuristic trade-offs in SaR planning.

Chapter 3 extended this approach by incorporating robustness to dynamic, nonlinear obstacles via robust TMPC, illustrating how predictive control can explicitly reason about uncertainty and environmental variability while preserving safety guarantees.

2. **How can optimal and human-inspired control methods be generalized to handle dynamic environments involving moving targets, uncertain obstacle behavior, and unmodeled disturbances, while providing guarantees of robustness and safety?**

   Chapters 3 and 5 addressed this question by developing complementary strategies that combine the strengths of predictive optimal control with the flexibility of human-inspired reasoning.

   Chapter 3, in particular, introduced a robust TMPC formulation that explicitly incorporates uncertain, nonlinear obstacle dynamics into the control problem, enabling safe navigation with formal guarantees while maintaining efficiency in complex, evolving environments.

   Building on this, Chapter 5 proposed a hierarchical architecture coupling supervisory MPC with local FLC systems. This design fused long-term optimality of MPC with rule-based adaptability and low computational cost of FLC , allowing rapid local responses to varying conditions, without sacrificing global mission objectives.

   Together, these chapters demonstrate how (robust) predictive control and human-inspired reasoning can be systematically combined to handle uncertainty, ensure safety, and maintain real-time responsiveness in dynamic SaR scenarios.

3. **How can vision-based perception be advanced and fused across heterogeneous ground and aerial robots to enable scalable, cooperative mapping and target tracking within SaR environments under limited sensing and communication constraints?**

   Chapters 4 and 6 tackled this research question by introducing advanced, multi-robot perception techniques that leverage machine vision, homography, geometrical relationships in 3D models of UAVs, and control-theoretic methods for enhanced robotic perception.

   On the one hand, Chapter 4 demonstrated how RGB camera-based perception can be leveraged and fused across heterogeneous flying and ground robots to achieve accurate victim localization, cooperative mapping, terrain elevation estimation, and robust target tracking under sensing limitations, extending YOLO-based detection to infer occluded body keypoints and relative distances, using tomography estimation, anthropometric proportions, and symmetry.

   On the other hand, by deploying occlusion avoidance based on LoS and incorporating it into a THC framework, Chapter 6 further enhanced Field of View (FoV) maintenance and enabled persistent situational awareness, even in cluttered or partially observable environments.

These contributions demonstrate how collaborative visual sensing can compensate for limited onboard capabilities and unreliable communication, offering a scalable perception framework for complex SaR missions.

4. **How can optimal and human-inspired control approaches be systematically integrated into hybrid architectures to exploit their complementary strengths, ensuring both real-time responsiveness and long-term optimality in SaR missions?**

   Chapter 5 addressed this question by proposing a hybrid, hierarchical control architecture that integrates the long-term planning and constraint-handling capabilities of supervisory MPC in the higher level, with the fast, heuristic responsiveness of FLC in the lower layer.

   This integration allows the system to adaptively tune control behavior in response to evolving mission demands and leverages the predictive power of MPC, while maintaining the computational efficiency and reactivity of FLC.

   Such a hierarchical design, where layers operate at different frequencies and employ distinct control strategies, allows the resulting layered framework to outperform both standalone MPC and purely heuristic approaches across multiple performance metrics, including optimality and computational efficiency.

5. **How can the proposed control and perception approaches be validated through rigorous simulation and real-world experiments, and how can their performance be benchmarked against state-of-the-art approaches under realistic SaR scenarios?**

   The thesis employed a comprehensive validation strategy, combining high-fidelity computer-based simulations with real-world experiments to benchmark the proposed frameworks against state-of-the-art methods.

   Simulated environments closely replicated important characteristics of disaster conditions, enabling controlled comparison with heuristic, sampling-based, and rule-based baselines.

   Chapters 2 and 4 further substantiated simulation findings through physical robot experiments in lab settings, demonstrating consistent improvements in victim detection, tracking accuracy, and mapping fidelity.

   This dual-layer validation confirms both the theoretical soundness and the practical feasibility of the proposed methods.

6. **How can explainable and trustworthy human-in-the-loop decision making be integrated with autonomous SaR robotics to balance human oversight and robot autonomy in time-critical missions?**

   Chapter 7 focused on this question by proposing a forward-looking framework that integrates FLC-based autonomy with XAI interfaces to support human oversight in firefighting scenarios.

By leveraging the transparent, linguistically interpretable structure of fuzzy rules, combined with heuristic decision making inspired by human reasoning, the framework enables robots to propose understandable navigation strategies that human operators can evaluate and modify through an intuitive graphical interface.

Although full real-world validation remains a future step, the framework lays critical groundwork for achieving transparent, trustworthy collaboration in safety-critical contexts to preserve human moral agency without undermining robotic autonomy, which is necessary for time-sensitive SaR operations.

Social and Scientific Impact

This section outlines the social and scientific impact of the research presented in this PhD thesis.

Social Impact

The social impact is related to the improvement of SaR operations, especially for low-income countries, and firefighting operations.

Enhancing SaR Operations The control frameworks developed in Chapters 2, 3, and 5 have the potential to significantly advance SaR operations, by enabling autonomous robot control for tasks that are impossible or hazardous for humans. This significantly improves safety and minimizes human exposure to danger.

For instance, the hierarchical control architecture proposed in Chapter 5 allows to reduce operational complexity and improve efficiency in joint human-robot missions.

While the initial deployment of robots may increase upfront costs, their expendability relative to human life yields a clear net social benefit. Additionally, the proposed approaches rely on widely available sensors and affordable hardware, such as the camera-based perception systems from Chapter 4, while Chapter 6 further reduces cost barriers by minimizing reliance on high-bandwidth communication or heavy on-board computation.

These contributions altogether make the frameworks of this thesis also accessible for low-income countries, which bear the brunt of natural disasters [157].

For scenarios with access to higher-end robotic platforms, the MPC-based planning frameworks, especially the one introduced in Chapter 2, offers superior victim tracking balanced with exploratory coverage. Meanwhile, the robust TMPC variant proposed in Chapter 3 ensures safe navigation around dynamic obstacles, enhancing the safety and reliability of more expensive robotic assets.

Improvement of Firefighting Operations Chapter 6 introduces a control architecture that enables a team of UAVs to autonomously track primary fire-suppressing UAVs. These UAVs, tasked with providing optimal viewpoints to human operators that steer the fire-suppressing UAVs, reduce cognitive workload and improve situational awareness. This ultimately enhances the efficiency and effectiveness of wildfire suppression operations.

Furthermore, Chapter 7 advances robot autonomy in firefighting missions by integrating explainable decision making mechanisms that make robotic behavior more

transparent and trustworthy for humans. This improves human-robot collaboration by allowing firefighters to better understand and oversee robotic actions, while reducing their operational workload and cognitive burden in high-stress scenarios.

SCIENTIFIC IMPACT

The scientific impact of this thesis lies primarily in advancing control approaches, especially MPC (including robust and hybrid frameworks), human-in-the-loop frameworks, and perception strategies for autonomous SaR robotics.

Impact on MPC Approaches The control approach proposed in Chapter 2 introduces a novel application of MPC by extending its traditional role in trajectory tracking to include area coverage objectives. This integration enables a principled trade-off between victim tracking and environment exploration through parameter tuning within the MPC formulation. Therefore, it broadens the scope of MPC in SaR contexts.

For robust MPC, Chapter 3 presents an innovative deployment of robust TMPC for path planning in dynamic, uncertain environments. Typically, nonlinear robust TMPC requires two controllers: a nominal controller (which excludes uncertainty) and an ancillary controller (which compensates for uncertainty). Here, the nominal controller is replaced with an MPC-based heuristic path planner that explicitly models moving obstacles, while the ancillary controller addresses uncertainty. This novel pairing demonstrates how robust TMPC can be adapted for safe, real-time planning in complex SaR scenarios.

Impact on Hierarchical Control with Combination of FLC and MPC Chapter 5 introduces a novel hierarchical control framework that integrates two distinct paradigms: FLC and MPC. This architecture exploits the complementary strengths of both approaches. The supervisory MPC layer enhances the heuristic decisions of FLC with predictive optimality and constraint handling, while the local FLC layer provides computationally efficient, fast responses for time-critical actions. By selectively triggering MPC only when necessary, the framework reduces computational demands without compromising control quality, establishing a new synergy between rule-based reasoning and model-based optimization.

This combination provides opportunities for improving both control approaches: On the one hand, through a supervisory MPC, the heuristic decision of an FLC controller can incorporate new levels of optimality; On the other hand, by allowing the regular decision to an efficient FLC, an MPC can become less computationally expensive, since it can be activated only when a certain event is triggered.

Impact on Human-in-the-loop Control The human-in-the-loop control framework presented in Chapter 7 introduces a dual integration of human decision making into the control loop. First, it allows humans to directly provide control inputs in parallel with environmental feedback. Second, it enables operators to select among multiple actions proposed by the controller that have been made interpretable for humans through fuzzy rules. This approach not only improves transparency and trust in autonomous systems,

but also empowers human oversight in safety-critical missions and ensures that robotic autonomy remains accountable to human judgment.

Impact on Perception Approaches The perception-related contributions of this thesis advance vision-based autonomy for resource-constrained SaR missions.

On the one hand, Chapter 4 extends YOLO to infer occluded keypoints from visible ones. This enables robust victim localization, even under visual clutter or partial occlusions. It further demonstrates novel applications of object detection in robotics, such as for victim tracking and cooperative mapping using only affordable RGB cameras. By incorporating homography-based image fusion and occlusion avoidance, it enables cooperative mapping and tracking in obstacle-rich environments with limited communication and sensing. This fusion of geometric vision techniques with multi-robot coordination establishes new methods for scalable, low-cost perception in disaster scenarios.

On the other hand, Chapter 6 advances multi-robot perception through LoS-aware viewpoint planning embedded within a THC framework to improve autonomous visual coverage for firefighting and SaR missions. The LoS-aware viewpoint coordination mechanism that dynamically assigns UAVs to observation positions maximizes visibility of target regions, while avoiding occlusions. This is achieved through geometric reasoning on LoS constraints and obstacle-aware positioning, and improves operator situational awareness and target monitoring even in cluttered, obstructed scenarios.

LIMITATIONS

This section outlines the key limitations of the research work presented in this PhD thesis and links these limitations to directions for future work aimed at addressing the remaining gaps.

First, the predictive models used in Chapter 2 for MPC-based victim tracking rely on approximated motion patterns of SaR victims. In practice, human behavior in emergencies is highly unpredictable and may deviate significantly from the modeled dynamics, leading to reduced tracking accuracy. Similarly, the robust TMPC framework in Chapter 3 assumes that obstacle motion can be approximated using predefined geometric models, including ellipsoidal trajectories. Such trajectories, however, may not capture arbitrary or erratic obstacle behavior that is often observed in real-world disaster scenarios.

Furthermore, both Chapters 2 and 3 handle uncertainty by modeling external disturbances as bounded disturbances. In reality, unbounded or highly variable uncertainties may violate these assumptions, and thus undermine safety guarantees and potentially compromise mission success in critical SaR deployments.

In Chapter 4, the proposed perception and mapping algorithms operate under the assumption that ground and aerial robots are either manually positioned or already in suitable locations. The chapter does not address autonomous coordination or motion planning for these heterogeneous robots, nor does it enforce inter-robot distance constraints. These, however, are vital for safe and efficient cooperative operation in cluttered disaster environments.

Chapter 5 introduces a hierarchical control approach, but omits local robot-to-robot communication mechanisms. This limitation restricts coordination during multi-robot

tasks (e.g., collaborative area coverage or formation control). Communication will also be instrumental in failure detection or task reallocation during mission execution, so it is a crucial aspect in multi-robot missions.

The perception-based collision avoidance strategy in Chapter 6, derived from LoS geometrical reasoning, is limited to static obstacles. This constraint restricts the applicability of the proposed approaches to ground-level hazards (e.g., trees or collapsed structures). However, multi-UAV operations in dynamic environments also require avoidance of aerial obstacles, including other UAVs, birds, or airborne debris. Similarly, obstacles that move on the ground (e.g., people, animals, vehicles) are common in SaR contexts, but remain unaccounted for in the framework of Chapter 6.

Finally, the human-in-the-loop framework of Chapter 7 inherently introduces delays in mission execution. While explainability and human oversight enhance safety and trustworthiness of robots, human decision making in various cases is slower than fully autonomous control algorithms, as it requires interpreting what is received through the interface and deliberating on system recommendations.

FUTURE WORK

In this section, we provide recommendations for future work, related to both theory and application.

THEORETICAL DIRECTIONS FOR FUTURE RESEARCH

This section outlines theoretical research avenues aimed at addressing the limitations identified in this thesis.

To address the challenge of modeling victim motion, future work should focus on data-driven motion prediction combined with frameworks that model social interactions, e.g., those based on game theory. Rather than relying on predefined parametric equations, which may fail to capture realistic pedestrian behaviors, motion models should be derived from, or enhanced by, empirical data. For victims, datasets can be collected from real-world scenarios, including building evacuations, pedestrian movement in urban areas, or post-disaster contexts. This requires large-scale, carefully designed experiments that employ diverse sensors for capturing multiple behavioral indicators, as well as interfaces (e.g., mobile apps) to regularly collect cognitive inputs and interaction tendencies of participants. Otherwise, such interfaces should be deployed in advance especially in areas with more risks of disasters, to collect data from real catastrophic events or pedestrians in urban contexts. Similarly, to enhance modeling obstacle motion, domain knowledge may be integrated with machine learning methods to precisely extract motion patterns in real time.

The limitations related to precise modeling of disturbances and terrain uncertainties mainly affect ground robots. While integrating domain knowledge and machine learning, as suggested earlier, is a promising avenue for enhancing such models, a practical solution is to avoid or reduce ground traversal by deploying flying robots. Although this extension eliminates challenges related to terrain-induced disturbances and helps to accelerate the mission execution, it introduces new challenges. A main one includes the need to remodel victim-safe zones from 2D into evolving 3D volumetric envelopes to prevent collisions with UAVs. For example, obstacle avoidance constraints should en-

sure that UAVs avoid unwanted contact with the head or raised arms of victims. Integrating safe set formulations with the LoS-based collision avoidance introduced in this thesis will allow to define repulsive constraints, while maintaining attractive forces toward search areas.

Finally, MPC-based trajectory generation may be integrated within the THC framework proposed in Chapter 6 to replace or complement the existing PIC-based planner. Additionally, incorporating the robust robust TMPC-based tracking approach from Chapter 3 enables safe navigation under bounded uncertainties. When further enhanced by learning-based motion models, MPC-driven planning, and LoS-based collision avoidance, the resulting THC framework yields robustness and generality, allowing SaR robots to safely operate in highly dynamic, cluttered, and uncertain environments.

Applied Directions for Future Research

This section discusses future work focused on practical applications.

While the control and perception methods developed in this thesis are tailored for SaR missions, their underlying principles and control frameworks have broad applicability across multiple other domains, including:

- *Environmental monitoring and conservation:* The multi-robot coordination, robust path planning, and active perception methodologies introduced in this thesis can be employed for tasks such as wildlife monitoring, habitat mapping, or pollution detection. In such applications, autonomous robots should navigate complex terrains and dynamically changing environments, while maintaining sensor visibility and avoiding obstacles.

- *Agricultural automation:* The hierarchical control framework that synergizes MPC and FLC can enhance precision farming operations, including crop monitoring, pest detection, and targeted spraying. This can be achieved by enabling UAVs and ground vehicles to cooperatively cover large fields in efficient and safe ways, while adapting to dynamic obstacles, e.g., farm workers and animals.

- *Industrial inspection and maintenance:* Autonomous aerial and ground robots equipped with the proposed perception and control strategies are suited for inspecting critical infrastructure (e.g., pipelines, power lines, large ships, industrial plants). Such robots should provide coordinated viewpoints from detection scenes and simultaneously adapt their trajectories to avoid obstacles or hazards.

- *Security and surveillance:* Coordinated teams of aerial and ground robots may deploy the trajectory planning and collision avoidance methods introduced in this thesis to perform perimeter patrols, crowd monitoring, or search operations in both urban and industrial settings. Such robots will further benefit from active perception and robust navigation methods developed in this thesis, especially when they operate in cluttered and dynamic environments.

- *Disaster response beyond SaR:* The frameworks may also support tasks, e.g., post-disaster infrastructure assessment, hazardous material monitoring, and environmental sampling, where autonomous coordination and adaptive responsiveness under uncertain conditions are crucial.

- *Smart cities and urban management:* Coordinated teams of robots can be deployed for traffic monitoring, infrastructure maintenance (e.g., inspecting bridges or road conditions), waste management, or emergency evacuation guidance, using the proposed control and perception frameworks to navigate complex urban landscapes safely and efficiently.

In all these applications, the integration of human-in-the-loop decision making, transparency through explainable autonomy, and scalable multi-robot coordination provides significant advantages and enables efficient and trustworthy deployment of robotic systems in diverse, real-world scenarios.

BIBLIOGRAPHY

[1] M. Baglioni and A. Jamshidnejad, "A novel MPC formulation for dynamic target tracking with increased area coverage for search-and-rescue robots", *Journal of Intelligent & Robotic Systems*, vol. 110, no. 4, p. 140, 2024.

[2] K. Rado, M. Baglioni, and A. Jamshidnejad, "Enabling robots to autonomously search dynamic cluttered post-disaster environments", *Scientific Reports*, vol. 15, no. 1, p. 34 778, 2025.

[3] B. Esteves Henriques, M. Baglioni, and A. Jamshidnejad, "Camera-based mapping in search-and-rescue via flying and ground robot teams", *Machine Vision and Applications*, vol. 35, no. 5, p. 117, 2024.

[4] C. Maxwell, M. Baglioni, and A. Jamshidnejad, "Model predictive fuzzy control: A hierarchical multi-agent control architecture for outdoor search-and-rescue robots", *arXiv preprint arXiv:2505.03257, submitted to Engineering Applications of Artificial Intelligence*, 2025.

[5] M. Baglioni, A. Patil, L. Sentis, and A. Jamshidnejad, "Achieving multi-UAV best viewpoint coordination in obstructed environments", *arXiv preprint arXiv:2410.08602, submitted to Robotics and Autonomous Systems*, 2024.

[6] K. Hewitt, "Environmental disasters in social context: Toward a preventive and precautionary approach", *Natural Hazards*, vol. 66, pp. 3–14, 2013.

[7] M. T. Chaudhary and A. Piracha, "Natural disasters—origins, impacts, management", *Encyclopedia*, vol. 1, no. 4, pp. 1101–1131, 2021.

[8] D. C. Cooper, *Fundamentals of search and rescue*. Jones & Bartlett Learning, 2005.

[9] L. K. Comfort, "International disaster assistance in the Mexico City earthquake", *FMHI publications*, 1986.

[10] A. Khorram-Manesh, K. Goniewicz, A. Hertelendy, and M. Dulebenets, *Handbook of disaster and emergency management*. Kompendiet, 2021.

[11] A. Aminzadeh and A. M. Khoshnood, "Multi-UAV cooperative search and coverage control in post-disaster assessment: Experimental implementation", *Intelligent Service Robotics*, vol. 16, no. 4, pp. 415–430, 2023.

[12] J. Liu, Y. Wang, B. Li, and S. Ma, "Current research, key performances and future development of search and rescue robots", *Frontiers of Mechanical Engineering in China*, vol. 2, pp. 404–416, 2007.

[13] A. Davids, "Urban search and rescue robots: From tragedy to technology", *IEEE Intelligent systems*, vol. 17, no. 2, pp. 81–83, 2002.

[14] S. Grogan, R. Pellerin, and M. Gamache, "The use of unmanned aerial vehicles and drones in search and rescue operations–A survey", *Proceedings of the PROLOG*, pp. 1–13, 2018.

[15] T. J. Tanzi, M. Chandra, J. Isnard, D. Camara, O. Sebastien, and F. Harivelo, "Towards "drone-borne" disaster management: Future application scenarios", in *XXIII ISPRS Congress, Commission VIII (Volume III-8)*, Copernicus GmbH, vol. 3, 2016, pp. 181–189.

[16] R. R. Murphy, S. Tadokoro, D. Nardi, *et al.*, "Search and rescue robotics", in *Springer handbook of robotics*, Springer, 2008, pp. 1151–1173.

[17] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 3, pp. 367–385, 2003.

[18] B. Shah and H. Choset, "Survey on urban search and rescue robots", *Journal of the Robotics Society of Japan*, vol. 22, no. 5, pp. 582–586, 2004.

[19] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective", *Journal of Intelligent & Robotic Systems*, vol. 72, pp. 147–165, 2013.

[20] J. B. Rawlings, D. Q. Mayne, M. Diehl, *et al.*, *Model predictive control: Theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[21] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory", *Journal of statistical mechanics: theory and experiment*, vol. 2005, no. 11, P11011, 2005.

[22] K. Michels, F. Klawonn, R. Kruse, and A. Nürnberger, *Fuzzy control: fundamentals, stability and design of fuzzy controllers*. Springer, 2007, vol. 200.

[23] R. R. Murphy, S. Tadokoro, and A. Kleiner, "Disaster robotics", in *Springer handbook of robotics*, Springer, 2016, pp. 1577–1604.

[24] J. Rajan, S. Shriwastav, A. Kashyap, A. Ratnoo, and D. Ghose, "Disaster management using unmanned aerial vehicles", in *Unmanned Aerial Systems*, Elsevier, 2021, pp. 129–155.

[25] 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", *Journal of Intelligent & Robotic Systems*, vol. 107, no. 3, p. 40, 2023.

[26] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances", *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[27] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications", *multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[28] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective", *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.

[29] Q. Gu, G. Bai, Y. Meng, G. Wang, J. Zhang, and L. Zhou, "Efficient path tracking control for autonomous driving of tracked emergency rescue robot under 6G network", *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, p. 5 593 033, 2021.

[30] F. A. de Alcantara Andrade, A. Reinier Hovenburg, L. Netto de Lima, *et al.*, "Autonomous unmanned aerial vehicles in search and rescue missions using real-time cooperative model predictive control", *Sensors*, vol. 19, no. 19, p. 4067, 2019.

[31] A. Bemporad and M. Morari, "Robust model predictive control: A survey", in *Robustness in identification and control*, Springer, 2007, pp. 207–226.

[32] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research", *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[33] E. Kang, Y. Liu, and H. Qiao, "Sliding mode-based adaptive tube model predictive control for robotic manipulators with model uncertainty and state constraints", *Control Theory and Technology*, vol. 21, no. 3, pp. 334–351, 2023.

[34] P. Bumroongsri, "Tube-based robust MPC for linear time-varying systems with bounded disturbances", *International Journal of Control, Automation and Systems*, vol. 13, no. 3, pp. 620–625, 2015.

[35] T. Peschke and D. Görges, "Robust adaptive tube tracking model predictive control for piece-wise constant reference signals", *International Journal of Robust and Nonlinear Control*, vol. 33, no. 14, pp. 8158–8182, 2023.

[36] A. Ryan and J. K. Hedrick, "A mode-switching path planner for UAV-assisted search and rescue", in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 1471–1476.

[37] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control", *IEEE transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[38] P. O. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems", *IEEE Transactions on Automatic control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[39] B. Kouvaritakis and M. Cannon, "Model predictive control", *Switzerland: Springer International Publishing*, vol. 38, no. 13-56, p. 7, 2016.

[40] T. Sun, Y. Pan, J. Zhang, and H. Yu, "Robust model predictive control for constrained continuous-time nonlinear systems", *International Journal of Control*, vol. 91, no. 2, pp. 359–368, 2018.

[41] F. Surma and A. Jamshidnejad, "State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances", *International Journal of Robust and Nonlinear Control*, vol. 35, no. 4, pp. 1319–1354, 2025.

[42] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey", *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[43] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", *Information fusion*, vol. 58, pp. 82–115, 2020.

[44] Y. Bai and D. Wang, "Fundamentals of fuzzy logic control—fuzzy sets, fuzzy rules and defuzzifications", in *Advanced fuzzy logic technologies in industrial applications*, Springer, 2006, pp. 17–36.

[45] D. J. Dubois, H. Prade, and R. R. Yager, *Readings in fuzzy sets for intelligent systems*. Morgan Kaufmann, 2014.

[46] A. Aboshosha and A. Zell, "Adaptation of rescue robot behaviour in unknown terrains based on stochastic and fuzzy logic approaches", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 3, 2003, pp. 2859–2864.

[47] M. Faisal, R. Hedjar, M. Al Sulaiman, and K. Al-Mutib, "Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment", *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 37, 2013.

[48] A. Din, M. Jabeen, K. Zia, A. Khalid, and D. K. Saini, "Behavior-based swarm robotic search and rescue using fuzzy controller", *Computers & Electrical Engineering*, vol. 70, pp. 53–65, 2018.

[49] J. A. Sánchez-Rojas, J. A. Arias-Aguilar, H. Takemura, and A. E. Petrilli-Barceló, "Staircase detection, characterization and approach pipeline for search and rescue robots", *Applied Sciences*, vol. 11, no. 22, p. 10 736, 2021.

[50] D.-A. Pham and S.-H. Han, "Design of combined neural network and fuzzy logic controller for marine rescue drone trajectory-tracking", *Journal of Marine Science and Engineering*, vol. 10, no. 11, p. 1716, 2022.

[51] J. Ni and S. X. Yang, "A fuzzy-logic based chaos GA for cooperative foraging of multi-robots in unknown environments", *International Journal of Robotics and Automation*, vol. 27, no. 1, p. 15, 2012.

[52] D. Dell'Anna and A. Jamshidnejad, "Evolving fuzzy logic systems for creative personalized socially assistive robots", *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105 064, 2022.

[53] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence", *Journal of International Humanitarian Action*, vol. 3, no. 1, pp. 1–18, 2018.

[54] Y. Wang, W. Liu, J. Liu, and C. Sun, "Cooperative USV–UAV marine search and rescue with visual navigation and reinforcement learning-based control", *ISA transactions*, vol. 137, pp. 222–235, 2023.

[55] H. Pan, X. Chen, J. Ren, *et al.*, "Deep reinforcement learning for flipper control of tracked robots in urban rescuing environments", *Remote Sensing*, vol. 15, no. 18, p. 4616, 2023.

[56] T. Yang, Z. Jiang, J. Dong, H. Feng, and C. Yang, "Multi agents to search and rescue based on group intelligent algorithm and edge computing", in *2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, IEEE, 2018, pp. 389–394.

[57] G. A. Cardona and J. M. Calderon, "Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations", *Applied Sciences*, vol. 9, no. 8, p. 1702, 2019.

[58] R. Krzysiak and S. Butail, "Information-based control of robots in search-and-rescue missions with human prior knowledge", *IEEE Transactions on Human-Machine Systems*, vol. 52, no. 1, pp. 52–63, 2021.

[59] L. E. Parker, D. Rus, and G. S. Sukhatme, "Multiple mobile robot systems", *Springer handbook of robotics*, pp. 1335–1384, 2016.

[60] F. Matoui, B. Boussaid, B. Metoui, and M. N. Abdelkrim, "Contribution to the path planning of a multi-robot system: Centralized architecture", *Intelligent Service Robotics*, vol. 13, no. 1, pp. 147–158, 2020.

[61] M. Raibail, A. H. A. Rahman, G. J. Al-Anizy, *et al.*, "Decentralized multi-robot collision avoidance: A systematic review from 2015 to 2021", *Symmetry*, vol. 14, no. 3, p. 610, 2022.

[62] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning", *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 809–824, 2022.

[63] C. Ocampo-Martínez, V. Puig, J. Grosso, and S. Montes-de-Oca, "Multi-layer decentralized MPC of large-scale networked systems", in *Distributed model predictive control made easy*, Springer, 2013, pp. 495–515.

[64] J. Kim, R. T. Fawcett, V. R. Kamidi, A. D. Ames, and K. A. Hamed, "Layered control for cooperative locomotion of two quadrupedal robots: Centralized and distributed approaches", *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4728–4748, 2023.

[65] Z. Liu, S. Zhou, H. Wang, Y. Shen, H. Li, and Y.-H. Liu, "A hierarchical framework for coordinating large-scale robot networks", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6672–6677.

[66] S. B. Slotine and B. Siciliano, "A general framework for managing multiple tasks in highly redundant robotic systems", in *Proceeding of 5th International Conference on Advanced Robotics*, vol. 2, Pisa, Italy, 1991, pp. 1211–1216.

[67] F. H. Panahi, F. H. Panahi, and T. Ohtsuki, "A reinforcement learning-based fire warning and suppression system using unmanned aerial vehicles", *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–16, 2022.

[68] E. Ausonio, P. Bagnerini, and M. Ghio, "Drone swarms in fire suppression activities: A conceptual framework", *Drones*, vol. 5, no. 1, p. 17, 2021.

[69] S. S. Moumgiakmas, G. G. Samatas, and G. A. Papakostas, "Computer vision for fire detection on UAVs—From software to hardware", *Future Internet*, vol. 13, no. 8, p. 200, 2021.

[70] A. Moffatt, N. Turcios, C. Edwards, *et al.*, "Collaboration between multiple UAVs for fire detection and suppression", in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2021, pp. 730–737.

[71] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks", in *2017 international conference on communication and signal processing (ICCSP)*, IEEE, 2017, pp. 0588–0592.

[72] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, *et al.*, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision", *Ieee Access*, vol. 8, pp. 191 617–191 643, 2020.

[73] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification", in *2017 IEEE 2nd International conference on big data analysis (ICBDA)*, IEEE, 2017, pp. 721–724.

[74] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects", *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[75] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS", *Machine learning and knowledge extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.

[76] L. Tian, N. M. Thalmann, D. Thalmann, Z. Fang, and J. Zheng, "Object grasping of humanoid robot based on YOLO", in *Advances in Computer Graphics: 36th Computer Graphics International Conference, CGI 2019, Calgary, AB, Canada, June 17–20, 2019, Proceedings 36*, Springer, 2019, pp. 476–482.

[77] D. H. Dos Reis, D. Welfer, M. A. De Souza Leite Cuadros, and D. F. T. Gamarra, "Mobile robot navigation using an object recognition software with RGBD images and the YOLO algorithm", *Applied Artificial Intelligence*, vol. 33, no. 14, pp. 1290–1305, 2019.

[78] S. Caputo, G. Castellano, F. Greco, C. Mencar, N. Petti, and G. Vessio, "Human detection in drone images using YOLO for search-and-rescue operations", in *International conference of the Italian association for artificial intelligence*, Springer, 2021, pp. 326–337.

[79] B. Valarmathi, J. Kshitij, R. Dimple, *et al.*, "Human detection and action recognition for search and rescue in disasters using YOLOv3 algorithm", *Journal of Electrical and Computer Engineering*, vol. 2023, no. 1, p. 5 419 384, 2023.

[80] A. Assa and F. Janabi-Sharifi, "A Kalman filter-based framework for enhanced sensor fusion", *IEEE Sensors Journal*, vol. 15, no. 6, pp. 3281–3292, 2015.

[81] J. Y. Loo, C. P. Tan, and S. G. Nurzaman, "H-infinity based extended Kalman filter for state estimation in highly non-linear soft robotic system", in *2019 American control conference (ACC)*, IEEE, 2019, pp. 5154–5160.

[82] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, "Contact-aided invariant extended Kalman filtering for legged robot state estimation", *arXiv preprint arXiv:1805.10410*, 2018.

[83] J. Dufek, X. Xiao, and R. R. Murphy, "Best viewpoints for external robots or sensors assisting other robots", *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4, pp. 324–334, 2021.

[84] G.-J. M. Kruijff, F. Colas, T. Svoboda, *et al.*, "Designing intelligent robots for human-robot teaming in urban search and rescue.", in *AAAI Spring Symposium: Designing Intelligent Robots*, vol. 16, 2012.

[85] M. Rodríguez, A. Al-Kaff, Á. Madridano, D. Martín, and A. de la Escalera, "Wilderness search and rescue with heterogeneous multi-robot systems", in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2020, pp. 110–116.

[86] A. Hashimoto, L. Heintzman, R. Koester, and N. Abaid, "An agent-based model reveals lost person behavior based on data from wilderness search and rescue", *Scientific reports*, vol. 12, no. 1, p. 5873, 2022.

[87] A. Murphy, M. Landamore, and R. Birmingham, "The role of autonomous underwater vehicles for marine search and rescue operations", *Underwater Technology*, vol. 27, no. 4, pp. 195–205, 2008.

[88] K. J. Rafferty and E. W. McGookin, "An autonomous air-sea rescue system using particle swarm optimization", in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, IEEE, 2013, pp. 459–464.

[89] D. Yanguas-Rojas, G. A. Cardona, J. Ramirez-Rugeles, and E. Mojica-Nava, "Victims search, identification, and evacuation with heterogeneous robot networks for search and rescue", in *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)*, IEEE, 2017, pp. 1–6.

[90] R. C. Richardson, A. Nagendran, and R. G. Scott, "Experimental tests of 'Bidi-bot': A mechanism designed for clearing loose debris from the path of mobile search and rescue robots", *Advanced Robotics*, vol. 26, no. 15, pp. 1799–1823, 2012.

[91] P. Xia, F. Xu, T. Zhou, and J. Du, "Benchmarking human versus robot performance in emergency structural inspection", *Journal of Construction Engineering and Management*, vol. 148, no. 8, p. 04 022 070, 2022.

[92] R. R. Murphy, D. Riddle, and E. Rasmussen, "Robot-assisted medical reachback: A survey of how medical personnel expect to interact with rescue robots", in *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No. 04TH8759)*, IEEE, 2004, pp. 301–306.

[93] P. T. Thavasi and C. Suriyakala, "Sensors and tracking methods used in wireless sensor network based unmanned search and rescue system-a review", *Procedia engineering*, vol. 38, pp. 1935–1945, 2012.

[94] S. Allali and M. Benchaïba, "Overview of search and rescue from robotics to wireless sensors and robots networks", in *Emergency and Disaster Management: Concepts, Methodologies, Tools, and Applications*, IGI Global, 2019, pp. 1212–1231.

[95] M. Bernard, K. Kondak, I. Maza, and A. Ollero, "Autonomous transportation and deployment with aerial robots for search and rescue missions", *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.

[96] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3D path planning and execution for search and rescue ground robots", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 722–727.

[97] G.-J. M. Kruijff, F. Pirri, M. Gianni, *et al.*, "Rescue robots at earthquake-hit Mirandola, Italy: A field report", in *2012 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, IEEE, 2012, pp. 1–8.

[98] H. Enshasy, M. Al-saleh, and A. Bu-shalf, "A comprehensive design of unmanned ground search and rescue robot", *J. Inf. Comput. Sci*, vol. 14, no. 1, pp. 52–80, 2019.

[99] L. Bruzzone and G. Quaglia, *Review article: Locomotion systems for ground mobile robots in unstructured environments*, 2012.

[100] D. Khaled, H. Aly, M. Khaled, N. Mahmoud, S. Shabaan, and A. Abdellatif, "Development of a sustainable unmanned surface vehicle (USV) for search and rescue operations", in *The international undergraduate research conference*, The Military Technical College, vol. 5, 2021, pp. 462–468.

[101] H. Mansor, M. H. Norhisam, Z. Z. Abidin, and T. S. Gunawan, "Autonomous surface vessel for search and rescue operation", *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1701–1708, 2021.

[102] S. Fattah, F. Abedin, M. Ansary, M. Rokib, N. Saha, and C. Shahnaz, "R3Diver: Remote robotic rescue diver for rapid underwater search and rescue operation", in *2016 IEEE Region 10 Conference (TENCON)*, IEEE, 2016, pp. 3280–3283.

[103] A. Matos, A. Martins, A. Dias, *et al.*, "Multiple robot operations for maritime search and rescue in euRathlon 2015 competition", in *OCEANS 2016-Shanghai*, IEEE, 2016, pp. 1–7.

[104] R. Bogue, "Disaster relief, and search and rescue robots: The way forward", *Industrial Robot: the international journal of robotics research and application*, vol. 46, no. 2, pp. 181–187, 2019.

[105] B. Choi, G. Park, and Y. Lee, "Practical control of a rescue robot while maneuvering on uneven terrain", *Journal of Mechanical Science and Technology*, vol. 32, pp. 2021–2028, 2018.

[106] D. Chatziparaschis, M. G. Lagoudakis, and P. Partsinevelos, "Aerial and ground robot collaboration for autonomous mapping in search and rescue missions", *Drones*, vol. 4, no. 4, p. 79, 2020.

[107] Z. Wang and H. Gu, "A review of locomotion mechanisms of urban search and rescue robot", *Industrial Robot: An International Journal*, vol. 34, no. 5, pp. 400–411, 2007.

[108] D. S. Drew, "Multi-agent systems for search and rescue applications", *Current Robotics Reports*, vol. 2, pp. 189–200, 2021.

[109] J. Sun, B. Li, Y. Jiang, and C.-y. Wen, "A camera-based target detection and positioning UAV system for search and rescue (SAR) purposes", *Sensors*, vol. 16, no. 11, p. 1778, 2016.

[110] H. Wang, C. Zhang, Y. Song, B. Pang, and G. Zhang, "Three-dimensional reconstruction based on visual SLAM of mobile robot in search and rescue disaster scenarios", *Robotica*, vol. 38, no. 2, pp. 350–373, 2020.

[111] K. Berns, A. Nezhadfard, M. Tosa, H. Balta, and G. D. Cubber, "Unmanned ground robots for rescue tasks", in IntechOpen, 2017, ch. 4, pp. 53–76.

[112] F. Lateef and Y. Ruichek, "Survey on semantic segmentation using deep learning techniques", *Neurocomputing*, vol. 338, pp. 321–348, 2019.

[113] T. Petříček, V. Šalanskỳ, K. Zimmermann, and T. Svoboda, "Simultaneous exploration and segmentation for search and rescue", *Journal of Field Robotics*, vol. 36, no. 4, pp. 696–709, 2019.

[114] D. Zhao, W. Zhang, and Y. Wang, "Research on personnel image segmentation based on MobileNetV2 H-Swish CBAM PSPNet in search and rescue scenarios", *Applied Sciences*, vol. 14, no. 22, p. 10 675, 2024.

[115] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey", *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[116] Z. Domozi, D. Stojcsics, A. Benhamida, M. Kozlovszky, and A. Molnar, "Real time object detection for aerial search and rescue missions for missing persons", in *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, IEEE, 2020, pp. 000 519–000 524.

[117] A. Banuls, A. Mandow, R. Vázquez-Martín, J. Morales, and A. García-Cerezo, "Object detection from thermal infrared and visible light cameras in search and rescue scenes", in *2020 IEEE international symposium on safety, security, and rescue robotics (ssrr)*, IEEE, 2020, pp. 380–386.

[118] M. Rahnemoonfar, T. Chowdhury, and R. Murphy, "RescueNet: A high resolution UAV semantic segmentation dataset for natural disaster damage assessment", *Scientific data*, vol. 10, no. 1, p. 913, 2023.

[119] M. Thoreau and F. Wilson, "SaRNet: A dataset for deep learning assisted search and rescue with satellite imagery", in *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA)*, IEEE, 2021, pp. 204–208.

[120] J. Gao, P. Li, Z. Chen, and J. Zhang, "A survey on deep learning for multimodal data fusion", *Neural computation*, vol. 32, no. 5, pp. 829–864, 2020.

[121] N. L. Ferreira, M. S. Couceiro, A. Araújo, and R. P. Rocha, "Multi-sensor fusion and classification with mobile robots for situation awareness in urban search and rescue using ROS", in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, 2013, pp. 1–6.

[122] H. Wang, C. Zhang, Y. Song, and B. Pang, "Information-fusion methods based simultaneous localization and mapping for robot adapting to search and rescue postdisaster environments", *Journal of Robotics*, vol. 2018, no. 1, p. 4 218 324, 2018.

[123] V. Zadorozhny and M. Lewis, "Information fusion based on collective intelligence for multi-robot search and rescue missions", in *2013 IEEE 14th International Conference on Mobile Data Management*, IEEE, vol. 1, 2013, pp. 275–278.

[124] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview", *Motion and operation planning of robotic systems: Background and practical approaches*, pp. 3–27, 2015.

[125] M. P. Garcia, O. Montiel, O. Castillo, R. Sepulveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation", *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, 2009.

[126] Y. Hu, S. X. Yang, L.-Z. Xu, and M.-H. Meng, "A knowledge based genetic algorithm for path planning in unstructured mobile robot environments", in *2004 IEEE international conference on robotics and biomimetics*, IEEE, 2004, pp. 767–772.

[127] H. Seraji and A. Howard, "Behavior-based robot navigation on challenging terrain: A fuzzy logic approach", *IEEE transactions on robotics and automation*, vol. 18, no. 3, pp. 308–321, 2002.

[128] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review", *Ieee access*, vol. 2, pp. 56–77, 2014.

[129] K. Kanjanawanishkul, "Motion control of a wheeled mobile robot using model predictive control: A survey", *KKU Research Journal*, vol. 17, no. 5, pp. 811–837, 2012.

[130] T. P. Nascimento, C. E. Dórea, and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: A survey", *Robotica*, vol. 36, no. 5, pp. 676–696, 2018.

[131] M. Pecka, V. Šalanskỳ, K. Zimmermann, and T. Svoboda, "Autonomous flipper control with safety constraints", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 2889–2894.

[132] M. Pecka, K. Zimmermann, M. Petrlík, and T. Svoboda, "Data-driven policy transfer with imprecise perception simulation", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3916–3921, 2018.

[133] J. Berger and N. Lo, "An innovative multi-agent search-and-rescue path planning approach", *Computers & Operations Research*, vol. 53, pp. 24–31, 2015.

[134] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments", *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[135] Y. Zhou and N. Mou, "Research on 2D obstacle avoidance model based on field rescue", in *Proceedings of the 2024 6th International Conference on Mechatronics Systems and Control Engineering*, 2024, pp. 6–10.

[136] Y. Chen, D. Shi, H. Yang, T. Li, and Z. Wang, "An anti-collision algorithm for robotic search-and-rescue tasks in unknown dynamic environments", *Frontiers of Information Technology & Electronic Engineering*, vol. 25, no. 4, pp. 569–584, 2024.

[137] G. Kumar, A. Anwar, A. Dikshit, A. Poddar, U. Soni, and W. K. Song, "Obstacle avoidance for a swarm of unmanned aerial vehicles operating on particle swarm optimization: A swarm intelligence approach for search and rescue missions", *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 44, no. 2, p. 56, 2022.

[138] M. Keidar and G. A. Kaminka, "Efficient frontier detection for robot exploration", *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 215–236, 2014.

[139] T. Rouček, M. Pecka, P. Čížek, *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments", in *Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6*, Springer, 2020, pp. 274–290.

[140] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[141] I. Lluvia, E. Lazkano, and A. Ansuategi, "Active mapping and robot exploration: A survey", *Sensors*, vol. 21, no. 7, p. 2445, 2021.

[142] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. Von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots", in *Robot Soccer World Cup*, Springer, 2013, pp. 624–631.

[143] M. Lewis, K. Sycara, and I. Nourbakhsh, "Developing a testbed for studying human-robot interaction in urban search and rescue", in *Human-Centered Computing*, CRC Press, 2019, pp. 270–274.

[144] J. Cacace, A. Finzi, and V. Lippiello, "Implicit robot selection for human multi-robot interaction in search and rescue missions", in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2016, pp. 803–808.

[145] S. Tejada, A. Cristina, P. Goodwyne, E. Normand, R. O'Hara, and S. Tarapore, "Virtual Synergy: A human-robot interface for urban search and rescue.", in *Aaai mobile robot competition*, 2003, pp. 13–19.

[146] T. Korhonen and S. Hostikka, "Fire dynamics simulator with evacuation: FDS+Evac (version 5)", *VTT Technical Research Centre of Finland*, 2014.

[147] L. Wilmes and A. Jamshidnejad, "GT-BDI model: A combined game-theoretic and BDI-based computational model for emergency evacuation with search and rescue robots", *IEEE Access*, 2025.

[148] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents", in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 2871–2878.

[149] Y.-Q. Jiang, W. Zhang, and S.-G. Zhou, "Comparison study of the reactive and predictive dynamic models for pedestrian flow", *Physica A: Statistical Mechanics and its Applications*, vol. 441, pp. 51–61, 2016.

[150] J. G. Freire and C. C. DaCamara, "Using cellular automata to simulate wildfire propagation and to assist in fire management", *Natural hazards and earth system sciences*, vol. 19, no. 1, pp. 169–179, 2019.

[151] A. Ohgai, Y. Gohnai, and K. Watanabe, "Cellular automata modeling of fire spread in built-up areas—A tool to aid community-based planning for disaster mitigation", *Computers, environment and urban systems*, vol. 31, no. 4, pp. 441–460, 2007.

[152] M. Khadem, J. O'Neill, Z. Mitros, L. Da Cruz, and C. Bergeles, "Autonomous steering of concentric tube robots via nonlinear model predictive control", *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1595–1602,

[153] E. Skjong, S. A. Nundal, F. S. Leira, and T. A. Johansen, "Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and avionics", in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 904–913.

[154] A. Pandey, S. Pandey, and D. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review", *Int Rob Auto J*, vol. 2, no. 3, p. 00 022, 2017.

[155] T. Ohki, K. Nagatani, and K. Yoshida, "Collision avoidance method for mobile robot considering motion and personal spaces of evacuees", in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 1819–1824.

[156] Z. Liu, M. Li, D. Fu, and S. Zhang, "Design of intelligent controller for obstacle avoidance and navigation of electric patrol mobile robot based on PLC", *Scientific Reports*, vol. 14, no. 1, p. 13 476, 2024.

[157] M. Zorn, "Natural disasters and less developed countries", in *Nature, tourism and ethnicity as drivers of (de) marginalization: Insights to marginality from perspective of sustainability and development*, Springer, 2017, pp. 59–78.

[158] S. Badrloo, M. Varshosaz, S. Pirasteh, and J. Li, "Image-based obstacle detection methods for the safe navigation of unmanned vehicles: A review", *Remote Sensing*, vol. 14, no. 15, p. 3824, 2022.

[159] E. Shreyas, M. H. Sheth, *et al.*, "3D object detection and tracking methods using deep learning for computer vision applications", in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, IEEE, 2021, pp. 735–738.

[160] A. Rom and I. Kelman, "Search without rescue? Evaluating the international search and rescue response to earthquake disasters", *BMJ global health*, vol. 5, no. 12, e002398, 2020.

[161] S. Meraglia and M. Lovera, "Smoother-based iterative learning control for UAV trajectory tracking", *IEEE Control Systems Letters*, vol. 6, pp. 1501–1506, 2021.

[162] K. B. Kidambi, C. Fermüller, Y. Aloimonos, and H. Xu, "Robust nonlinear control-based trajectory tracking for quadrotors under uncertainty", *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2042–2047, 2020.

[163] Z. Huang, Q. Liu, J. Liu, and B. Huang, "A comparative study of model approximation methods applied to economic MPC", *The Canadian Journal of Chemical Engineering*, vol. 100, no. 8, pp. 1676–1702, 2022.

[164] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees", *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[165] C. Nattero, C. T. Recchiuto, A. Sgorbissa, and F. Wanderlingh, "Coverage algorithms for search and rescue with UAV drones", in *Artificial Intelligence, Workshop of the XIII AI* IA Symposium on*, vol. 12, 2014.

[166] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics", *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[167] C. Cadena, L. Carlone, H. Carrillo, *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age", *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[168] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system", *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[169] M. Chandarana, D. Hughes, M. Lewis, K. Sycara, and S. Scherer, "Planning and monitoring multi-job type swarm search and service missions", *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–14, 2021.

[170] Z. Kashino, G. Nejat, and B. Benhabib, "Aerial wilderness search and rescue with ground support", *Journal of Intelligent & Robotic Systems*, vol. 99, pp. 147–163, 2020.

[171] A. M. Khamis, A. M. Elmogy, and F. O. Karray, "Complex task allocation in mobile surveillance systems", *Journal of Intelligent & Robotic Systems*, vol. 64, pp. 33–55, 2011.

[172] Z. Husain, A. A. Zaabi, H. Hildmann, F. Saffre, D. Ruta, and A. F. Isakovic, "Search and rescue in a maze-like environment with ant and dijkstra algorithms", *arXiv preprint arXiv:2111.08882*, 2021.

[173] J. R. Cooper, "Optimal multi-agent search and rescue using potential field theory", in *AIAA Scitech 2020 Forum*, 2020, p. 0879.

[174] R. Dubé, A. Gawel, C. Cadena, R. Siegwart, L. Freda, and M. Gianni, "3D localization, mapping and path planning for search and rescue operations", in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, 2016, pp. 272–273.

[175] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 3105–3112.

[176] F. Mohseni, A. Doustmohammadi, and M. B. Menhaj, "Distributed model predictive coverage control for decoupled mobile robots", *Robotica*, vol. 35, no. 4, pp. 922–941, 2017.

[177] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen, "Hierarchical model predictive control for autonomous vehicle area coverage", *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 79–84, 2019.

[178] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen, "Improved area covering in dynamic environments by nonlinear model predictive path following control", *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 418–423, 2019.

[179] S. Agarwal and S. Akella, "Area coverage with multiple capacity-constrained robots", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3734–3741, 2022.

[180] K. R. Guruprasad and T. D. Ranjitha, "CPC algorithm: Exact area coverage by a mobile robot using approximate cellular decomposition", *Robotica*, vol. 39, no. 7, pp. 1141–1162, 2021.

[181] C. Carr and P. Wang, "Fast-spanning ant colony optimisation (FaSACO) for mobile robot coverage path planning", *arXiv preprint arXiv:2205.15691*, 2022.

[182] V. S. Juan, M. Santos, and J. M. Andújar, "Intelligent UAV map generation and discrete path planning for search and rescue operations", *Complexity*, vol. 2018, 2018.

[183] D. Paez, J. P. Romero, B. Noriega, G. A. Cardona, and J. M. Calderon, "Distributed particle swarm optimization for multi-robot system in search and rescue operations", *IFAC-PapersOnLine*, vol. 54, no. 4, pp. 1–6, 2021.

[184] S. Kazemdehbashi and Y. Liu, "An exact coverage path planning algorithm for UAV-based search and rescue operations", *arXiv preprint arXiv:2405.11399*, 2024.

[185] Y. Liu and G. Nejat, "Multirobot cooperative learning for semiautonomous control in urban search and rescue applications", *Journal of Field Robotics*, vol. 33, no. 4, pp. 512–536, 2016.

[186] A. Hong, O. Igharoro, Y. Liu, F. Niroui, G. Nejat, and B. Benhabib, "Investigating human-robot teams for learning-based semi-autonomous control in urban search and rescue environments", *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3, pp. 669–686, 2019.

[187] K. G. S. Apuroop, A. V. Le, M. R. Elara, and B. J. Sheu, "Reinforcement learning-based complete area coverage path planning for a modified hTrihex robot", *Sensors*, vol. 21, no. 4, p. 1067, 2021.

[188] O. Saha, G. Ren, J. Heydari, V. Ganapathy, and M. Shah, "Deep reinforcement learning based online area covering autonomous robot", in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, Prague, Czech Republic, 2021, pp. 21–25.

[189] O. Saha, G. Ren, J. Heydari, V. Ganapathy, and M. Shah, "Online area covering robot in unknown dynamic environments", in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, Prague, Czech Republic, 2021, pp. 38–42.

[190] A. Carron and M. N. Zeilinger, "Model predictive coverage control", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6107–6112, 2020.

[191] M. Hoy, A. S. Matveev, and A. V. Savkin, "Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments", *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1253–1266, 2012.

[192] M. Farrokhsiar, G. Pavlik, and H. Najjaran, "An integrated robust probing motion planning and control scheme: A tube-based MPC approach", *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1379–1391, 2013.

[193] A. Jamshidnejad and E. Frazzoli, "Adaptive optimal receding-horizon robot navigation via short-term policy development", in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, Singapore, 2018, pp. 21–28.

[194] X. Cao, M. Li, Y. Tao, and P. Lu, "HMA-SAR: Multi-agent search and rescue for unknown located dynamic targets in completely unknown environments", *IEEE Robotics and Automation Letters*, 2024.

[195] N. Li and S. I. Han, "Adaptive bi-directional RRT algorithm for three-dimensional path planning of unmanned aerial vehicles in complex environments", *IEEE Access*, 2025.

[196] X. Zhang, W. Zhao, C. Liu, and J. Li, "Distributed multi-target search and surveillance mission planning for unmanned aerial vehicles in uncertain environments", *Drones*, vol. 7, no. 6, p. 355, 2023.

[197] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics: A survey", *Autonomous robots*, vol. 31, pp. 299–316, 2011.

[198] A. Pierson and D. Rus, "Distributed target tracking in cluttered environments with guaranteed collision avoidance", in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, 2017, pp. 83–89.

[199] M. Sani, B. Robu, and A. Hably, "Pursuit-evasion games based on game-theoretic and model predictive control algorithms", in *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*, IEEE, 2021, pp. 1–6.

[200] M. Sani, B. Robu, and A. Hably, "Limited information model predictive control for pursuit-evasion games", in *2021 60th IEEE Conference on Decision and Control (CDC)*, IEEE, 2021, pp. 265–270.

[201] D. D. Simone, N. Scianca, P. P. Ferrari, L. Lanari, and G. Oriolo, "MPC-based humanoid pursuit-evasion in the presence of obstacles", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 5245–5250.

[202] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation", *arXiv preprint arXiv:1903.01959*, 2019.

[203] A. H. Tan, S. Narasimhan, and G. Nejat, "4CNet: A confidence-aware, contrastive, conditional, consistency model for robot map prediction in multi-robot environments", *arXiv preprint arXiv:2402.17904*, 2024.

[204] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural SLAM", *arXiv preprint arXiv:2004.05155*, 2020.

[205] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep reinforcement learning for decentralized multi-robot exploration with macro actions", *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 272–279, 2022.

[206] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi, "Towards generalization in target-driven visual navigation by using deep reinforcement learning", *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1546–1561, 2020.

[207] L. Mezghan, S. Sukhbaatar, T. Lavril, *et al.*, "Memory-augmented reinforcement learning for image-goal navigation", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 3316–3323.

[208] A. Fung, L. Y. Wang, K. Zhang, G. Nejat, and B. Benhabib, "Using deep learning to find victims in unknown cluttered urban search and rescue environments", *Current Robotics Reports*, vol. 1, pp. 105–115, 2020.

[209] H. Wang, A. H. Tan, and G. Nejat, "NavFormer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments", *IEEE Robotics and Automation Letters*, 2024.

[210] S. C. Mohamed, A. Fung, and G. Nejat, "A multirobot person search system for finding multiple dynamic users in human-centered environments", *IEEE Transactions on Cybernetics*, vol. 53, no. 1, pp. 628–640, 2022.

[211] N. H. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "VLFM: Vision-language frontier maps for zero-shot semantic navigation", in *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[212] E. Ronchi and D. Nilsson, "Fire evacuation in high-rise buildings: A review of human behaviour and modelling research", *Fire science reviews*, vol. 2, no. 1, p. 7, 2013.

[213] E. Ronchi, E. D. Kuligowski, P. A. Reneke, R. D. Peacock, and D. Nilsson, "The process of verification and validation of building fire evacuation models", 2013.

[214] R. Lovreglio, E. Kuligowski, S. Gwynne, and K. Boyce, "A pre-evacuation database for use in egress simulations", *Fire safety journal*, vol. 105, pp. 107–128, 2019.

[215] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics", *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[216] C.-T. Chen, *Linear system theory and design*. Saunders college publishing, 1984.

[217] D. Q. Mayne and E. C. Kerrigan, "Tube-based robust nonlinear model predictive control", *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 36–41, 2007.

[218] Y. Wang and S. Boyd, "Fast model predictive control using online optimization", *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.

[219] P. Kumar, B. B. Anoohya, and R. Padhi, "Model predictive static programming for optimal command tracking: A fast model predictive control paradigm", *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 2, p. 021 014, 2019.

[220] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes", *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[221] Robotis, *TurtleBot3*, https://www.robotis.us/turtlebot-3/.

[222] Robotis, *Robotis-git/turtlebot3*, https://github.com/ROBOTIS-GIT/turtlebot3.

[223] M. Baglioni, *Robot and victims trajectories in search-and-rescue scenario*, https://doi.org/10.4121/22270498, 2023.

[224] A. Tringali and S. Cocuzza, "Globally optimal inverse kinematics method for a redundant robot manipulator with linear and nonlinear constraints", *Robotics*, vol. 9, no. 3, p. 61, 2020.

[225] S. Dhouib, "Hierarchical coverage repair policies optimization by Dhouib-Matrix-4 metaheuristic for wireless sensor networks using mobile robot", *International Journal of Engineering*, vol. 36, no. 12, pp. 2153–2160, 2023.

[226] J. Li, M. Ran, H. Wang, and L. Xie, "MPC-based unified trajectory planning and tracking control approach for automated guided vehicles", in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, IEEE, 2019, pp. 374–380.

[227] A. Brooks, T. Kaupp, and A. Makarenko, "Randomised MPC-based motion-planning for mobile robot obstacle avoidance", in *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 3962–3967.

[228] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[229] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "BA*: An online complete coverage algorithm for cleaning robots", *Applied intelligence*, vol. 39, no. 2, pp. 217–235, 2013.

[230] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition", in *Field and Service Robotics*, A. Zelinsky, Ed., London: Springer, 1998, pp. 203–209.

[231] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation", *Nonlinear model predictive control: towards new challenging applications*, pp. 391–417, 2009.

[232] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer, "A computationally efficient robust model predictive control framework for uncertain nonlinear systems", *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 794–801, 2020.

[233] A. Boccia, L. Grüne, and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints", *Systems & control letters*, vol. 72, pp. 14–21, 2014.

[234] X. Fang and W.-H. Chen, "Model predictive control with preview: Recursive feasibility and stability", *IEEE Control Systems Letters*, vol. 6, pp. 2647–2652, 2022.

[235] iRobot Education, *iRobot Create 3*, https://edu.irobot.com/what-we-offer/create3.

[236] R. R. Murphy, *Disaster Robotics. Intelligent robotics and autonomous agents series*. The MIT press, 2014.

[237] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions", *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[238] R. Nagasawa, E. Mas, L. Moya, and S. Koshimura, "Model-based analysis of multi-UAV path planning for surveying postdisaster building damage", *Scientific Reports*, vol. 11, no. 1, p. 18 588, 2021.

[239] M. Serra, P. Sathe, I. Rypina, *et al.*, "Search and rescue at sea aided by hidden flow structures", *Nature Communications*, vol. 11, no. 1, p. 2525, 2020.

[240] D. C. Schedl, I. Kurmi, and O. Bimber, "Search and rescue with airborne optical sectioning", *Nature Machine Intelligence*, vol. 2, no. 12, pp. 783–790, 2020.

[241] R. Arnold, J. Jablonski, B. Abruzzo, and E. Mezzacappa, "Heterogeneous UAV multi-role swarming behaviors for search and rescue", in *2020 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, IEEE, 2020, pp. 122–128.

[242] V. San Juan, M. Santos, and J. M. Andújar, "Intelligent UAV map generation and discrete path planning for search and rescue operations", *Complexity*, vol. 2018, no. 1, p. 6 879 419, 2018.

[243] G. Loukas and S. Timotheou, "Connecting trapped civilians to a wireless ad hoc network of emergency response robots", in *2008 11th IEEE Singapore International Conference on Communication Systems*, IEEE, 2008, pp. 599–603.

[244] W. Stecz and K. Gromada, "UAV mission planning with SAR application", *Sensors*, vol. 20, no. 4, p. 1080, 2020.

[245] Z. Liu and O. Stursberg, "Recursive feasibility and stability of MPC with time-varying and uncertain state constraints", in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 1766–1771.

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

[247] T. Belvedere, M. Cognetti, G. Oriolo, and P. R. Giordano, "Sensitivity-aware model predictive control for robots with parametric uncertainty", *IEEE Transactions on Robotics*, 2025.

[248] E. Soleimani, A. Nikoofard, and E. Nejabat, "Cascade explicit tube model predictive controller: Application for a multi-robot system", *Control Theory and Technology*, pp. 1–16, 2025.

[249] Y. Guo, A. Song, J. Bao, T. Hongru, and J. Cui, "A combination of terrain prediction and correction for search and rescue robot autonomous navigation", *International Journal of Advanced Robotic Systems*, vol. 6, no. 3, p. 24, 2009.

[250] Y. Xue and W. Chen, "RLoPlanner: Combining learning and motion planner for UAV safe navigation in cluttered unknown environments", *IEEE Transactions on Vehicular Technology*, vol. 73, no. 4, pp. 4904–4917, 2023.

[251] W. Chen and Y. Xue, "Hierarchical UAV autonomous navigation algorithm based on event-triggered deep reinforcement learning", *IEEE Transactions on Vehicular Technology*, 2025.

[252] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using Lagrange and Newton-Euler methodologies: A unified framework", *Advances in Robotics & Automation*, vol. 2, no. 2, pp. 1–7, 2013.

[253] R. Scattolini, "Architectures for distributed and hierarchical model predictive control - a review", *Journal of Process Control*, vol. 19, pp. 723–731, 2009.

[254] M. Basescu and J. Moore, "Direct NMPC for post-stall motion planning with fixed-wing UAVs", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9592–9598.

[255] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT press, 1998.

[256] V. Torczon, "On the convergence of pattern search algorithms", *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[257] Y. Chen, Z. He, and S. Li, "Horizon-based lazy optimal RRT for fast, efficient replanning in dynamic environment", *Autonomous Robots*, vol. 43, no. 8, pp. 2271–2292, 2019.

[258] H. Lyu and Y. Yin, "COLREGS-constrained real-time path planning for autonomous ships using modified artificial potential fields", *The Journal of Navigation*, vol. 72, no. 3, pp. 588–608, 2019.

[259] M. Baglioni, *Tables with parameters values underlying the publication: Enabling robots to autonomously search dynamic cluttered post-disaster environments*, https://doi.org/10.4121/aa7528da-0986-453c-b196-4277a2db4daa, 2024.

[260] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation", in *Robot Soccer World Cup*, Springer, 2002, pp. 288–295.

[261] M.-S. Yang, C.-Y. Lai, and C.-Y. Lin, "A robust EM clustering algorithm for Gaussian mixture models", *Pattern Recognition*, vol. 45, no. 11, pp. 3950–3961, 2012.

[262] J. Qi, D. Song, H. Shang, *et al.*, "Search and rescue rotary-wing UAV and its application to the Lushan Ms 7.0 earthquake", *Journal of Field Robotics*, vol. 33, no. 3, pp. 290–321, Jul. 2016.

[263] C. D. Rodin, L. N. de Lima, F. A. de Alcantara Andrade, D. B. Haddad, T. A. Johansen, and R. Storvold, "Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems", in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil: IEEE, Oct. 2018, pp. 1–8.

[264] J. McGee, S. J. Mathew, and F. Gonzalez, "Unmanned aerial vehicle and artificial intelligence for thermal target detection in search and rescue applications", in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, Athens, Greece: IEEE, Sep. 2020, pp. 883–891.

[265] J. P. Zelten, "Digital photography and the dynamics of technology innovation", Ph.D. dissertation, Massachusetts Institute of Technology, Boston, MA, USA, Feb. 2002.

[266] S. Hummel, A. Hudak, E. Uebler, M. Falkowski, and K. Megown, "A comparison of accuracy and cost of LiDAR versus stand exam data for landscape management on the Malheur National Forest", *Journal of Forestry*, vol. 109, pp. 267–273, Jul. 2011.

[267] L. Zhaohua and G. Bochao, "Radar sensors in automatic driving cars", in *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, Nanchang, China: IEEE, Oct. 2020, pp. 239–242.

[268] D. Falconer, R. Ficklin, and K. Konolige, "Robot-mounted through-wall radar for detecting, locating, and identifying building occupants", in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, San Francisco, CA, USA: IEEE, Apr. 2000, pp. 1868–1875.

[269] J. Geisheimer, W. Marshall, and E. Greneker, "A continuous-wave (CW) radar for gait analysis", in *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, USA: IEEE, Nov. 2001, pp. 834–838.

[270] Q. Zhou, J. Liu, A. Host-Madsen, O. Boric-Lubecke, and V. Lubecke, "Detection of multiple heartbeats using Doppler radar", in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 2, Toulouse, France: IEEE, May 2006, pp. II–II.

[271] I. C. Condotta, T. M. Brown-Brandl, S. K. Pitla, J. P. Stinn, and K. O. Silva-Miranda, "Evaluation of low-cost depth cameras for agricultural applications", *Computers and Electronics in Agriculture*, vol. 173, p. 105 394, Jun. 2020.

[272] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "Audiomoth: A low-cost acoustic device for monitoring biodiversity and the environment", *HardwareX*, vol. 6, e00073, Oct. 2019.

[273] H. H. Titi, *Feasibility Study for a Freeway Corridor Infrastructure Health Monitoring (HM) Instrumentation Testbed*. Madison, WI, USA: Wisconsin DOT Research & Library Unit, Jul. 2012.

[274] D. Van Nam and K. Gon-Woo, "Solid-state LiDAR based-SLAM: A concise review and application", in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju Island, Korea (South): IEEE, Jan. 2021, pp. 302–305.

[275] P. Beňo, F. Duchoň, P. Hubinský, M. Dekan, M. Tölgyessy, and M. Dobiš, "RGBD mapping solution for low-cost robot", *Machine Vision and Applications*, vol. 33, p. 21, 2022.

[276] P. Fankhauser, M. Bloesch, P. Krüsi, *et al.*, "Collaborative navigation for flying and walking robots", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (South): IEEE, Dec. 2016, pp. 2859–2866.

[277] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 664–671, Jan. 2017.

[278] M. Adel Musallam, R. Baptista, K. Al Ismaeil, and D. Aouada, "Temporal 3D human pose estimation for action recognition from arbitrary viewpoints", in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, Dec. 2019, pp. 253–258.

[279] N. D. Nath, C.-S. Cheng, and A. H. Behzadan, "Drone mapping of damage information in GPS-denied disaster sites", *Advanced Engineering Informatics*, vol. 51, p. 101 450, Jan. 2022.

[280] L. Xing, X. Fan, Y. Dong, *et al.*, "Multi-UAV cooperative system for search and rescue based on YOLOv5", *International Journal of Disaster Risk Reduction*, vol. 76, p. 102 972, Jun. 2022.

[281] M. B. Bejiga, A. Zeggada, and F. Melgani, "Convolutional neural networks for near real-time object detection from UAV imagery in avalanche search and rescue operations", in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, China: IEEE, Jul. 2016, pp. 693–696.

[282] V. A. Feraru, R. E. Andersen, and E. Boukas, "Towards an autonomous UAV-based system to assist search and rescue operations in man overboard incidents", in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Abu Dhabi, United Arab Emirates: IEEE, Nov. 2020, pp. 57–64.

[283] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Search and rescue operation using UAVs: A case study", *Expert Systems with Applications*, vol. 178, p. 114 937, Sep. 2021.

[284] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of Yolo algorithm developments", *Procedia Computer Science*, vol. 199, pp. 1066–1073, Oct. 2022.

[285] Y. I. Putra, A. H. Alasiry, A. Darmawan, H. Oktavianto, and Z. M. E. Darmawan, "Camera-based object detection and identification using YOLO method for Indonesian search and rescue robot competition", in *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Dec. 2022, pp. 508–513.

[286] V. Kodipaka, L. Marques, R. Cortesão, and H. Araújo, "APH-YOLOv7t: A YOLO attention prediction head for search and rescue with drones", in *Iberian Robotics conference*, Springer, 2023, pp. 256–268.

[287] N. Bachir and Q. A. Memon, "Benchmarking YOLOv5 models for improved human detection in search and rescue missions", *Journal of Electronic Science and Technology*, vol. 22, no. 1, p. 100 243, 2024.

[288] H.-Y. Chien, Z.-W. Huang, W.-Y. Chen, Y.-C. Chang, and C.-C. Sun, "The YOLO object detection and thermal imaging in natural disaster environments advanced intelligent rescue system", in *IET Conference Proceedings CP840*, IET, vol. 35, 2023, pp. 134–135.

[289] Z. Jin, T. He, L. Qiao, *et al.*, "MES-YOLO: An efficient lightweight maritime search and rescue object detection algorithm with improved feature fusion pyramid network", *Journal of Visual Communication and Image Representation*, p. 104 453, 2025.

[290] S. Vasuhi, M. Vijayakumar, and V. Vaidehi, "Real time multiple human tracking using Kalman filter", in *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India: IEEE, Mar. 2015, pp. 1–6.

[291] P. Li, T. Zhang, and B. Ma, "Unscented Kalman filter for visual curve tracking", *Image and Vision Computing*, vol. 22, no. 2, pp. 157–164, Feb. 2004.

[292] Y. Tian, Y. Luo, S. Zhou, K. Zhang, and Y. Wang, "A Kalman filter-based submersible position prediction model and a multi-target dynamic search and rescue scheme", in *2024 18th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2024, pp. 320–325.

[293] S. Pang, B. Zhang, J. Lu, *et al.*, "Application of IMU/GPS integrated navigation system based on adaptive unscented kalman filter algorithm in 3D positioning of forest rescue personnel", *Sensors*, vol. 24, no. 18, p. 5873, 2024.

[294] S. Yeom, "Thermal image tracking for search and rescue missions with a drone", *Drones*, vol. 8, no. 2, p. 53, 2024.

[295] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment", *Journal of Robotic Systems*, vol. 18, no. 10, pp. 577–587, Apr. 2001.

[296] L. Du, Y. Pang, W. Ni, *et al.*, "Forest terrain and canopy height estimation using stereo images and spaceborne LiDAR data from GF-7 satellite", *Geo-spatial Information Science*, vol. 27, no. 3, pp. 811–821, 2024.

[297] J. Xia and J. Gong, "Computer vision based first floor elevation estimation from mobile LiDAR data", *Automation in Construction*, vol. 159, p. 105 258, 2024.

[298] G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics*, version 8.0.0, Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[299] C. D. Fryar, Q. Gu, C. L. Ogden, and K. M. Flegal, "Anthropometric reference data for children and adults; United States, 2011-2014", *Vital and health statistics. Series 3, Data from the National Health and Nutrition Examination Survey*, vol. 39, Aug. 2016.

[300] S. Verykokou, A. Doulamis, G. Athanasiou, C. Ioannidis, and A. Amditis, "UAV-based 3D modelling of disaster scenes for urban search and rescue", in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*, IEEE, 2016, pp. 106–111.

[301] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques", *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 601–627, 2019.

[302] R. R. Murphy, S. Tadokoro, D. Nardi, *et al.*, "Search and rescue robotics", in Springer, 2008, ch. 50, pp. 1151–1173, ISBN: 978-3-540-30301-5.

[303] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning", *Advances in neural information processing systems*, vol. 26, 2013.

[304] N. Geng, Q. Meng, D. Gong, and P. W. Chung, "How good are distributed allocation algorithms for solving urban search and rescue problems? a comparative study with centralized algorithms", *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 478–485, 2018.

[305] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "LSAR: Multi-UAV collaboration for search and rescue missions", *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.

[306] Y. Tian, K. Liu, K. Ok, *et al.*, "Search and rescue under the forest canopy using multiple UAVs", *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[307] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams", *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[308] V. Gervasi and G. Prencipe, "Coordination without communication: The case of the flocking problem", *Discrete Applied Mathematics*, vol. 144, no. 3, pp. 324–344, 2004.

[309] P. Schermerhorn and M. Scheutz, "Social coordination without communication in multi-agent territory exploration tasks", in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006, pp. 654–661.

[310] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita, "Rendezvous of two robots with constant memory", in *Structural Information and Communication Complexity: 20th International Colloquium, SIROCCO 2013, Ischia, Italy, July 1-3, 2013, Revised Selected Papers 20*, Springer, 2013, pp. 189–200.

[311] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems", *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013.

[312] L. Yan, T. Stouraitis, and S. Vijayakumar, "Decentralized ability-aware adaptive control for multi-robot collaborative manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2311–2318, 2021.

[313] Y. Zhai, B. Ding, X. Liu, H. Jia, Y. Zhao, and J. Luo, "Decentralized multi-robot collision avoidance in complex scenarios with selective communication", *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8379–8386, 2021.

[314] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration", *IEEE/ASME transactions on mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[315] J. Scherer, S. Yahyanejad, S. Hayat, *et al.*, "An autonomous multi-uav system for search and rescue", in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, 2015, pp. 33–38.

[316] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective UAV path planning for search and rescue", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5569–5574.

[317] S. Hayat, E. Yanmaz, C. Bettstetter, and T. X. Brown, "Multi-objective drone path planning for search and rescue with quality-of-service requirements", *Autonomous Robots*, vol. 44, no. 7, pp. 1183–1198, 2020.

[318] A. Nath, A. Arun, and R. Niyogi, "A distributed approach for road clearance with multi-robot in urban search and rescue environment", *International journal of intelligent robotics and applications*, vol. 3, pp. 392–406, 2019.

[319] M. Boldrer, F. Pasqualetti, L. Palopoli, and D. Fontanelli, "Multiagent persistent monitoring via time-inverted kuramoto dynamics", *IEEE Control Systems Letters*, vol. 6, pp. 2798–2803, 2022.

[320] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control", in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 2872–2879.

[321] K. Elamvazhuthi and S. Berman, "Optimal control of stochastic coverage strategies for robotic swarms", in *2015 ieee international conference on robotics and automation (icra)*, IEEE, 2015, pp. 1822–1829.

[322] S. Stavrinidis, P. Zacharia, and E. Xidias, "A fuzzy control strategy for multi-goal autonomous robot navigation", *Sensors*, vol. 25, no. 2, p. 446, 2025.

[323] L. C. Sousa, Y. M. Silva, V. B. Schettino, *et al.*, "Obstacle avoidance technique for mobile robots at autonomous human-robot collaborative warehouse environments", *Sensors*, vol. 25, no. 8, p. 2387, 2025.

[324] S. Mukherjee and S. Borah, "A novel hybrid fuzzy controller for path navigation in static and dynamic environments", *International Journal of Intelligent Robotics and Applications*, pp. 1–16, 2025.

[325] N. E. Daidzic, "General solution of the wind triangle problem and the critical tail-wind angle", *The International Journal of Aviation Sciences (IJAS)*, vol. 1, no. 1, pp. 57–93, 2016.

[326] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[327] L.-X. Wang, "A supervisory controller for fuzzy control systems that guarantees stability", in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, IEEE, 1994, pp. 1035–1039.

[328] R. Scattolini and P. Colaneri, "Hierarchical model predictive control", in *2007 46th IEEE conference on decision and control*, IEEE, 2007, pp. 4803–4808.

[329] J. H. Lilly, "Fuzzy control and identification", in John Wiley & Sons, 2011, ch. Takagi-Sugeno Fuzzy Systems, pp. 88–105.

[330] V. M. Peri and D. Simon, "Fuzzy logic control for an autonomous robot", in *NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society*, IEEE, 2005, pp. 337–342.

[331] C. Maxwell, *Code for model-predictive fuzzy controller for search-and-rescue path-planning of multi-agent systems*, https://doi.org/10.4121/8050f9cb-d0b0-4149-bd24-02f13c2410db.v1, 2024.

[332] C. Maxwell, *Integrated model predictive fuzzy control for disaster victim detection path planning in MATLAB*, https://github.com/craigmax-dev/Integrated-Model-Predictive-Fuzzy-Control-for-Disaster-Victim-Detection-Path-Planning-in-MATLAB, 2024.

[333] S. S. Mansouri, C. Kanellakis, G. Georgoulas, D. Kominiak, T. Gustafsson, and G. Nikolakopoulos, "2D visual area coverage and path planning coupled with camera footprints", *Control Engineering Practice*, vol. 75, pp. 1–16, 2018.

[334] B. Gravell and T. Summers, "Centralized collision-free polynomial trajectories and goal assignment for aerial swarms", *Control Engineering Practice*, vol. 109, p. 104 753, 2021.

[335] C. A. Toro-Arcila, H. M. Becerra, and G. Arechavaleta, "Visual path following with obstacle avoidance for quadcopters in indoor environments", *Control Engineering Practice*, vol. 135, p. 105 493, 2023.

[336] B. Németh and P. Gáspár, "Hierarchical motion control strategies for handling interactions of automated vehicles", *Control Engineering Practice*, vol. 136, p. 105 523, 2023.

[337] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control", *Autonomous Robots*, vol. 40, pp. 17–31, 2016.

[338] W. Sun, Y. Yuan, and H. Gao, "Hierarchical control for partially feasible tasks with arbitrary dimensions: Stability analysis for the tracking case", *IEEE Transactions on Automatic Control*, 2024.

[339] D. De Benedittis, M. Garabini, and L. Pallottino, "Managing conflicting tasks in heterogeneous multi-robot systems through hierarchical optimization", *IEEE Robotics and Automation Letters*, 2025.

[340] B. Zhang, Z. Liu, J. Hui, and P. Huang, "Hierarchical control of manipulator with null-space compliance at the kinematic level", *Control Engineering Practice*, vol. 142, p. 105 770, 2024.

[341] G. Xin, H.-C. Lin, J. Smith, O. Cebe, and M. Mistry, "A model-based hierarchical controller for legged systems subject to external disturbances", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Brisbane, QLD, Australia, 2018, pp. 4375–4382.

[342] C. Rosales, P. Leica, M. Sarcinelli-Filho, G. Scaglia, and R. Carelli, "3D formation control of autonomous vehicles based on null-space", *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 453–467, 2016.

[343] P. Cai, X. Yue, M. Wang, and Y. Cui, "Hierarchical motion planning at the acceleration level based on task priority matrix for space robot", *Nonlinear Dynamics*, vol. 107, no. 3, pp. 2309–2326, 2022.

[344] T. Kim, G. Park, K. Kwak, J. Bae, and W. Lee, "Smooth model predictive path integral control without smoothing", *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 406–10 413, 2022.

[345] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2864–2871, 2020.

[346] W. Zhu, X. Guo, Y. Fang, and X. Zhang, "A path-integral-based reinforcement learning algorithm for path following of an autoassembly mobile robot", *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4487–4499, 2019.

[347] E. Theodorou, F. Stulp, J. Buchli, and S. Schaal, "An iterative path integral stochastic optimal control approach for learning robotic tasks", *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 594–11 601, 2011.

[348] K. Honda, N. Akai, K. Suzuki, *et al.*, "Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles", in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 7020–7026.

[349] C. E. Schroeder, D. A. Wilson, T. Radman, H. Scharfman, and P. Lakatos, "Dynamics of active sensing and perceptual selection", *Current opinion in neurobiology*, vol. 20, no. 2, pp. 172–176, 2010.

[350] V. Cichella, T. Marinho, D. Stipanović, N. Hovakimyan, I. Kaminer, and A. Trujillo, "Collision avoidance based on line-of-sight angle: Guaranteed safety using limited information about the obstacle", *Journal of Intelligent & Robotic Systems*, vol. 89, pp. 139–153, 2018.

[351] X. Wu, Z. Feng, J. Zhu, and R. Allen, "Line of sight guidance with intelligent obstacle avoidance for autonomous underwater vehicles", in *OCEANS 2006*, IEEE, 2006, pp. 1–6.

[352] S. Moe and K. Y. Pettersen, "Set-based line-of-sight (los) path following with collision avoidance for underactuated unmanned surface vessels under the influence of ocean currents", in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2017, pp. 241–248.

[353] S. Kumari, S. Ghosh, D. Mitra, S. Sengupta, and S. Mukhopadhyay, "Collision risk assessment based on line of sight", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 972–14 977, 2020.

[354] H. Guo, Y.-H. Lee, Y.-L. Chen, H.-W. Tseng, and C.-F. Yang, "Simulation method for non-line-of-sight collision avoidance warning system", *Sensors and Materials*, vol. 35, no. 6, pp. 1919–1928, 2023.

[355] O. M. Bushnaq, A. Chaaban, and T. Y. Al-Naffouri, "The role of uav-iot networks in future wildfire detection", *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 984–16 999, 2021.

[356] S. Rabinovich, R. E. Curry, and G. H. Elkaim, "Toward dynamic monitoring and suppressing uncertainty in wildfire by multiple unmanned air vehicle system", *Journal of Robotics*, vol. 2018, no. 1, p. 6 892 153, 2018.

[357] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, "Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 835–840.

[358] D. A. Saikin, T. Baca, M. Gurtner, and M. Saska, "Wildfire fighting by unmanned aerial system exploiting its time-varying mass", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2674–2681, 2020.

[359] X. Chen, Z. Xiao, Y. Cheng, *et al.*, "SOScheduler: Toward proactive and adaptive wildfire suppression via multi-UAV collaborative scheduling", *IEEE Internet of Things Journal*, 2024.

[360] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The null-space-based behavioral control for autonomous robotic systems", *Intelligent Service Robotics*, vol. 1, pp. 27–39, 2008.

[361] XPRIZE, *XPRIZE Wildfire competition*, https://www.xprize.org/prizes/wildfire.

[362] J. Delmerico, S. Mintchev, A. Giusti, *et al.*, "The current state and future outlook of rescue robotics", *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[363] F. Santoni de Sio and J. Van den Hoven, "Meaningful human control over autonomous systems: A philosophical account", *Frontiers in Robotics and AI*, vol. 5, p. 323 836, 2018.

[364] M. A. Neerincx, W. Van Vught, O. Blanson Henkemans, *et al.*, "Socio-cognitive engineering of a robotic partner for child's diabetes self-management", *Frontiers in Robotics and AI*, vol. 6, p. 118, 2019.

[365] R. S. Verhagen, M. A. Neerincx, and M. L. Tielman, "Meaningful human control and variable autonomy in human-robot teams for firefighting", *Frontiers in Robotics and AI*, vol. 11, p. 1 323 980, 2024.

[366] Z. Akata, D. Balliet, M. De Rijke, *et al.*, "A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence", *Computer*, vol. 53, no. 8, pp. 18–28, 2020.

[367] G.-J. M. Kruijff, I. Kruijff-Korbayová, S. Keshavdas, *et al.*, "Designing, developing, and deploying systems to support human–robot teams in disaster response", *Advanced Robotics*, vol. 28, no. 23, pp. 1547–1570, 2014.

[368] I. Kruijff-Korbayová, F. Colas, M. Gianni, *et al.*, "Tradr project: Long-term human-robot teaming for robot assisted disaster response", *KI-Künstliche Intelligenz*, vol. 29, pp. 193–201, 2015.

[369] L. Frering, M. Eder, B. Kubicek, *et al.*, "Enabling and assessing trust when cooperating with robots in disaster response (easier)", *arXiv preprint arXiv:2207.03763*, 2022.

[370] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems", *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[371] M. R. Endsley, "Situation awareness global assessment technique (sagat)", in *Proceedings of the IEEE 1988 national aerospace and electronics conference*, IEEE, 1988, pp. 789–795.

[372] E. Salas, C. Prince, D. P. Baker, and L. Shrestha, "Situation awareness in team performance: Implications for measurement and training", *Human factors*, vol. 37, no. 1, pp. 123–136, 1995.

[373] A. A. Nofi, "Defining and measuring shared situational awareness", 2000.

[374] J. Y. Chen, S. G. Lakhmani, K. Stowers, A. R. Selkowitz, J. L. Wright, and M. Barnes, "Situation awareness-based agent transparency and human-autonomy teaming effectiveness", *Theoretical issues in ergonomics science*, vol. 19, no. 3, pp. 259–282, 2018.

[375] E. Salas, D. E. Sims, and C. S. Burke, "Is there a "big five" in teamwork?", *Small group research*, vol. 36, no. 5, pp. 555–599, 2005.

[376] J. E. Mercado, M. A. Rupp, J. Y. Chen, M. J. Barnes, D. Barber, and K. Procci, "Intelligent agent transparency in human–agent teaming for multi-uxv management", *Human factors*, vol. 58, no. 3, pp. 401–415, 2016.

[377] A. R. Selkowitz, S. G. Lakhmani, and J. Y. Chen, "Using agent transparency to support situation awareness of the autonomous squad member", *Cognitive Systems Research*, vol. 46, pp. 13–25, 2017.

[378] J. L. Wright, J. Y. Chen, M. J. Barnes, and P. A. Hancock, "The effect of agent reasoning transparency on complacent behavior: An analysis of eye movements and response performance", in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications Sage CA: Los Angeles, CA, vol. 61, 2017, pp. 1594–1598.

[379] L. Methnani, A. Aler Tubella, V. Dignum, and A. Theodorou, "Let me take over: Variable autonomy for meaningful human control", *Frontiers in Artificial Intelligence*, vol. 4, p. 737 072, 2021.

[380] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance", *Human factors*, vol. 46, no. 1, pp. 50–80, 2004.

[381] L. Cavalcante Siebert, M. L. Lupetti, E. Aizenberg, *et al.*, "Meaningful human control: Actionable properties for ai system development", *AI and Ethics*, vol. 3, no. 1, pp. 241–255, 2023.

[382] J. van der Waa, S. Verdult, K. van den Bosch, *et al.*, "Moral decision making in human-agent teams: Human control and the role of explanations", *Frontiers in Robotics and AI*, vol. 8, p. 640 647, 2021.

[383] B. Friedman and D. G. Hendry, *Value sensitive design: Shaping technology with moral imagination*. Mit Press, 2019.

[384] J. van Diggelen and M. Johnson, "Team design patterns", in *Proceedings of the 7th International Conference on Human-Agent Interaction*, 2019, pp. 118–126.

[385] K. Baum, S. Mantel, E. Schmidt, and T. Speith, "From responsibility to reason-giving explainable artificial intelligence", *Philosophy & Technology*, vol. 35, no. 1, p. 12, 2022.

[386] J. van Diggelen, K. van den Bosch, M. Neerincx, and M. Steen, "Designing for meaningful human control in military human-machine teams", in *Research handbook on meaningful human control of artificial intelligence systems*, Edward Elgar Publishing, 2024, pp. 232–252.

[387] J. van der Waa, J. van Diggelen, L. Cavalcante Siebert, M. Neerincx, and C. Jonker, "Allocation of moral decision-making in human-agent teams: A pattern approach", in *Engineering Psychology and Cognitive Ergonomics. Cognition and Design: 17th International Conference, EPCE 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*, Springer, 2020, pp. 203–220.

[388] R. Crootof, "A meaningful floor for meaningful human control", *Temp. Int'l & Comp. LJ*, vol. 30, p. 53, 2016.

[389] S. G. Tzafestas, "Mobile robot control and navigation: A global overview", *Journal of Intelligent & Robotic Systems*, vol. 91, pp. 35–58, 2018.

[390] H. Bozorgi and T. D. Ngo, "Beyond shared autonomy: Joint perception and action for human-in-the-loop mobile robot navigation systems", *Journal of Intelligent & Robotic Systems*, vol. 109, no. 1, p. 20, 2023.

[391] A. Franchi, C. Masone, and P. Robuffo Giordano, "A synergetic high-level/reactive planning framework with application to human-assisted navigation", in *2012 IEEE IROS Workshop on Robot Motion Planning: Online, Reactive, and in Real-time*, 2012, pp. 15–20.

[392] Y. Che, A. M. Okamura, and D. Sadigh, "Efficient and trustworthy social navigation via explicit and implicit robot–human communication", *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 692–707, 2020.

[393] P.-H. Ciou, Y.-T. Hsiao, Z.-Z. Wu, S.-H. Tseng, and L.-C. Fu, "Composite reinforcement learning for social robot navigation", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2553–2558.

[394] A. Sharma, A. Balasundaram, A. Shaik, and C. A. Vaithilingam, "A novel voice in head actor critic reinforcement learning with human feedback framework for enhanced robot navigation", *Scientific Reports*, vol. 15, no. 1, p. 7237, 2025.

[395] F. Ferracuti, A. Freddi, S. Iarlori, A. Monteriù, K. I. M. Omer, and C. Porcaro, "A human-in-the-loop approach for enhancing mobile robot navigation in presence of obstacles not detected by the sensory set", *Frontiers in Robotics and AI*, vol. 9, p. 909 971, 2022.

[396] K. Omer, F. Ferracuti, A. Freddi, S. Iarlori, A. Monteriù, and C. Porcaro, "Human-in-the-loop approach for enhanced mobile robot navigation", in *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, IEEE, 2022, pp. 416–421.

[397] H. Omrane, M. S. Masmoudi, and M. Masmoudi, "Fuzzy logic based control for autonomous mobile robot navigation", *Computational intelligence and neuroscience*, vol. 2016, no. 1, p. 9 548 482, 2016.

[398] N. H. Singh and K. Thongam, "Mobile robot navigation using fuzzy logic in static environments", *Procedia Computer Science*, vol. 125, pp. 11–17, 2018.

[399] N. Kumar, M. Takács, and Z. Vámossy, "Robot navigation in unknown environment using fuzzy logic", in *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, IEEE, 2017, pp. 000 279–000 284.

[400] C. F. Riman and P. E. Abi-Char, "Fuzzy logic control for mobile robot navigation in automated storage", *Int J Mech Eng Robot Res*, vol. 12, pp. 313–323, 2023.

[401] O. A. Khan, F. Kunwar, U. S. Khan, and H. Jabbar, "Z-number-based fuzzy logic approach for mobile robot navigation", *IEEE Access*, vol. 11, pp. 131 979–131 997, 2023.

[402] S. Agrawal, B. Patle, and S. Sanap, "Navigation control of unmanned aerial vehicles in dynamic collaborative indoor environment using probability fuzzy logic approach", *Cognitive Robotics*, 2025.

[403] D. K. Mishra, A. Thomas, J. Kuruvilla, P. Kalyanasundaram, K. R. Prasad, and A. Haldorai, "Design of mobile robot navigation controller using neuro-fuzzy logic system", *Computers and Electrical Engineering*, vol. 101, p. 108 044, 2022.

[404] S. Ayub, N. Singh, M. Z. Hussain, M. Ashraf, D. K. Singh, and A. Haldorai, "Hybrid approach to implement multi-robotic navigation system using neural network, fuzzy logic, and bio-inspired optimization methodologies", *Computational Intelligence*, vol. 39, no. 4, pp. 592–606, 2023.

[405] B. Hadikhani and M. H. Asemani, "Mobile robot navigation using twin delayed deep deterministic policy gradient and fuzzy logic", in *2024 12th RSI International Conference on Robotics and Mechatronics (ICRoM)*, IEEE, 2024, pp. 351–355.

[406] D. Babunski, J. Berisha, E. Zaev, and X. Bajrami, "Application of fuzzy logic and pid controller for mobile robot navigation", in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2020, pp. 1–4.

[407] J. van der Waa, E. Nieuwburg, A. Cremers, and M. Neerincx, "Evaluating XAI: A comparison of rule-based and example-based explanations", *Artificial intelligence*, vol. 291, p. 103 404, 2021.

[408] K. Shihabudheen and G. N. Pillai, "Recent advances in neuro-fuzzy system: A survey", *Knowledge-Based Systems*, vol. 152, pp. 136–162, 2018.

APPENDICES

A. THEORETICAL BASICS OF MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) is an optimal control technique in which the control actions are computed to minimize a cost function over a receding horizon for a dynamical system subject to constraints.

At each time step, the MPC controller receives or estimates the current state of the system. It then computes the sequence of control actions that minimizes the cost over the horizon by solving a constrained optimization problem that uses an internal system model. The controller then applies to the system only the first computed control action. In the following time step the process is repeated.

We consider a system, in general nonlinear:

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

$$\boldsymbol{y}(k) = h(\boldsymbol{x}(k)) \tag{2}$$

MPC minimize the following cost function J_N subject to constraints:

$$\min_{\tilde{\boldsymbol{u}}(k)} J_N(\tilde{\boldsymbol{x}}(k), \tilde{\boldsymbol{u}}(k)) \tag{3}$$

$$\begin{aligned}
\text{s.t.} \quad & \boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)) && (4)\\
& \boldsymbol{y}(k) = h(\boldsymbol{x}(k)) && (5)\\
& \tilde{\boldsymbol{u}}(k) \in \mathbb{U} && (6)\\
& \tilde{\boldsymbol{x}}(k) \in \mathbb{X} && (7)\\
& \boldsymbol{x}(N) \in \mathbb{X}_f && (8)
\end{aligned}$$

where N is the prediction horizon, the symbol $\tilde{\cdot}$ is used with a variable to show the sequence of that variable within the prediction horizon, so that $\tilde{\boldsymbol{x}}$ is the state sequence with states for k from 1 to N, $\tilde{\boldsymbol{u}}$ is the control input sequence with control inputs for k from $0, 1$ to $N-1$, and $\mathbb{U}$, $\mathbb{X}$ and $\mathbb{X}_f$ are the constraint set for the control input sequence, state sequence and terminal state, respectively.

After that, the first element of $\tilde{\boldsymbol{u}}(k)$ is the control action injected to the system, and then the optimal control problem is repeated.

B. THEORETICAL BASICS OF ROBUST TUBE-BASED MODEL PREDICTIVE CONTROL

In Robust Tube-based Model Predictive Control (robust TMPC), an optimal control problem is solved for systems under constraints and, differently from conventional MPC, with

uncertainties. The controlled system can be both linear and nonlinear, therefore here we consider the more general nonlinear case. The uncertain system to be controlled is described by a nonlinear difference equation with an additive bounded disturbance:

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)) + \boldsymbol{w}(k) \tag{9}$$

where $\boldsymbol{x}(k) \in \mathbb{X}$ and $\boldsymbol{u}(k) \in \mathbb{U}$ are the state and control input, respectively, with $\mathbb{X}$ and $\mathbb{U}$ the state and control admissible sets, and $\boldsymbol{w}(k) \in \mathbb{W}$ is the disturbance, with $\mathbb{W}$ specifying bounds on the uncertainty.

The nominal system is described by:

$$\boldsymbol{z}(k+1) = f(\boldsymbol{z}(k), \boldsymbol{v}(k)) \tag{10}$$

where $\boldsymbol{z}(k) \in \mathbb{Z}$ and $\boldsymbol{v}(k) \in \mathbb{V}$ are the nominal state and control input, respectively, with $\mathbb{Z} \subset \mathbb{X}$ and $\mathbb{V} \subset \mathbb{U}$.

In robust TMPC, firstly we determine a nominal controller to generate a central nominal trajectory, and secondly we determine an ancillary controller that keeps the state of the uncertain system close to the nominal trajectory, where due to the disturbance the real trajectory lies in a region called *tube*, that includes the nominal trajectory and satisfies all state and control constraints for every admissible disturbance sequence.

Therefore, the nominal optimal control problem is solved in the absence of disturbances and is the following:

$$J_N^{\text{nom}^*}(\tilde{\boldsymbol{z}}(k)) = \min_{\tilde{\boldsymbol{v}}(k)} J_N^{\text{nom}}(\tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{v}}(k)) \tag{11}$$

$$\tilde{\boldsymbol{v}}^*(k) \in \arg\min_{\tilde{\boldsymbol{v}}(k)} J_N^{\text{nom}}(\tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{v}}(k)) \tag{12}$$

where N is the prediction horizon, the symbol $\tilde{\cdot}$ is used with a variable to show the sequence of that variable within the prediction horizon, so that $\tilde{\boldsymbol{z}}(k)$ is the state sequence with states for k from 1 to N, $\tilde{\boldsymbol{v}}(k)$ is the control input sequence with control inputs for k from $0, 1$ to $N-1$, and J_N^{nom} is the nominal cost function.

After that, the ancillary optimal control problem is solved to minimize the cost of the deviation between the trajectories of the two systems, uncertain and nominal, as follows:

$$J_N^{\text{anc}^*}(\tilde{\boldsymbol{x}}(k), \tilde{\boldsymbol{z}}(k)) = \min_{\tilde{\boldsymbol{u}}(k)} J^{\text{anc}}(\tilde{\boldsymbol{x}}(k) - \tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{v}}(k)) \tag{13}$$

where J_N^{anc} is the ancillary cost function. This optimization problem generates the control input:

$$\tilde{\boldsymbol{u}}^*(k) \in \arg\min_{\tilde{\boldsymbol{u}}(k)} J^{\text{anc}}(\tilde{\boldsymbol{x}}(k) - \tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{v}}(k)) \tag{14}$$

The first element of the ancillary control input $\tilde{\boldsymbol{u}}^*(k)$ will be added to the nominal input $\boldsymbol{v}(k)$ to control system (9).

For further information, see [217].

ACKNOWLEDGEMENTS

During this five-year PhD I have had the possibility to live an extraordinary experience, form as a person, improve my skills and progress my professional career. This has also happened thanks to the people that I have met and that have supported me and contributed to the success of the PhD.

First, I would like to thank my supervisors, Anahita Jamshidnejad and Hans Hellendorn, for their valuable guidance and support. In particular, their feedback and advice have been fundamental during these years. Regarding my supervisory team, I am also thankful to Luis Sentis, who has welcomed and guided me during the research visit at UT Austin as the host supervisor.

After that, I would like to thank all those people who have contributed to the papers that are part of this PhD thesis, and who have written such papers as co-authors. Among them, in particular, the students who I have supervised during the PhD for their MSc thesis, the members of my Delft AI lab involved in the joint projects, and the collaborators of UT Austin for the work during my research visit in USA.

Moreover, I am grateful to all the people that I have been able to meet during the PhD, including the colleagues of my department at the Aerospace Engineering faculty, in addition to students, PhD candidates, professors and staff within the TU Delft; the members of UT Austin and the people I have met in Texas; in general, all the friends that I have known in Delft and in the Netherlands; and my family that has supported me from Italy.

CURRICULUM VITÆ

Mirko BAGLIONI

1993 Born in Arezzo, Italy.

EDUCATION

2012–2015 **BSc. Electronics Engineering**
University of Pisa, Italy

2015–2018 **MSc. Automation Engineering**
University of Pisa, Italy

2020–2025 **PhD. Robotics Engineering**
Delft University of Technology, The Netherlands
Thesis: Integrated Model Predictive and Human-Inspired Control for Search-and-Rescue Robotics: Perception, Planning, and Mapping
Promotor: Prof. dr. ir. J. Hellendoorn
Copromotor: Dr. A. Jamshidnejad

EXPERIENCE

2018 **Traineeship for MSc Thesis**
Örebro University, Sweden

2019–2020 **Embedded Software Developer**
Isac, Italy

2024 **Research Visit**
University of Texas at Austin, USA

LIST OF PUBLICATIONS

5. C. Maxwell, **M. Baglioni**, and A. Jamshidnejad, *Model predictive fuzzy control: A hierarchical multi-agent control architecture for outdoor search-and-rescue robots*, Engineering Applications of Artificial Intelligence, (2025), *under review*.

4. **M. Baglioni**, A. Patil, L. Sentis, and A. Jamshidnejad, *Achieving multi-UAV best viewpoint coordination in obstructed environments*, Robotics and Autonomous Systems, (2025), *under review*.

3. K. Rado, **M. Baglioni**, and A. Jamshidnejad, *Enabling robots to autonomously search dynamic cluttered post-disaster environments*, Scientific Reports, vol. 15, no. 1, p. 34778, (2025).

2. B. Esteves Henriques, **M. Baglioni**, and A. Jamshidnejad, *Camera-based mapping in search-and-rescue via flying and ground robot teams*, Machine Vision and Applications, vol. 35, no. 5, p. 117, (2024).

1. **M. Baglioni** and A. Jamshidnejad, *A novel MPC formulation for dynamic target tracking with increased area coverage for search-and-rescue robots*, Journal of Intelligent & Robotic Systems, vol. 110, no. 4, p. 140, (2024).

Propositions

accompanying the dissertation

INTEGRATED MODEL PREDICTIVE AND HUMAN-INSPIRED CONTROL FOR SEARCH-AND-RESCUE ROBOTICS

PERCEPTION, PLANNING, AND MAPPING

by

Mirko BAGLIONI

1. Model predictive controllers should rely on either simplified models or substantial computational resources to be deployed in real life (This thesis, Chapters 2, 3, 5 and 6).
2. Robust control is essential for handling real-world uncertainties, but it can solve only specific problems because robustness can only be guaranteed within certain bounds (This thesis, Chapters 2 and 3).
3. Heterogeneity in teams of robots is a benefit for achieving complex tasks, just as diversity enhances collaboration in human teams (This thesis, Chapter 4).
4. Combining both intelligent and classical control approaches will improve such control approaches by exploiting the benefits of both (This thesis, Chapter 5).
5. In the future, robots will substitute humans in many dangerous jobs, therefore safety of workers such as search-and-rescue crews will be improved.
6. Reproducing the same results from research papers that involve complex algorithms with several parameters is unlikely without access to source code.
7. Supervising MSc students by a PhD candidate during their PhD journey should be recommended since it allows the development of management, leadership and mentoring skills.
8. Given the proficiency of Dutch people in English, learning Dutch by foreigners is important for other reasons than communication, similarly to learning an ancient language such as Latin.
9. Regarding sustainability, big organizations and companies can contribute with a more significant overall impact compared to the contribution of a single individual.
10. We should accept and appreciate the good results that we achieve in a limited time, even if they are not excellent and it would require unlimited time to achieve them.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor Prof.dr.ir. J. Hellendoorn and copromotor Dr. A. Jamshidnejad.