Towards Robotic Convoying in Unstructured Environments

Charles Noren

CMU-RI-TR-25-53

August 2025

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee

Sebastian Scherer	Carnegie Mellon University	Chair
Matthew Travers	Carnegie Mellon University	Chair
John Dolan	Carnegie Mellon University	
Tuomas Sandholm	Carnegie Mellon University	
Ali-akbar Agha-mohammadi	Field AI	

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

To my mom, dad, and brother.

Abstract

Multi-agent robotic teaming is the only realistic solution to many large-scale autonomous operations. Conventionally, operations are modeled as a set of tasks that are largely decoupled from each other and the environment at execution time. However, this operational model fails when the successful execution of a task requires multiple agents to synchronize their actions and adapt those actions to new operating information. This thesis addresses this limitation by explicitly encoding these required inter-agent operational couplings as execution-time synchronization constraints in a hierarchical task allocation and task execution framework. We apply our multi-agent coordination framework to robotic convoy operations to demonstrate its capability to realize solutions to operations with a high degree of synchronized multi-agent collaboration.

We then discuss how environmental interaction fits into that framework, bypassing common modeling assumptions which decouple environmental information from the task model. These assumptions cause agents to passively react to or avoid interactions with the environment, often leading to conservative agent behaviors and a false understanding of action or task feasibility. By relaxing traditional notions of obstacle avoidance in interleaved motion planning strategies, we enable an expanded feasibility of tasks that would otherwise be impossible using traditional passive avoidance behaviors. We then apply the benefits of this approach to robotic convoy operations in unstructured environments, demonstrating how an interaction-aware framework gives rise to new opportunities for synchronized multi-agent collaborations.

Finally, erroneous environmental information can have catastrophic effects on task allocation and multi-agent coordination. To mitigate these effects, we imbue our multi-agent team with the capability to communicate and adapt to newly discovered environmental information. To ensure this communication capability even in communication-deprived unstructured environments, we design a wireless ad hoc network construction technique that maintains an observed minimum signal strength between agents. With this assured communication, we can then address dynamic task allocation problems arising from inaccurate environmental data by utilizing the agents themselves as mobile environmental sensors. We then demonstrate robotic convoy operations in an unstructured environment where both agent and team reallocation and rerouting are required in response to *a priori* unknown information about the environment. By addressing the confluence of coupling effects, our framework effectively addresses a class of synchronized task allocation and execution problems and extends the capabilities of multi-agent robotic systems to new operational paradigms and requirements.

Acknowledgments

It should come as no surprise that the content of this thesis could only be created through the continued dedication of many colleagues, friends, and family members. Thank you all for your support. As this list will certainly not be exhaustive in its inclusions —there are simply too many names to recount— if you do not find your name, please accept my apologies. Many hands have guided my academic journey, each leaving their own unique mark. I assure you that I take something away from each and every interaction that I have had the opportunity to hold during my studies.

First, to my advisors. Dr. Sebastian Scherer and Dr. Matthew Travers have provided integral support in my academic journey. I cannot thank you enough for believing in me as I completed this work. You have each created an environment that has supported my research and challenged me in both the generation and defense of that research. From in-depth discussions about robotics to casual chats during field tests, the opportunity to learn from each of you has been one of the greatest honors I could ask for in my life. In addition, I would like to extend these thanks to each of my committee members: Dr. John Dolan, Dr. Tuomas Sandholm, and Dr. Ali-Akbar Agha-Mohammadi. From your willingness to serve on my committee and to ideate solutions with me, it is your continued pursuit of excellence that has driven me in the preparation of this thesis. I can only hope that each of you has felt the same excitement working with me as I have felt working with each of you.

Throughout my time at Carnegie Mellon University, I have also had the great privilege of collaborating with Dr. Bhaskar Vundurthy. His relentless pursuit of scientific studies and infectious attitude has given light to my life in even the bleakest of days. Dr. Vundurthy is a mentor in every sense of the word, and this journey would not have been completed without him and his family. From our time working together, I have no doubt that the students who will one day call Dr. Vundurthy their advisor will be some of the best trained in the world, solving novel problems under the guidance of his distinct style. I cannot wait to meet them and see your continued body of work grow. From the bottom of my heart, thank you for everything.

Thank you to the SMART Scholarship Program and the personnel of the Ground Vehicle Systems Center (GVSC). Dr. Oleg Sapunkov has been a constant joy to work with, and his unmatched knowledge of vehicle systems inspires me every day. There is no person more responsible for the success of SMART Scholars at GVSC than Linda Durant, who has supported me no matter the challenge. Thank you to Dr. Jonathon Smereka, who always pushes me to see a wider context for my work and provides an unparalleled insight into robotics research. Additionally, I would like to thank Shay Dooley, Geneva Garza, and Jeffery

Ratowski for all of their help over the last two years.

To the members of the AirLab and Biorobotics Lab, thank you for all the unique collaborations and research discussions. The culture of each lab provides for an array of visions for the future of robotics that could not be found anywhere else outside of these groups. First, to Dr. Howie Choset, I cannot thank you enough for the engaging conversations and for the work on the VRPMS-CC. Your ability to uncover the truth about any robotics problem always inspires me, and I am honored to have had the opportunity to work with you. I have had the great pleasure of realizing an extensive robotic system with the Biorobotics MMPUG Team. Namya Bagree, Ralph Boirum, Sahil Chaudhary, Ryan Darnley, James Maier, Darwin Mick, Dr. Zongyuan Shen, Burhanuddin Shirose, Joshua Spisak, and Prasanna Sriganesh are some of the best robotics research engineers and robotics research scientists that I have ever worked with. Each of you has taught me so much during my time at Carnegie Mellon University, and I cannot thank you enough for the opportunity to learn from each of you. From this list, I would like to highlight Prasanna Sriganesh, who gave me the great honor of mentoring him during his thesis. I have no doubt that you will continue to have a large impact on the world of legged robotic systems, no matter where you go. The AirLab has so many talented scholars that it is difficult to name them all in a single paragraph. Undoubtedly, Dr. Muqing Cao, Mateo Guaman Castro, Ian Higgins, Dr. Cherie Ho, Dr. Yaoyu Hu, Andrew Jong, Jay Karhade, Nikhil Keetha, John Keller, Seungchan Kim, Yifei Liu, Rebecca Martin, Dr. Brady Moon, Micah Nye, Dr. Jay Patrikar, Yuheng Qiu, Andrew Saba, Matthew Sivaprakasam, Nayana Suvarna, Samuel Triest, Dr. Viktor Walter, Dr. Wenshan Wang, Steve Willits, and Shibo Zhao have all influenced my research career in one way or another. I would like to specifically highlight Nayana Suvarna, as she gifted me with the opportunity to mentor her during her thesis at Carnegie Mellon. You did great things while you were here at Carnegie Mellon University, and I know you will continue to do great things in industry and beyond.

Thank you to the CMU-RIT-NREC RACER "DEAD-FAST" Team for all the hard work and once-in-a-lifetime research experiences. I will forever be grateful for having the opportunity to work with Norman (Norm) Papernick, whose passion and skill for automation, quick turn-of-phrase, and banter made field operations a pleasure. I will forever remember our time in the Data Hut. To Dr. Jose Gonzalez-Mora, whom I consider a mentor in every sense of the word, thank you for letting me work with you on Chimera. Like Norm, you have shaped my research interests far beyond the RACER program, and I hope to live up to your standard of excellence every time I write a line of code. Thank you to Dr. Herman, who showed me the meaning of leadership and who always found a way to fix any problem that I could find. Thank you to Eric Damm, Dr. Tom Howard, Ashwin Menon, and Joshua Rosser for being our research partners for this adventure. I am honored to have worked alongside each of you, and I

will never forget your willingness to jump into difficult problems alongside me. I also wish to thank Bob Bittner, Edsel Burkholder, Eric Foote, Eric Haywiser, and Israel Ilufoye for their invaluable help, mentorship, and friendship during the RACER program.

Thank you to Dr. Changliu Liu for your patience and mentorship on the nature of conducting research. I have carried every lesson I learned from you throughout this academic journey. I am still honored to have had the opportunity to study under your guidance during my first years at Carnegie Mellon University. I also wish to thank Dr. Tianhao Wei and Dr. Weiye Zhao of the Intelligent Control Laboratory for their camaraderie and brotherhood during our first years.

There are a number of people at the Robotics Institute who have made my life much easier and more fulfilling due to all their hard work and effort. Dr. David Wettergreen and Suzanne Muth always had the answer to any question I could come up with, and their efforts to build and maintain a sustainable Robotics PhD program at Carnegie Mellon University are greatly appreciated. Jean Harpley has also made a profound impact on my time at Carnegie Mellon University. Her tenacity and dedication to improving student life at Carnegie Mellon University is unbounded, and her ability to make my day better with just a "Hello!" should speak volumes about her character and impact. I will miss you so much, Jean! In addition, Victor Valle, Janice Phillips, Dayna Larson, and Peggy Martin have all made my life so much easier during my time at Carnegie Mellon University. Thank you all for your help!

To the Carnegie Mellon University Ballroom Dance Competition Team, thank you for the many hours of joy! It has been the privilege of a lifetime to dance with you all, and I hope to see each of you again on the dance floor. In particular, I would like to thank Rebecca Li, Jolie Ma, and Yuning Zheng for being my dance partners and for creating so many memories that I will carry with me for the rest of my life. Thank you to Dmitry Demidov and Christine Zona for your teaching, knowledge, and passion for dancing. I came to Carnegie Mellon University knowing nothing, and you two have made me the dancer I am today. Finally, thank you to Bhuvan Agrawal, James Apfel, Kyr Brenneman, Aditya Chanana, Hannah Clark, Yeye Deng, Rui Fang, Aaron Fairchild, David Jensen, Jamie Kojiro, Austin Lin, Sasha Shefter, Brandon Smithson, Vanessa Wang, Brenna Wrubel, Xinwen Xu, Abby Quigley, Stephanie and Stephen Zieger, and Tommy Zinkovsky for your mentorship and friendship.

Thank you to Bassam Bikdash for his unending support and unwavering friendship. I cannot thank you enough for everything. Thank you to Chiheb Boussema for his quick wit, sharp mind, and great ideas. Without you, I would not have made it past my first year at Carnegie Mellon University. Thank you to Dr. Jay Evans for his kindness, thoughtfulness, and mentorship on all matters of life.

Ever since we met at Texas A&M University, you have brought a consistent calming force into my life. Thank you to Dr. Cherie Ho for her mentorship and friendship in all things. Your ever-present kind nature and willingness to share the things you enjoy with me have made me feel at home here in Pittsburgh. Similarly, thank you to Dr. Mononito Goswami for his camaraderie and unyielding patience. There is never a dull moment when you are around, and I can only hope one day to capture a fraction of your infectious energy that I am lucky enough to enjoy. Thank you to Dominic Guri for the most engaging conversations I have had at Carnegie Mellon University. I hope that they will continue for many years, even when we are no longer together on campus. Thank you to Dr. Alex LaGrassa, who always provides an insightful perspective and rational voice to a chaotic world. There are very few who can express truths as well as you, and I will always be amazed by your ability to say what should be said. Thank you to Dr. Yeeho Song, who embodies the meaning of bravery, dedication, scholarship, and brotherhood. You have long been my squad member, and I know that you always have my back when I need it the most. Thank you to Sarah Costrell for being an amazing office mate who is always willing to listen to my difficulties and share the joys and burdens of PhD life. You, Thisbe, Hilbert, and Dr. Avik De have been great friends to me, and I will find it a great personal loss when we are no longer officemates. I cannot wait to see what you do in the future, and I wish you the best of luck in your defense! This also applies to Ravi Pandya, who is always willing to answer any question that I can think of. Thank you to Mohamad Qadri for sharing his intellect, time, and perspective with me. I always felt that we could never have a long enough conversation because there were always too many interesting things to talk about. Finally, thank you to Dr. Kevin Zhang, who was always able to make me laugh when I needed it most.

To all my friends in the FIRST Robotics Competition (FRC) community, thank you all for the inspiration that started this path in high school. I owe great thanks to Dr. George Kantor and Elizabeth Kysel for the opportunity to mentor FRC 3504, "Girls of Steel," during my time at Carnegie Mellon University. Furthermore, working with Thomas Pope, Shannon Werntz, Sarah Withee, Joe Jackson, and Linda Hartman to build an impactful Chairman's Program was a great honor. A special shoutout to Tim Angert, who is one of the most insightful and dedicated FRC (and RI) mentors I have ever met. You have always been an inspiration to me, and I cannot wait to see your continued impact on FIRST in PA. Then, to all my Texas FIRST Robotics Volunteers, I miss each of you and I cannot thank you enough for all your friendship and support. In particular, Aaron Graeve and Kolton Coats have been some of the closest friends I have made in my life. Finally, I must thank Scott Rippetoe, Matt Davies, Sherry Coats, and Alan Coats for making FRC 1477 "Texas Torque" a transformative experience in my life.

To Dr. Gregory Chamitoff and Dr. John Valasek, thank you for your mentorship at Texas A&M University. As an undergraduate student, I had a great first exposure to research under your guidance. You both showed me the challenge and excitement of research and helped guide me to this chapter in my life. As he has not received a mention yet, Bill Caraway was one of the closest friends I had at Texas A&M University, and I am still amazed by his work ethic and good nature in the face of adversity.

Before I came to Carnegie Mellon University, I had a once-in-a-lifetime opportunity to work at the NASA Langley Research Center (LaRC). I cannot thank Dr. Danette Allen enough for always believing in me and allowing me to contribute to cutting-edge research at the Autonomy Incubator. Your support during and after my undergraduate prepared me for the rigors of Carnegie Mellon University, and getting to see your impact on the world of robotics led me to conduct my own PhD journey. At LaRC, I also had the pleasure to meet Dr. Javier Puig-Navarro, Dr. Brian Duvall, and Dr. Meghan Saephan, who further inspired me to continue to pursue a career in research.

Thank you to the Kennedy family and the Yu family, who have supported me every step of the way since I was a young boy. I could not have made this journey without your help and guidance. There is nothing more I can say than that I love you all and that I cannot imagine my life without you all.

Finally, this acknowledgment would not be complete without thanking my family. To my grandmother, the Rosenbalms, the Pates, and the Stanleys, you are the joy and light in my life. I have missed each one of you every day I have been here in PA, and I cannot wait to return to you soon. To Drew and Ali, I could not ask for a better brother and sister-in-law. Each of you has provided so much happiness in my life, and I have never felt alone here in PA, even though we have been miles apart. Finally, to my mother and father, thank you for everything. From the calls of support, to my upbringing, to having a home to rest at after every semester, you two gave me the strength and resources I needed to finish this milestone. This thesis is for you.

Funding

This work was in part supported by OUSD/R&E (The Under Secretary of Defense-Research and Engineering), National Defense Education Program (NDEP) / BA-1, Basic Research. The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Chapter 8 - Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

Part of the research presented within the confines of this document (Chapter 8) was sponsored by the Defense Advanced Research Projects Agency (DARPA) (HR001121S0004). The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Figure 1.1: A convoy of robotic agents traveling between points of interest.

One fundamental aspect of multi-agent robotic systems is that any given agent does not exist solely in a state of isolation. The behavior or action of any given agent in a multi-agent system is at least influenced by, or coupled to, the behaviors of the other agents. This coupling forms the basis of multi-agent coordination, where multiple agents work together to solve a task or set of tasks. When a task requires the synchronized behavior of multiple agents, in order to perform efficient and effective multi-agent coordination, it is necessary to address this coupling during both the task allocation and task execution phases. Further complicating this problem are many practical constraints that couple the agent's actions with both the physical and information environments within which the agents operate. The framework developed in this thesis presents a unified approach that addresses multi-agent task coupling arising from synchronized tasks, information coupling resulting from environmental dynamism, and environmental coupling resulting from interactions with the environment. We provide a holistic approach to multi-agent coordination with a focus on realizing off-road multi-agent convoying operations.

1.1 Motivating Example - Convoy Operations

(a) (b)

Figure 1.2: Examples of two different applications in the convoying domain. Multi-vehicle convoying has historically taken place across both land (Figure 1.2a, "KSC Battalion Conducts Convoy Movement Training [Image 10 of 19]" by Sang Yun Kim is in the Public Domain [1]) and maritime (Figure 1.2b, "USCGC Bear (WMEC 901) Participates in Operation Nanook [Image 5 of 14]" by Matthew Abban is in the Public Domain [2]) environments. Multiple vehicles may participate in a convoy to either satisfy large-scale logistical mission requirements or for mutual protection. The appearance of U.S. Department of Defense (DoD) visual information (Figure 1.2a, Figure 1.2b) does not imply or constitute DoD endorsement.

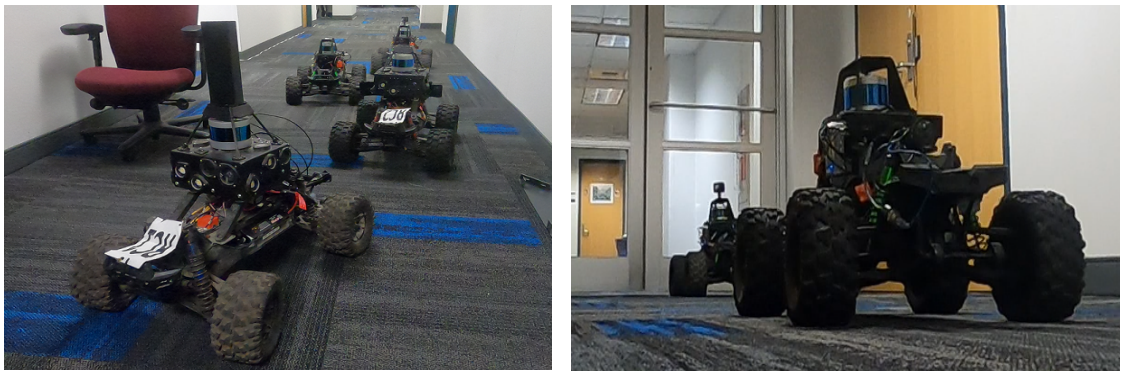
During operations in Iraq, ground transports reportedly moved ninety-eight percent of the military's equipment and supplies in on-road and off-road environments [5]. This reliance on ground transports, often organized into dedicated supply "convoys," was not unique to operations in Iraq and has been reflected in multiple conflicts since the Second World War [6, 7]. Due to the scale of convoy logistics challenges and the associated operational dangers to ground transport crews, the scientific community has conducted research on replacing crewed vehicles with automated platforms, with a specific focus on convoy formations [5, 8].

The majority of these previous works focus on the principal mechanism for executing convoy tasks: formation-keeping. A multi-agent formation-keeping behavior couples the spatial and temporal motions of individual robotic platforms to ensure that each agent maintains its relative position with respect to the other agents. Although formation-keeping is indeed an important first step towards realizing automated convoying systems, the nature of the work itself suggests further study on how higher-level task allocation reasoning frameworks embed low-level multi-agent coupled task execution behaviors. In the context of robotic convoy, existing literature often considers the question of "*how to convoy*," but then fails to consider profound questions regarding "*when and where agents should form*

a convoy." Thus, allocations of agents to highly-coupled tasks (e.g., Which agents should be assigned to which convoys?) must observe the initial conditions required to address inter-agent coupling in the task execution phase.

For many task allocation algorithms, allocation mechanisms assign agents to tasks under the assumption of information certainty. However, in many real-world allocation problems, this certainty does not hold. Instead, there often exists an "evolutionary" aspect to information availability [9], leading to information dynamism in the allocation problem. This dynamism can then lead to an underlying infeasibility in the allocation phase or to inefficiencies in the agent allocations. Unfortunately, as automated convoying operations entail the synchronized behaviors of multiple robotic agents, the added coupling between agents causes these inefficiencies to propagate to multiple assets. This operational risk suggests the need for adaptive or dynamic responses to operational (e.g., environmental) data. However, responding to operational data often requires the exchange of information between agents. Given the commonality of information dynamism, agents are often equipped with onboard communication systems to support the exchange of information between the agents. Under certain conditions (e.g., small inter-agent distances), this communication system then enables the transfer of task allocation and task execution data between agents. Thus, to encourage adaptive multi-agent behaviors in response to information (e.g., environmental) dynamism, the design of the robotic system must support information transfer to capture the evolving environmental conditions during the system's operation.

Finally, while many existing works have developed automated convoying systems for on-road environments, these works do not address the challenges associated with off-road or "unstructured" environments. One defining aspect of off-road operations is that there is no guarantee of a single discernible paved track to the destination location [10]. In on-road driving, this single discernible track provides environmental structure that the agents may exploit during task allocation and execution. When this environmental structure is nonexistent, automated systems need to consider how to incorporate interactions with unique off-road environmental features (e.g., vegetation, gates, doors) not found in on-road environments. Figure 1.3 illustrates two contrasting examples of environmental objects. In Figure 1.3a, the agents observe a classical notion of obstacle avoidance to avoid interaction with the environmental clutter. While the object interferes with the agents' motion, the agents do not have to interact with the object to travel around it. Figure 1.3b stands in contrast to the scenario presented in Figure 1.3a, as the environmental objects prevent the agents from progressing to their next task. The only way to progress would be to interact with the door (e.g., opening it). Thus, to extend both task allocation and task execution techniques to off-road environments, the effects of the interactions must be



(a) (b)

Figure 1.3: A convoy of small wheeled platforms encounters different types of objects during system operation. Figure 1.3a illustrates environmental clutter, which prompts the convoy team to adjust its formation structure to pass. Figure 1.3b demonstrates an environmental obstacle (a closed door), which can impede the motion of the agents. The necessity for interaction strategies in the task execution phase is evident in both cases; however, accounting for environmental interactions must also be incorporated into the task allocation phase for allocation strategies to represent the task execution accurately.

captured by the underlying model. As such, convoys that operate in off-road environments also exhibit a non-trivial coupling with the physical environment and must be augmented with additional capabilities to handle the missing environmental structure provided by on-road environments.

1.2 Generalized Multi-agent Teaming Research Questions

Given our discussion in the previous section, it is clear that multi-agent systems that perform automated convoying operations demonstrate at least three types of coupling. These are: 1) task allocation and task execution coupling, 2) information coupling, and 3) environmental coupling. In order to support a generalized framework for multi-agent teaming in unstructured environments, we are thus concerned with the following three research questions (RQ):

RQ 1: (Task Allocation and Task Execution Coupling): What are the properties of and solution methods to multi-agent system operations that require the synchronized actions of individual agents?

RQ 2: (Information Coupling): How does dynamism affect multi-agent system operations, and how can the design of automated network construction algorithms

mitigate the influence of this dynamism?

RQ 3: (Environmental Coupling): How can relaxations of common environmental assumptions (e.g., the obstacle avoidance paradigm) influence task allocation and task execution in multi-agent systems?

1.3 Thesis Statement

The purpose of this thesis is to develop methodologies that may improve the command and control capabilities for coalitions of automated platforms that must operate in convoy formations. We argue that addressing the different forms of coupling outlined above is the critical step towards realizing robotic convoying in unstructured environments. With this perspective in mind, we assert the following thesis statement.

Thesis Statement:

The performance of multi-agent systems in unstructured environments with tasks that have coupled execution requirements can be improved by:

1. forming representative task abstractions,
2. responding to evolving mission information, and
3. reasoning about environmental interactability.

Pertinent definitions to the thesis statement, above, are as follows:

- "*performance . . . improved*" is considered within the context of traditional routing problems (e.g., decreasing the total distance traveled by all agents or the total time of the operation).
- "*unstructured environments*" are defined as operating environments where no discernible or object-free path-to-goal exists. Examples of such environments include off-trail traversals in a natural landscape (e.g., with vegetative objects) or in-building traversals (e.g., with furniture or door objects).
- "*coupled execution requirements*" is defined as a task that can only be satisfied if multiple agents spatially and temporally couple their states or actions together during the execution of the task.
- "*representative task abstractions*" are functional representations of tasks that accurately capture the constraints and costs associated with task completion.
- "*evolving mission information*" is described as information pertinent to the conducted operation that may be discovered during the operation.

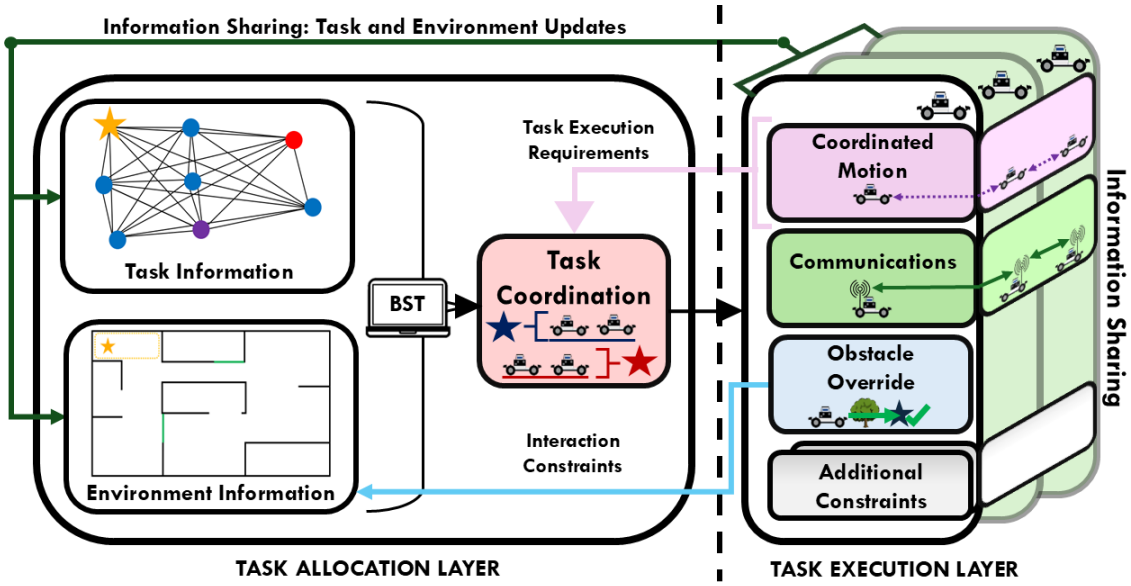


Figure 1.4: The interaction-aware synchronized task allocation and task execution framework developed to support the above thesis statement

- "*environmental interactability*" is defined as the relaxation of obstacle avoidance requirements such that an agent's state may be co-located with a subset of environmental objects under certain conditions.

The principal claim of this work is that for operations in unstructured environments that require multiple agents to synchronize their actions, robotic agents must have effective task abstractions, communication strategies that support the transfer of operation information, and interaction-aware agent motion models. To support this hypothesis, we present a multi-agent mission planning framework tailored specifically to address the challenges associated with robotic convoy operations. Figure 1.4 depicts our two-layer task allocation and task execution framework.

The core component of the framework is a centralized task allocation layer that casts convoy operations as a variant of the vehicle routing problem with multiple synchronizations (VRPMS). We introduce multi-agent coordinating constraints that require a specific number of platforms to travel as a convoy to accomplish the task at each goal (waypoint). Our particular formulation of the vehicle routing problem is named the *Vehicle Routing Problem with Multiple Synchronizations - Convoy Constraints* (VRPMS-CC). This layer can also reroute or reallocate agents to different convoying tasks in light of *a priori* unknown environmental information.

The next component of the framework is a task execution layer. This layer consists of a

series of individual behaviors implemented on each robotic agent. Each behavior enables the agents to perform tasks central to automated convoying. In particular, we introduce a coordinated motion behavior, a communications network construction behavior ("peel-off"), and an obstacle override behavior. To address task execution coupling, the coordinated motion behavior describes how agents must respond to a convoy assignment from the task allocation layer. The communication network construction behavior enables agents to monitor the communication environment and deploy communication-range-extending nodes to maintain a minimum radio signal strength during system operations. We then address information coupling by combining this capability with the agent re-allocation capability in the previous layer. Finally, from the factors discussed above, the unstructured environment may impact the agents' routes in a manner that becomes apparent only during the execution of the route. As this may lead to a routing problem where environmental interactions influence the feasibility of executing the route (e.g., a "*traversable*" interaction that allows the vehicle to travel along the route), we equip each robotic agent with an interaction-aware task execution policy. This policy enables the agent to interact with a subset of environmental objects, thereby addressing environmental coupling. Combining this interaction-aware policy with our re-allocation capability enables our system to address a wide range of real-world environments with different realistic constraints on environmental interactability.

1.4 Research Question Discussion

In this section, we break down each research question into an associated set of technical challenges. We aim to provide technical contributions to address these challenges in the context of automated convoying in unstructured environments (e.g., off-road terrains).

Research Question 1

What are the properties of and solution methods to multi-agent system operations that require the synchronized actions of individual agents?

Many multi-agent system architecture designs decompose the system operation (mission) into atomic tasks that the individual agents then complete. The decision framework underpinning such an architecture design thus assumes that the tasks it forms exhibit an atomic nature, independent of inter-agent coupling. However, many crucial multi-agent system operations require synchronicity (coupling) to exist between the states and actions of multiple robotic agents. Further complicating synchronized robotic agent operations is

that coupling may exist in both the task allocation (e.g., assigning multiple agents to a set of synchronized tasks) and task execution (e.g., agents must couple their actions together to complete a single task) aspects of the system operation. While monolithic frameworks can ensure the observance of both task allocation coupling and task execution coupling, pragmatic concerns surrounding high-rate coupling can make such frameworks difficult to implement [8, 11]. In the context of automated convoying, the allocation of agents into formations and the subsequent formation-keeping constraint represent task allocation and task execution coupling, respectively. Upon inspecting these two aspects of autonomous convoying, we notice a natural functional abstraction of the convoying task. This functional abstraction represents convoys as their start and goal locations, along with the agents assigned to each convoy. We note that the abstract representation of the convoying task is distinct from the task itself, implying that we can also consider the synchronization of the agents separately during task execution. Our observation then leads to the question of whether we can leverage this natural functional abstraction to form a distinct task allocation and execution framework for multi-agent convoying systems.

Research Question 2

How does dynamism affect multi-agent system operations, and how can the design of automated network construction algorithms mitigate the influence of this dynamism?

Given **Research Question 1**, we note many practical concerns regarding the deployment of such frameworks to real-world environments. In particular, task allocation schemes optimize agent performance based on the available information at the time of the task allocation. This optimization is often dependent on associating accurate costs with individual tasks. Yet, in many robotics applications, predicting this task cost is non-trivial. This non-trivial task cost estimation becomes even more challenging if, at the time of task allocation, an unknown aspect that could significantly affect the task cost estimate is unknown (not in the context of uncertainty, but rather an "unknown" unknown). Such matters are the concern of the field of dynamic routing; however, implementing a dynamic routing algorithm on a robotic agent team presents several impediments. The most significant impediment is that in many robotic operations, inter-agent communication may not be inherently available from the existing communications infrastructure in the environment. In such a scenario, robotic agents can establish an ad hoc network to facilitate inter-agent communication. We seek to answer questions regarding 1) how to facilitate the construction of this ad hoc network, and 2) how task allocation and task execution coupling affect the construction of this ad hoc network.

Research Question 3

How can relaxations of common environmental assumptions (e.g., obstacle avoidance paradigm) influence task allocation and task execution in multi-agent systems?

Traditional robotics paradigms indicate that robotic agents should not interact with objects in the environment. While this paradigm may prove useful in highly-controlled environments, allowing interactions with objects in the environment may alter the feasibility and optimality of an agent's motion through the environment [12, 13, 14, 15]. Yet, accurately realizing and representing interaction-aware behaviors at both a task allocation and task execution level is non-trivial. At the task allocation layer, environmental interactability may lead to more accurate and representative estimates of costs associated with task execution. These changing estimates are associated with the earlier discussed altered task feasibility and optimality. Depending on the sensitivity of the task allocation to certain constraints, modifying these estimates may lead to a larger feasible set of solutions at the task allocation layer. However, interaction-aware behaviors are challenging to realize on robotic systems. While previous methods often rely on perception mechanisms to classify the object space into interactable-object classes, the interaction feasibility may depend on the agent's state at the point of interaction. For automated convoying systems interacting with environmental vegetation, numerous works in the off-road mobility and modeling community confirm this fact [16, 17, 18]. There are clear advantages to creating interaction-aware robotic agent behaviors and incorporating these behaviors into both the task allocation and task execution layers of the framework designed in **Research Question 1**. Creating an interaction-aware control policy for common vegetative objects is invaluable for convoy operations, as such operations often occur in environments where such objects are present. As such, vegetation interaction forms the basis of our investigation in this thesis.

1.5 Thesis Organization

Given a short description of each challenge above, we divide this thesis into three parts. Each part corresponds to a particular research question. We then further divide each part into two individual chapters, tackling specific aspects of each respective research question. The three parts of this thesis are as follows:

1. Part I: The thesis begins with Chapter 2, where we first consider how to perform coordinated motion. The capability to perform this task forms the basis of the task execution layer, where we address the primary expression of coupling for our

task: convoy formation-keeping. In this chapter, we introduce an optimization-based formation controller for a robotic convoying system and then utilize it to abstract the formation control task into a functional robot behavior. We then leverage this functional convoy task abstraction in Chapter 3 to perform the synchronized routing of multiple agents. In Chapter 3, we investigate a variant of the vehicle routing problem with multiple synchronization constraints (VRPMS) to ensure that each agent meets the necessary initial conditions for convoy assignments. We name this variant the Vehicle Routing Problem with Multiple Synchronizations - Convoy Constraints (VRPMS-CC). We investigate multiple solution techniques to this vehicle routing problem variant and demonstrate the effectiveness of a decomposition-based heuristic for warm-starting the routing problem solver. By combining our convoy controller with the VRPMS-CC, we form the basis for the framework illustrated in Figure 1.4.

2. Part II: Our work in Part I demonstrates the need for dynamic routing capabilities during robotic system operation. To enable these dynamic routing capabilities in communications-deprived environments, we introduce an ad hoc wireless network construction technique in Chapter 4. This technique novelly combines a spanning tree variant, which we call the Maximin Communications Spanning Tree (MCST), with a node placement logic developed during the DARPA Subterranean Challenge. We first introduce the MCST in Chapter 4, and conclude the section with several simulation and hardware results. We then apply the MCST to convoy operations in Chapter 5, where we develop additional application-specific behaviors to support convoy operations in communications-deprived environments. We utilize this ad hoc wireless network construction technique to support a demonstration of a dynamic version of the VRPMS-CC (Dynamic VRPMS-CC) in Chapter 3.

3. Part III: Finally, we consider the implications of environmental coupling in Part III. In particular, we consider how the unstructured terrain enables aggressive behaviors that would be unnecessary or impermissible in more structured terrains. We first consider a common unstructured terrain: natural off-road environments, and introduce a custom trajectory optimization controller for vegetation override in Chapter 6. This custom controller utilizes a semi-analytical model to enable wheeled robotic systems to override small obstacles, such as vegetation and posts. We then introduce an interleaved motion planning and control scheme in Chapter 7 that selectively utilizes the controller developed in Chapter 6 to plan paths to a goal. This interleaved motion planning and control scheme combines the advantages of sampling-based planning

with trajectory optimization to account for and balance interactions with objects in the environment, thereby enhancing the overall system performance.

1.6 Summary of Technical Contributions

On the nature of automated convoy organization and execution

Citation:

> N. Bagree, **C. Noren**, D. Singh, M. Travers, and B. Vundurthy, "Distributed Optimal Control Framework for High-Speed Convoys: Theory and Hardware Results," in *2023 IFAC-PapersOnline*, vol. 56, no. 2, pp. 2127-2133, November 2023

Citation:

> **C. Noren**, B. Vundurthy, S. Scherer, H. Choset and M. Travers, "A Synchronized Task Formulation for Robotic Convoy Operations," in *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 6808-6815, July 2025

On the automated construction of wireless ad hoc communications networks

Citation:

> **C. Noren**, B. Vundurthy, N. Bagree, M. Travers, "A Max-min Tree Approach to the Automated Construction of Ad hoc Wireless Networks in Unknown Environments," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, Los Angeles, USA, August 2025

Citation:

> **C. Noren**, S. Chaudhary, B. Shirose, B. Vundurthy, M. Travers, "Communication Network Construction Behaviors for Robotic Convoying," in *Proceedings of the 2025 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, NDIA, Novi, USA, August 2025

On the development of vegetation override capabilities for robotic systems

Citation:

> **C. Noren**, B. Shirose, B. Vundurthy, S. Scherer, and M. Travers. "An Interaction-Aware Two-Level Robotic Planning and Control System for Vegetation Override," in *Proceedings of the 21st ISTVS International and 12th Asia-Pacific Regional Conference of the ISTVS*, Yokohama, Japan, October 2024

Citation:

> **C. Noren**, B. Vundurthy, S. Scherer, and M. Travers. "Interaction-aware Control for Robotic Vegetation Override in Off-road Environments," in *2025 Journal of Terramechanics*, vol. 117, pp. 2127-2133, February 2025

Citation:

> **C. Noren**, B. Vundurthy, S. Scherer, and M. Travers. "Trajectory Optimization for Vegetation Override in Off-road Driving," in *Proceedings of the 16th European-African Regional Conference of the ISTVS*, Lublin, Poland, October 2023

1.7 Additional Peer-Reviewed Works

Citation:

> P. Sriganesh, J. Maier, A. Johnson, B. Shirose, R. Chandrasekar, **C. Noren**, J. Spisak, R. Darnley, B. Vundurthy, and M. Travers, "Modular, Resilient, and Scalable System Design Approaches - Lessons learned in the years after DARPA Subterranean Challenge," in *2024 IEEE International Conference on Robotics and Automation (ICRA) Workshop on Field Robotics*, Yokohama, Japan, May 2024

Citation:

> **C. Noren**, W. Zhao, and C. Liu, "Safe Adaptation with Multiplicative Uncertainties Using Robust Safe Set Algorithm," in *Proceedings of the 2021 Modeling Estimation and Controls Conference (MECC)*, vol 54, no 20, pp. 360-365, December 2021

Part I

Convoy Routing and Control

Chapter 2

A Distributed Optimal Control Framework for High-Speed Convoys

The distributed multi-platform control scheme described in this chapter was presented at the 22nd International Federation of Automatic Control (IFAC) World Congress.

Citation:

> N. Bagree, **C. Noren**, D. Singh, M. Travers, and B. Vundurthy, "Distributed Optimal Control Framework for High-Speed Convoys: Theory and Hardware Results," in *2023 IFAC-PapersOnline*, vol. 56, no. 2, pp. 2127-2133, November 2023

2.1 Introduction

The first step in building a multi-agent system lies in understanding the fundamental tasks it must complete. Undeniably, the multi-agent system performing a convoy operation must have the capability to travel as a convoy. Developing this fundamental multi-agent convoying capability is the focus of this chapter. We develop this capability with the understanding that the agents may be traveling in unstructured environments. Operating in an unstructured environment yields an additional context that affects the design of the underlying convoy formation-keeping algorithm (controller).

In particular, that additional context arises from the empirical observation that coordinated fleets of mobile automated platforms (agents) in unstructured environments benefit from small inter-platform distances, especially when these platforms move at higher speeds. This yields a new challenge in that as speed increases, reducing the space between platforms also reduces the time available to the platforms to respond to sudden motion variations of the surrounding platforms. However, in specific examples, the benefits in performance due to traveling at closer distances can outweigh the potential instability issues. These benefits are well-documented for on-road environments, with the most highly cited example being reduced fuel consumption resulting from a reduction in aerodynamic drag [8, 19]. Small mobile robots can also achieve performance improvements by operating at high speeds in close proximity to other agents in unstructured environments. A practical challenge in search and rescue missions is maintaining consistent communication between the robots, especially in cluttered environments where the loss of direct line of sight can

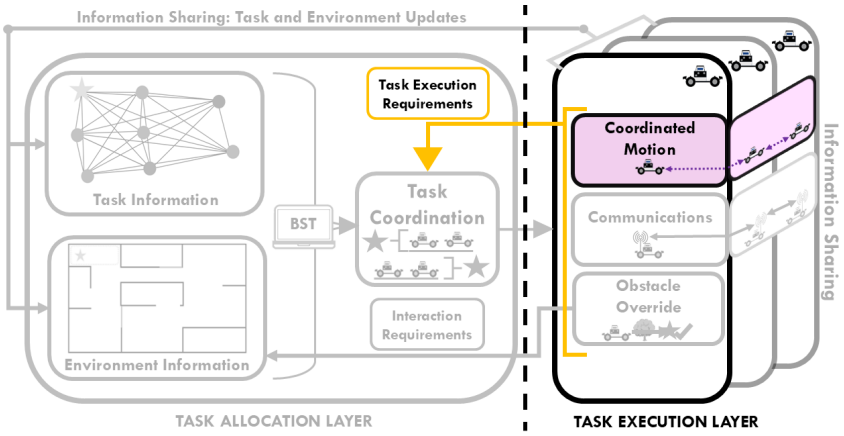


Figure 2.1: Coordinated motion (convoy formation-keeping) forms the basis of the task execution layer of our framework. We focus on the task execution layer in this section and address the task execution requirements in Chapter 3.

hinder inter-robot communications [20].

To achieve this kind of closely coordinated fleet behavior, we introduce a model predictive optimal control framework that directly accounts for the nonlinear dynamics of the agents in the fleet while planning motions for each platform. The platforms can follow each other closely at high speeds by proactively making predictions and reactively adjusting their responses based on state information from the adjacent platforms. This control framework is naturally decentralized and empirically demonstrates lower inter-platform distances at higher speeds compared to existing formation-keeping controllers.

2.2 Literature review of convoying systems

To provide context for our control approach, we review relevant prior work that addresses the convoy formation-keeping problem. We focus our discussion on convoy formation-keeping controllers designed for unstructured and cluttered environments.

One of the earliest works in convoy control mimics a leader-follower behavior where each vehicle estimates and stores the path of its predecessor as a set of points [21]. The follower then estimates the predecessor's path curvature around a selected target and follows the trajectory. Nestlinger et al. [11] extends this work to store position measurements over time and apply a spline-approximation technique to obtain a smooth reference path for the underlying motion controllers. These methods force each agent in the convoy to track the exact positions of its predecessors, a strict formation-keeping requirement that restricts system flexibility in cluttered environments (e.g., around obstacles).

In contrast to these strict formation-keeping requirements, Albrecht et al. [22] provides a framework for switching between exact pose tracking and a flexible path search and tracking mode based on the environment. The added flexible path search mode relaxes the strict formation-keeping requirements, yielding better performance in real-world conditions. However, there is no interaction between the outputs of the two tracking methods, resulting in potentially conflicting tracking requirements when switching between the modes.

Shin et al. [23] incorporates a dedicated obstacle avoidance module in the control framework to tackle the challenges of unstructured environments. Here, the authors approach the convoy problem using a passivity-based model predictive control method that integrates a traversability map into the planner. However, this framework does not incorporate feedback from the following vehicles, which can result in high inter-robot distances on cluttered terrains. The framework also relies on continuous communication between the vehicles and a central node to operate.

Finally, Turri et al. [19] presents an alternative to "follow-the-leader" style frameworks.

The authors in [19] adopt a centralized approach to designing their controller, with a primary focus on straight-line velocity profiles. The goal of the presented approach is to improve fuel efficiency while ensuring desired safe following distances between vehicles. As formation size increases, the computational cost increases quadratically with the number of agents. Furthermore, the approach needs to be solved on a single computer, creating a single point of failure. As in Shin et al. [23], the approach in [19] also requires high-rate continuous communications between the robots and the base node for safe convoying.

2.3 Convoy problem formulation

Consider a convoy consisting of at least three agents, structured in a column formation. In a column formation, the agents position themselves spatially one after the other [7]. We assume a fixed formation structure during system operation. In this formation structure, we order the convoy such that every agent in the "interior" of the column (i.e., not the first or last vehicle in the column) is preceded by a "leading" (lead) agent and is succeeded by a "following" (follow) agent. Instead of requiring the agents to maintain the strict spatial positioning between the lead and follow agents, we relax the problem such that the agents must only maintain the formation ordering while trying to minimize their distance to a fixed point along the desired path. We illustrate this idea in Figure 2.2, where agent i (green) minimizes error e (given a metric, e.g., Euclidean distance) between its current location and a target point of interest defined by d^{ref} along the desired path.

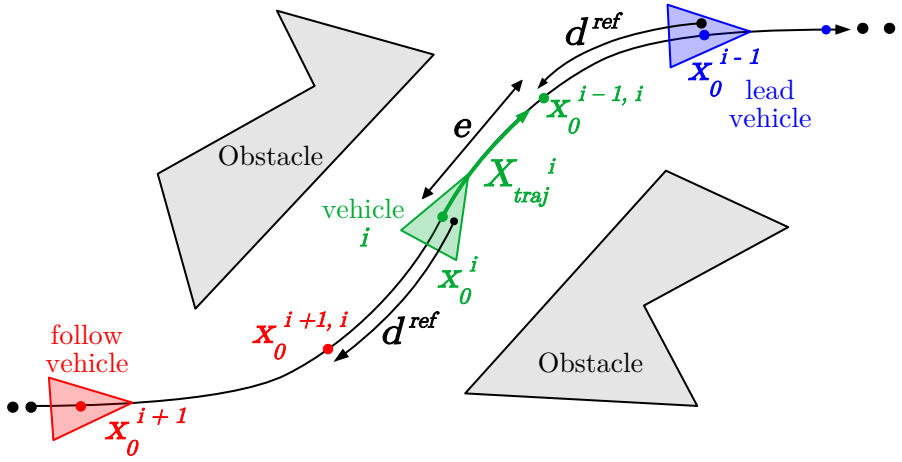


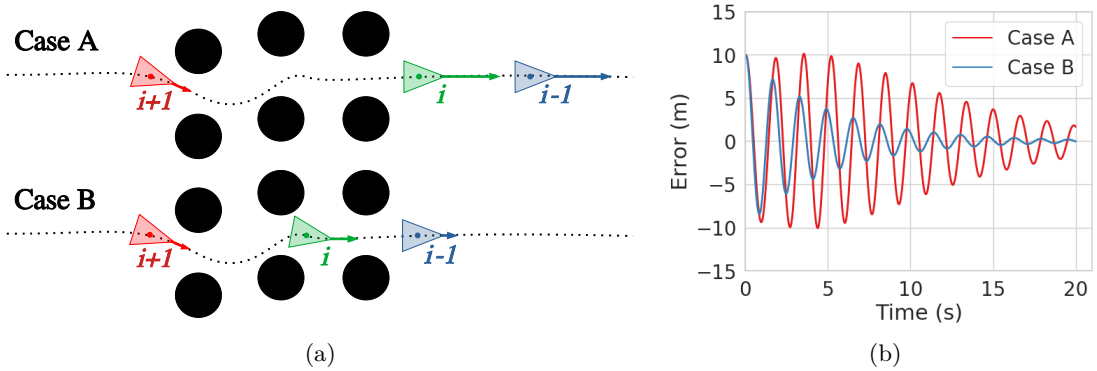
Figure 2.2: Schematic for the convoy control problem from the perspective of an "interior" agent i. Error e is defined as the difference between the agent's desired position (e.g., d^{ref} behind the lead agent as measured using the Euclidean distance) and its current location.

Given a formation of L agents, we first define an agent index associated with that agent's position in the formation. The index is assigned such that it begins with a value of 1 (i.e., the first agent in the convoy) and increases with each subsequent agent. Thus, for agent i which is neither the first nor last agent in the formation, the neighboring preceding (leading) agent and succeeding (following) agent are denoted as $i-1$ and $i+1$, respectively. We can then represent the entire list of agents via an index set $\mathcal{I}_L = \{1, 2, \cdots, L\}$, where $i \in \mathcal{I}_L$ represents agent i through its index value.

Our goal is to design a convoy controller that generates a set of control outputs, $u_0^i \; \forall \; i \in \mathcal{I}_L$, enabling the agents to travel to a waypoint of interest while maintaining the provided convoy formation ordering. Thus, if we consider the convoy motion itself as a "task" in a broader mission that agents must perform, this formation ordering constraint is an example of task execution coupling that is present during system operation. In particular, there are three desired objectives shaping each agent's motion in the convoy. Each agent should:

1. maintain a fixed distance d^{ref} along the desired path from its lead vehicle;
2. travel at a specified desired speed;
3. react to disturbances (e.g., avoiding obstacles) in the environment.

However, we must also ensure that the execution of this task accurately represents the behavior we wish the multi-agent system to exhibit. One challenge with "follow-the-leader" style controller architectures is that the primary dependency (coupling) is often conditioned only on the leading agent. We illustrate a scenario in Figure 2.3 to better understand the



(a) (b)

Figure 2.3: Comparison of convoys when following robots are neglected in the framework (Case A) vs. when they are not (Case B). In Figure 2.3a, an environmental situation where agent $i+1$ is "stuck" is shown. In Figure 2.3b, a fleet response demonstrating the "accordion" problem is shown.

comparison of a "follow-the-leader" (Case A) behavior against our desired behavior (Case B). In both scenarios, the obstacles in the environment force agent $i+1$ to slow down. To demonstrate the two behaviors, we modeled the convoy controller as a spring-mass-damper system defined between agents in the convoy. In Case A, this spring-mass-damper system is defined pairwise between subsequent agents in the convoy (e.g., between agent $i-1$ and agent i). This structure imposes a coupling between agents where each agent is only affected by its lead agent. However, in Case B, both adjacent agents (leading $(i-1)$ and following $(i+1)$) exhibit a fictitious force on agent i.

Let us consider the effects of this difference in convoy modeling. Again, the goal of our convoy controller design is to represent the desired behavior in the task execution. First, of fundamental importance, consider if the obstacle field observed in Figure 2.3a caused agent $i+1$ to become permanently "stuck." Under the "follow-the-leader" paradigm, such a permanently "stuck" agent would "split" the convoy due to the presence of obstacles, undoubtedly violating the desired behavior. However, even if agent $i+1$ is not permanently "stuck," Figure 2.3b demonstrates that the agent impacted by the obstacle disturbance induces an oscillatory effect in the convoy structure. Figure 2.3b further indicates that under this oscillatory effect, Case B stabilizes faster and reduces oscillatory effects between the agents. We hypothesize that this structure will enable the agents to maintain lower inter-robot distances throughout the environment, thereby facilitating improved real-time communication between them.

2.4 Convoy controller design

To achieve the desired behaviors, we pose the convoy problem presented in Section 2.3 as an optimal control problem with a bi-objective cost structure. The terms in the controller objective balance deviance from the ego-agent's (e.g., agent i's) desired trajectory along the path, with a penalty associated with breaking the convoy formation structure. We now describe the proposed controller in detail.

2.4.1 Posing the convoy problem in an optimal control framework

We consider a model-based framework, where each agent's motion is captured by a discrete-time state-transition model $x^i_{k+1} = f(x^i_k, u^i_k, \Delta t)$. Here, $x^i_k \in \mathbb{R}^m$ represents the state of agent i at time step k, $u^i_k \in \mathbb{R}^n$ represents a control input for agent i at time step k, and Δt represents the assumed fixed time discretization of the state transition function $(\Delta t = t_{k+1} - t_k)$. For a finite time horizon consisting of N time steps, we can then define

a state and control trajectory for each agent. For agent i, the agent's state and control trajectories are defined as $X^i = [x_0^i, x_1^i, ..., x_N^i]$ and $U^i = [u_0^i, u_1^i, ..., u_{N-1}^i]$, respectively.

For the i^{th} agent participating in the convoy formation, we pose the discrete-time bi-objective optimal control problem framework as:

$$\min_{X^i,\, U^i} \quad C_{traj}(X^i, U^i) + C_{convoy}(X^i, X^{i-1}, X^{i+1}) \tag{2.1a}$$

$$\text{subject to} \quad x_{k+1}^i = f(x_k^i, u_k^i, \Delta t), \forall\ k = 0, ...,\ N-1, \tag{2.1b}$$

$$x_0^i = x^i(0), \tag{2.1c}$$

$$u_0^i = u^i(0), \tag{2.1d}$$

The first component of the cost function is defined in (2.2). This cost is a quadratic trajectory tracking cost that penalizes deviations from a given convoy trajectory, denoted as x_{traj}. In this chapter, the reference path x_{traj} is provided to the controller as either a predefined path (see Chapter 3, where these paths are provided in the task allocation phase) or one created for each agent by observing the motion of other agents in the convoy (as in [11, 21]). This cost is defined as

$$\begin{aligned} C_{traj}(X^i, U^i) = &\sum_{k=0}^{N-1} (x_k^i - x_{traj,k}^i)^T Q (x_k^i - x_{traj,k}^i) \\ &+ (u_k^i)^T R (u_k^i) + (x_N^i - x_{traj,N}^i)^T Q_f (x_N^i - x_{traj,N}^i), \end{aligned} \tag{2.2}$$

where both $Q \in \mathbb{R}^{m \times m}$ and $Q_f \in \mathbb{R}^{m \times m}$ are symmetric positive definite matrices, and $R \in \mathbb{R}^{n \times n}$ is a positive definite matrix.

The second cost term, C_{convoy}, penalizes deviance from the convoy structure over the future time horizon. This cost is defined in (2.3), where $x_k^{ref,i-1,i}, x_k^{ref,i,i+1}$ are the desired reference positions of the i^{th} agent given the positions of the $i-1$ and $i+1$ agents, respectively. Given our interest in maintaining a position with respect to both the leading and following agents, superscript ref refers to the reference point defined between the agents with the indices following ref (i.e., , either $(i-1, i)$ or $(i, i+1)$). Here, $Q_{lead} \in \mathbb{R}^{m \times m}$ and $Q_{follow} \in \mathbb{R}^{m \times m}$ are tunable positive semi-definite constant matrices. The second cost term is given as

$$C_{convoy}(X^{i-1}, X^i, X^{i+1}) = \sum_{k=0}^{N-1} (x_k^i - x_k^{ref,i-1,i})^T Q_{lead}(x_k^i - x_k^{ref,i-1,i}) + (x_k^i - x_k^{ref,i,i+1})^T Q_{follow}(x_k^i - x_k^{ref,i,i+1}). \tag{2.3}$$

As the run-time cost is evaluated over the same time interval, the run-time costs in (2.2) and (2.3) may be collapsed into a single quadratic cost expression. This new expression is defined as

$$g(x_k^i, u_k^i) = (x_k^i - Q_T^{-1} y_T) Q_T (x_k^i - Q_T^{-1} y_T) \ - (y_T)^T Q_T (y_T) + Z_T + (u_k^i)^T R(u_k^i), \tag{2.4}$$

where:

$$\begin{aligned} Q_T &= Q + Q_{lead} + Q_{follow} \\ y_T &= Q x_{traj,k}^i + Q_{lead} x_k^{ref,i-1,i} + Q_{follow} x_k^{ref,i,i+1} \\ Z_T &= (x_{traj,k}^i)^T Q x_{traj,k}^i + (x_k^{ref,i-1,i})^T Q_{lead}(x_k^{ref,i-1,i}) + (x_k^{ref,i,i+1})^T Q_{follow}(x_k^{ref,i,i+1}). \end{aligned}$$

A similar combination of quadratic expressions can be performed on the terminal cost, yielding $(Q_F,\ y_F,\ Z_F)$, respectively. These parameters may be used to rephrase the terminal cost:

$$\phi(X_N) = (x_N^i - Q_F^{-1} y_F) Q_F (x_N^i - Q_F^{-1} y_F) - (y_F)^T Q_F (y_F) + Z_F.$$

Thus, the cost function for agent i may be phrased as:

$$J = C_{traj} + C_{convoy} = \sum_{k=0}^{N-1} \{g(x_k^i, u_k^i)\} + \phi(X_N).$$

Furthermore, by linearizing the system around $x_k,\ u_k$ and defining $A_k = \frac{\partial}{\partial x_k} f(x_k, u_k)$ and $B_k = \frac{\partial}{\partial u_k}$, the optimization problem can be interpreted and solved online as an iterative Linear Quadratic Regulator (iLQR) ([24]). This yields a control law:

$$u_k^i = -[R_k + B_k^T P_{k+1} B_k]^{-1} B_k^T P_{k+1} A_k (x_k^i - Q_T^{-1} y_T) = -K_k (x_k^i - Q_T^{-1} y_T),$$

with P_k representing the solution to the Riccati Equation and K_k being the optimal control gain matrix.

Figure 2.4: Agents (platforms) similar to those demonstrated in this chapter

2.4.2 Controller implementation and design discussion

In this chapter, we demonstrate our approach on a set of small wheeled vehicles, shown in Figure 2.4. We modeled the vehicle dynamics as a nonlinear bicycle with the state: $\mathbf{x} = [p_x, p_y, \psi, v]$. The elements of this state vector are the vehicle's x-position, y-position, heading, and velocity, respectively. The available vehicle controls included velocity and steering angle: $\mathbf{u} = [a, \delta]$. We model the continuous-time vehicle dynamics as

$$\begin{aligned} \dot{p}_x &= v * \cos\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right), \\ \dot{p}_y &= v * \sin\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right), \\ \dot{\psi} &= \frac{v}{L} \cos\left(\operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right) * \tan(\delta) \\ \dot{v} &= a, \end{aligned} \tag{2.5}$$

and the characterizing platform parameters are listed in Appendix A in Table A.1.

The value of x^{ref} depends on the desired inter-robot distance, d^{ref}. We define the desired inter-robot distance as

$$d^{ref} = \lambda_1 v_t + \lambda_2 (v_i - v_{i-1}) + K,$$

where v_t is the desired target velocity and v_i corresponds to current velocity for agent i. λ_1 and λ_2 are tunable parameters where λ_1 and λ_2 are non-negative values. K is a constant minimum inter-robot distance for safe operation. In this implementation, we only consider the current velocities of vehicle i and the prior robot in the convoy structure.

As shown in Fig. 2.2, the reference positions for agents $i-1$ and $i+1$ over horizon $1:N$

are recovered by performing an open-loop forward rollout using the linearized dynamics at the $i-1$ or $i+1$ agent's state. The $i-1$ and $i+1$ agents' current velocity and steering are assumed to be constant over the rollout. The reference positions for (2.3) are set by moving backward d^{ref} along the convoy trajectory from the predicted $i-1$ agent positions X^{i-1} and moving the same distance ahead of the $i+1$ agent positions X^{i+1}. Thus, adjacent agents experience task execution coupling in both the position and velocity space.

The desired reactivity discussed in Section 2.3 is needed to prevent collisions between agents due to sudden variations in speed. To enable this behavior, the controller computes a weighting factor, w_{convoy}, between the costs (2.2) and (2.3) during run-time. We compute this factor given the desired convoy spacing and the current Euclidean distance $dist_{i,j}$ between agents i and j. These weights are multiplied to the Q matrices in (2.3) and the weighting factor is computed as:

$$Q_{lead} = w_{i,i-1} * Q_{lead}$$

$$Q_{follow} = w_{i,i+1} * Q_{follow}$$

$$w_{i,j} = \begin{cases} 1 + \frac{w_{far} \times (dist_{i,j} - d^{ref})}{dist_{i,j}} & \text{if } dist_{i,j} \geq d^{ref} \\ 1 + \frac{w_{near} \times (d^{ref} - dist_{i,j})}{dist_{i,j}} & \text{otherwise.} \end{cases}$$

The proposed formulation allows the robot to track its predecessor and follower through the predicted x^{ref} terms while also tracking its desired planned path. We can interpret the proposed additional convoy cost provided in the optimal control formulation as a modification of the local linearization point used in the LQR. This modification of the coordinate transfer from the reference path (x_{traj}) through both the user-defined weightings (Q, Q_{lead}, Q_{follow}) and reference trajectories of the leading (X^{lead}) and following (X^{follow}) robots in the fleet.

2.4.3 Local Planner and Trajectory Controller

In the absence of an obstacle-free path, Algorithm 1 ensures high-speed following and maintains the convoy formation with the help of an additional velocity scaling term:

$$v_T = (1 + \alpha) v_{tc}$$

$$\alpha = \lambda_3 (d_1 - d^{ref}) - \lambda_4 d_2$$

Where the λ's are tunable parameters, v_{tc} is the desired target velocity from the convoy

Algorithm 1 Convoy controller with obstacle avoidance for robot i

Input: Agent states (x_0^{i-1}, x_0^i, x_0^{i+1})
Output: Control sequence (U^i)

```
while Agents i ∈ I_L are in convoy do
    Run convoy controller, Section 2.4
    if Output path X^i is obstacle-free then
        return Control sequence U^i
    else
        Define D_LookAhead based on velocity v_t
        Calculate desired velocity v_T and direction θ_T
        Run the local planner [25]
        Get the modified obstacle-free path X^i
        Send X^i into a path following iLQR controller
        return Control sequence U^i
    end if
end while
```

controller, v_T is the modified desired target velocity for the planner, v_r is the robot's velocity, v_l is the leader's velocity, d_1 is the distance between the robot and the leader along the trajectory, and d_2 is the distance between the robot and the follower along the trajectory. We select the desired direction (θ_T) using a look-ahead distance $D_{LookAhead}$ along the optimal trajectory. This is done to track the desired controller path to the greatest extent before the robot switches back into the convoy controller mode.

The local planner takes these inputs and generates a feasible path that avoids obstacles while tracking outputs from the convoy controller. The planner we use is based on the planner defined in [25]. The selected path is then sent to an iLQR trajectory following controller to generate the control sequence.

2.5 Performance in simulation

We characterize the performance of our proposed controller ("Convoy Controller") in a variety of simulated environments. We simulate these environments, shown in Figure 2.5, using the Gazebo Physics Simulator [3]. As discussed in Section 2.2, there are many different approaches to enforcing convoy structure at the control and planning level. To develop a comparison between the presented methodology and existing literature, we create a baseline "Base Controller" that combines the local planning and distance variation behaviors from Zhao [26] with additional modifications [22, 27, 28]. This combination creates a decentralized controller, similar to our proposed method, that outperforms the

individual performance of each baseline work separately in terms of minimizing inter-agent distances without compromising the convoy formation structure.

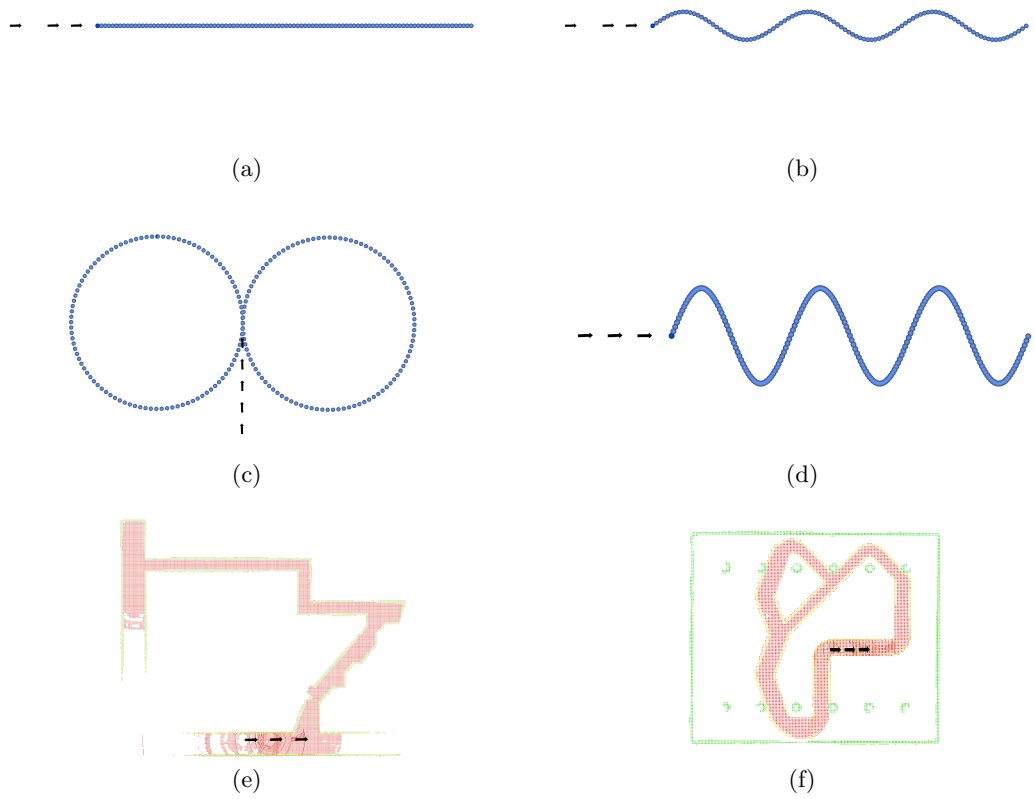


Figure 2.5: A depiction of the different simulation environments investigated in this chapter. Figure 2.5a represents a straight line path. Figure 2.5b depicts a path with low-curvature turns. Figure 2.5c shows a path with an ∞ loop track. Figure 2.5d reflects a sinusoidal path with tight turns. Figure 2.5e shows a simulated tunnel environment from the Gazebo Physics Simulator [3]. Finally, Figure 2.5f shows a racetrack environment, also from Gazebo [3]

2.5.1 Error metric design

We define two error metrics to compare the performance of our Convoy Controller with the Base Controller. The first metric is

$$e_{m_1} = \begin{cases} \text{abs}((d_{i-1,i+1}/2) - d_{i-1,i}) & i \in (1, L), \\ \text{abs}((d_{L-2,L}/2) - d_{L-1,L}) & i = L. \end{cases}$$

This metric aims to assess how well an agent can maintain a position midway between its adjacent agents. Variation in this metric provides insight into the accordion-like behaviors that negatively affect fleet performance when the individual agents continuously accelerate and brake during the operation. However, this error metric can maintain a low value if all agents maintain an equally large following distance, which is not desirable. To that end, we define a second error metric as follows:

$$e_{m_2} = \text{abs}(d_{i-1,i} - d_{desired})$$

This metric measures distance from the desired gap between robots and, along with the first metric, can provide a thorough understanding of the convoy performance.

2.5.2 Simulation results

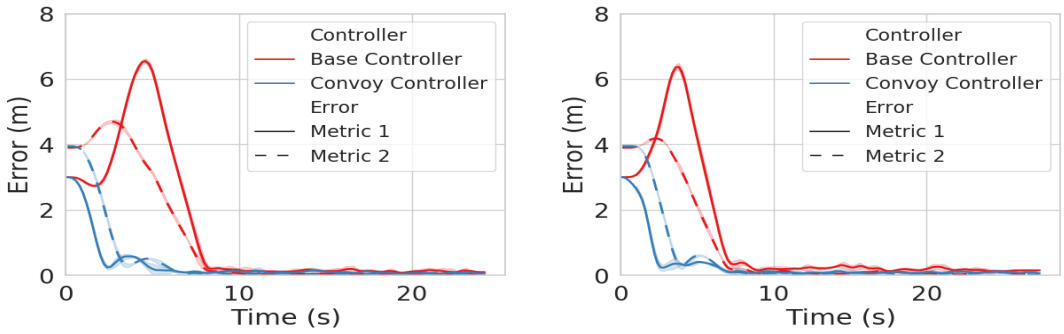
We first consider the straight-line and low-curvature turn paths. Following this, we simulated runs in environments with tight turns, including a tunnel environment and a racetrack environment. We placed additional constraints on agent operability and speed on these runs to understand formation performance in edge scenarios. Finally, to test controller performance in continuous turns at different speeds, we tested the controller with commanded speeds varying between 4 [m/s] and 8 [m/s] on an ∞ loop track with a radius of 20 [m].

Straight line and low curvature runs

We performed the first set of simulations on a group of agents operating on simple paths while starting from scattered initial locations. Our aim with this experiment was to understand how quickly the various controllers settle and how the error values vary during that period. As shown in Figure 2.6, our convoy controller settles faster while maintaining lower values for both error metrics. These charts illustrate the faster formation of convoy structures using our framework.

Tight turns - performance in constrained environments

In this simulated trial, we provided a series of waypoints only to the lead agent, not to the subsequent agents. On running the Base Controller, the following agents would tend to overshoot due to the presence of sharp turns. We observed this behavior even when we tuned down the look-ahead value to zero, indicating an inherent difficulty in maintaining the convoy structure with such sharp turns at high speeds. Figure 2.7a illustrates a comparison between the controllers on the error metrics defined earlier in this section.



(a) (b)

Figure 2.6: Simulation results for the straight line and low curvature paths

Tunnel environments - performance with discrete disturbances

Agents might get temporarily stuck during operation, especially in indoor environments at high speeds. To simulate this, we initially stopped the first robot in place for 8 [s] and then reverted its operation to nominal conditions. In Figure 2.7b, the error metrics are capped in our approach, whereas they continuously rise with the Base Controller as the last robot is unable to get back into the convoy. The reason for this difference arises from the cost terms corresponding to the convoy structural deviance penalty on both the leading and following agents. As soon as one of the agents fails to maintain the desired convoy structure, the inter-robot gaps increase, resulting in a corresponding rise in the cost term associated with the structural penalty. To minimize the total cost, the sharp increase in the penalty term causes the remaining agents to slow down and come to a stop until the slowed-down follower agent rejoins, thereby capping the error metric.

Racetrack - performance with unexpected agent heterogeneity

We are also interested in simulating a trial that would mimic an agent with an operational anomaly. To investigate this aspect, we commanded the convoy to travel at 4 [m/s] through a constrained outdoor environment; however, one of the robots was unable to achieve speeds higher than 3 [m/s]. We ran this setup in a racetrack environment, as shown in Figure 2.5f, with the associated error metrics presented in Figure 2.7c. The initial performance between the controllers is similar until the point where the leader makes a turn after the straight section on the racetrack. On the straight section, the Base Controller notices a continuous drop in performance that it is unable to recover past a turn, and the slow robot can no longer track the lead along the racetrack. Our system, on the other hand, adapts to this

agent and, despite a slight loss in performance, maintains an upper limit on the error metric, allowing the convoy to continue along the desired path.

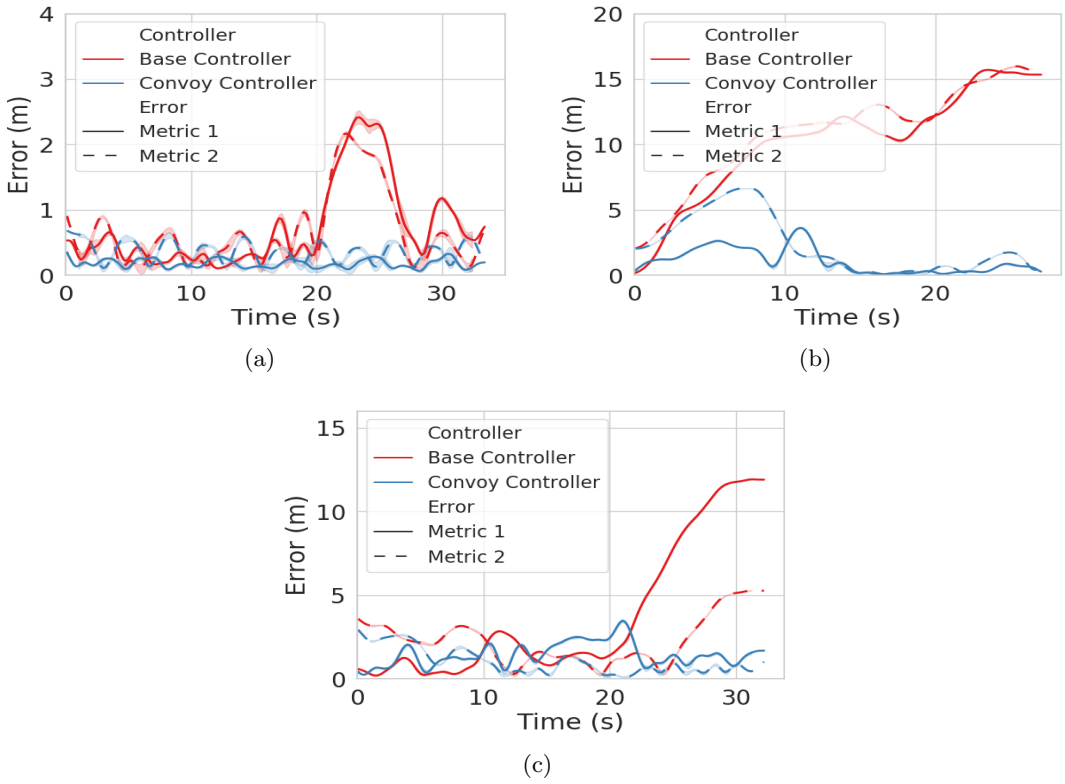


Figure 2.7: Simulation results which reflect edge cases for the controller performance. Tight turn environment performance is shown in Figure 2.7a, tunnel performance in Figure 2.7b, and on the racetrack environment in Figure 2.7c

Figure-eight loop - checking error metrics at variable speed

Finally, we ran the agents in an ∞ loop environment at various speeds to understand system capability limits. Figure 2.8 illustrates the error metrics at various target speeds between 4 [m/s] and 8 [m/s]. These figures demonstrate that at lower speeds, both controllers can stabilize the convoy formation. However, as speeds increased, only our Convoy Controller was capable of stabilizing the formation structure.

In summary, addressing all requirements from the problem definition in Section 2.3, our convoy controller achieves lower inter-robot distances, quickly adapts to environmental variations, and exhibits faster convergence for error metrics when compared to the Base Controller. Table 2.1 summarizes the results of all simulated trials and demonstrates that,

on average, our distributed convoy formation controller minimizes inter-agent distances over the Base Controller.

Table 2.1: Overall results comparison

Environment	Average Error Metric 1 (m)		Average Error Metric 2 (m)	
	Base Controller	Our Controller	Base Controller	Our Controller
Straight Line	1.35	0.31	1.08	0.44
Sine Curve	1.12	0.30	0.80	0.47
∞ loop	1.69	0.32	2.82	0.73
Tight Turn	0.72	0.18	0.65	0.37
Tunnel	10.20	1.08	10.94	2.24
Race Track	3.77	1.45	2.29	1.02
Column Room	1.91	0.97	5.61	0.95
Long Corridor	3.78	0.89	5.14	0.68

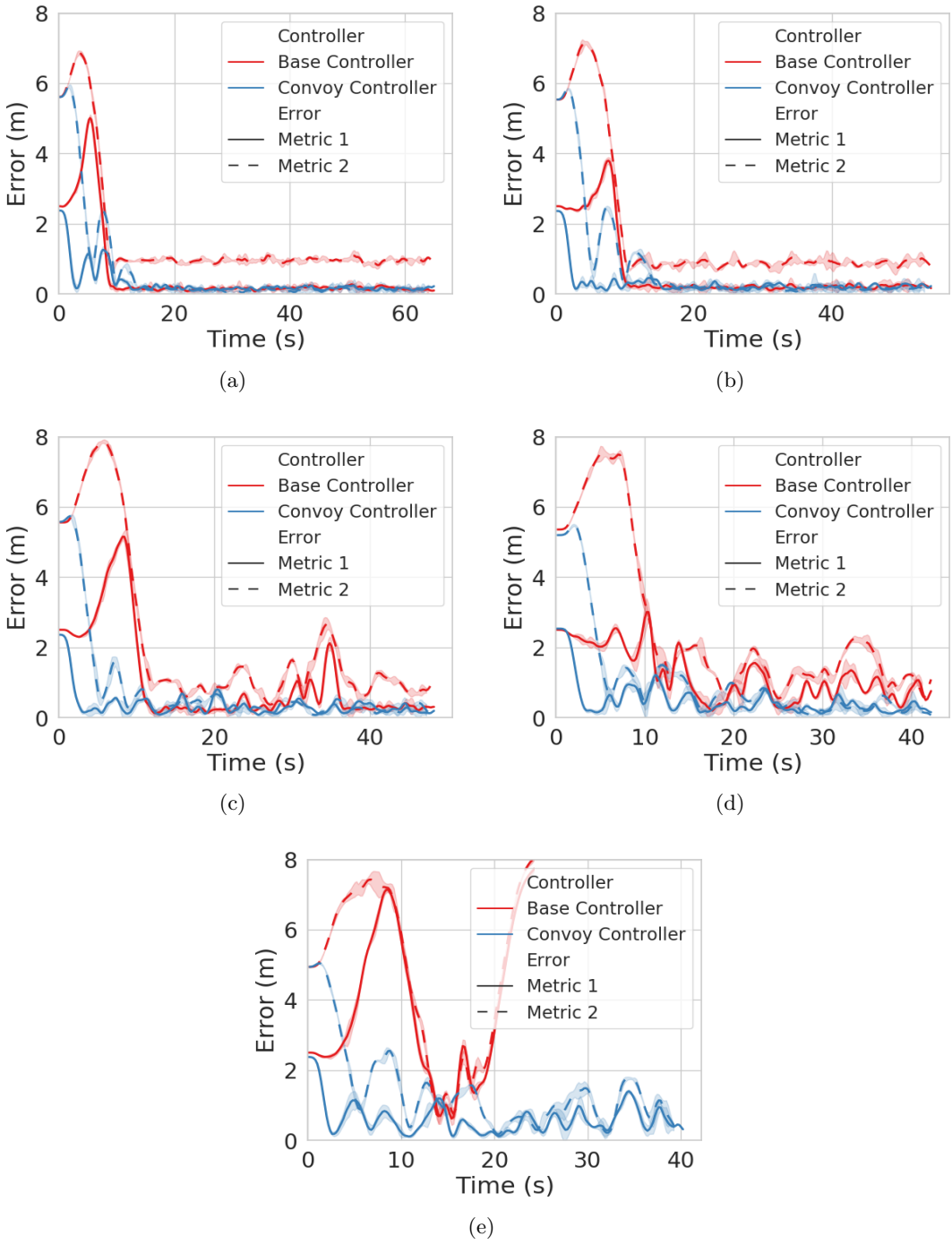


Figure 2.8: Speed changes in an ∞ loop. The speed increases by one meter-per-second from four meters-per-second (Fig. 2.8a) to eight meters-per-second (Fig. 2.8e)

Figure 2.9: A robot convoy posed to perform a search and rescue task.

2.6 Hardware Trials

To validate our controller performance, we conducted hardware experiments on a team of three ruggedized wheeled autonomous convoying robots (Fig. 2.9). We specifically designed the robots to operate in formations and perform search, rescue, and resupply missions. Each platform is equipped with a Jetson AGX Xavier for on-board compute. Furthermore, each platform is equipped with a custom payload that can be tailored for multiple possible missions and environmental conditions. In our experiments, each sensor payload comprises multiple RGB cameras (providing 360-degree coverage), an IMU, and a Velodyne 16 Lite LiDAR. The flexibility of each chassis and the symmetry between multiple hulls make the coalition of vehicles a convenient tool for multi-agent research. Please see Appendix A for more details about the convoying system.

2.6.1 Hardware trial testing environments

We conducted tests in multiple environments representative of indoor urban spaces. These environments not only contained long open corridors (Figure 2.10a and Figure 2.10c), but also tight enclosed spaces (Figure 2.10b). As can be seen in these photos, each testing environment had small environmental debris present throughout the operating area. The author would like to note that none of this environmental debris fully obstructed the agents' paths. We show different debris types in more detail in Figure 2.11.

(a)

(b)

(c)

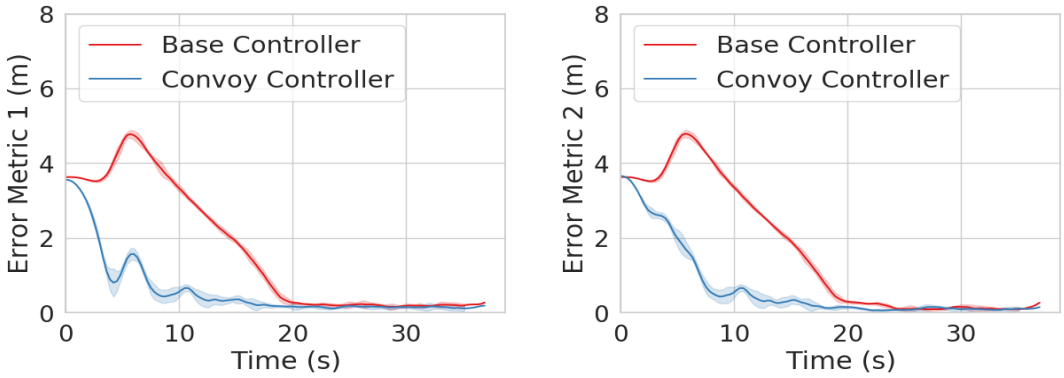
Figure 2.10: Pictures of the hardware trial environments. (Straight Line (Fig. 2.6a), Column Room (Fig. 2.10b), and longer runs (Fig. 2.10c))

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

Figure 2.11: Ground conditions for convoy testing which included puddles (Fig. 2.11a), carpets (Fig. 2.11b), stones (Fig. 2.11c), and broken floors (Fig. 2.11d)

2.6.2 Straight line tests - validating simulation results

We first tested the Convoy Controller in similar conditions to those tested in Section 2.5.2. The hardware trial results validate the simulations as the error metrics in Figure 2.12a and Figure 2.12b vary similarly to those in Figure 2.6a and Figure 2.6b.



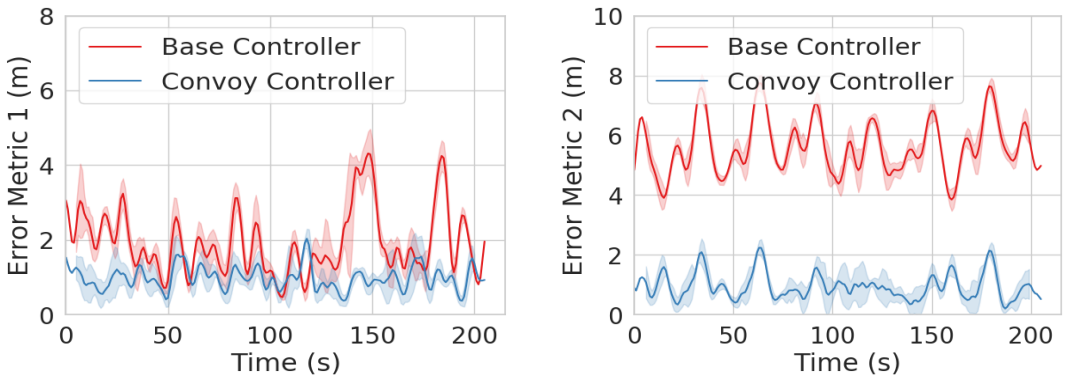
(a) (b)

Figure 2.12: Straight line test performance depicting error metric 1 (e_{m_1}) (Fig. 2.12a) and error metric 2 (e_{m_2}) (Fig. 2.12b).

2.6.3 Column room - convoy agility tests

We selected the environment shown in Figure 2.10b to test convoy agility. In particular, we were interested in testing the Convoy Controller's capability to maintain a stable convoy structure during high-speed maneuvers in obstacle-dense environments. To safely travel

through the columns, each agent must perform low-curvature maneuvers while continually avoiding the environmental obstacles (columns). The room has dimensions of 35 [m] by 20 [m]. In the test, the operator commanded a general direction of motion to the lead vehicle, and all three robots made use of the convoy controller to maintain the convoy formation at 4 [m/s]. The agents make continuous turns to avoid hitting walls, columns, and other obstacles. Over the span of the test, the agents covered a total distance of around 600 [m]. Figure 2.13a and Figure 2.13b illustrate the Convoy Controller's performance and the Base Controller's performance on the error metrics defined in Section 2.5. The Convoy Controller maintained an inter-agent gap of just over 5 [m] at a 4 [m/s] operating speed compared to the 9 [m] gap achieved by the Base Controller.

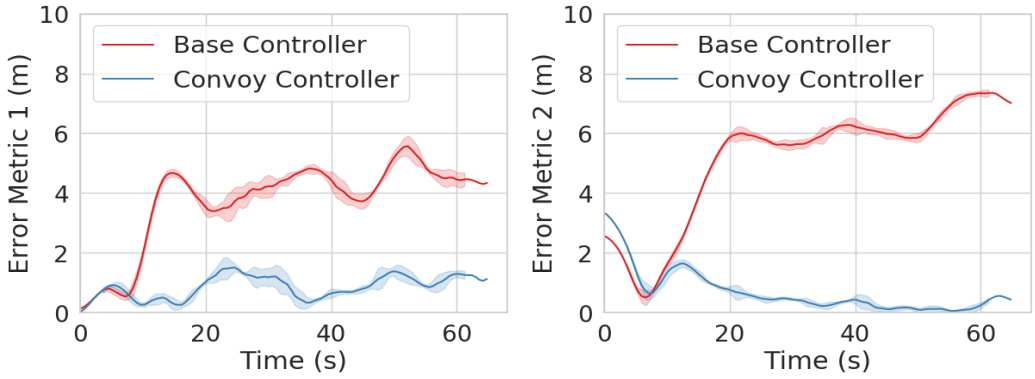


(a) (b)

Figure 2.13: Hardware tests demonstrating agile driving scenarios in the column room. error metric 1 (e_{m_1}) is illustrated in Figure 2.13a, and error metric 2 (e_{m_2}) is illustrated in Figure 2.13b.

2.6.4 Long corridors - endurance performance

In the final hardware trial, we demonstrate high-speed convoy performance in urban environments with turns and doorways (Figure 2.10c. Figure 2.14a and Figure 2.14b again illustrate the performance difference between our Convoy Controller and the Base Controller. The average inter-robot distances achieved by our Convoy Controller are just under 4 [m] at high speeds, which is significantly lower than the lowest gaps of 9-15 [m] maintained by the Base Controller. The deviations in the metrics occur only when the robots make a turn into another corridor, move over rough terrain, or navigate through doorways. The agents were able to navigate over 3 [km] in such environments without running into each other or leaving a robot behind, displaying a robust and safe system.



(a) (b)

Figure 2.14: Hardware tests for long runs, including error metric 1 (e_{m_1}) in Figure 2.14a and error metric 2 (e_{m_2}) in Figure 2.14b.

2.7 Conclusions

In this chapter, we designed an optimal control system capable of performing multi-agent convoying tasks. Our design incorporates future predictions about the adjacent agent's motion, and considers both the leading and trailing agents, enabling the agents to operate closer to each other at high speeds. This, in turn, enabled the agents to operate well in environments that require continuous sharp turns and variations in speeds. We have demonstrated, through simulation and hardware experimentation, an improvement in performance over current state-of-the-art methods, and have reduced following distances against state-of-the-art methods by half. The system can potentially be improved in the future by incorporating environment-dependent control optimizations and integrating hybrid communication control methods with increased data transfer to reduce computational loads associated with state prediction.

Chapter 3

A Synchronized Task Formulation for Robotic Convoy Operations

The synchronized routing schema for robotic convoy operations outlined in this chapter was accepted for publication in the IEEE Robotics and Automation Letters in 2025

Acknowledgments:

> The authors would like to acknowledge the contributions of Mr. Burhanuddin Shirose for his assistance with the hardware experiments conducted in the paper.

Citation:

> **C. Noren**, B. Vundurthy, S. Scherer, H. Choset and M. Travers, "A Synchronized Task Formulation for Robotic Convoy Operations," in *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 6808-6815, July 2025

3.1 Overview

Now that we have designed a multi-agent convoying behavior that enables a team of agents to travel between two locations, we are interested in leveraging this behavior to perform more complicated missions. Future ground logistics missions may require multiple robotic agents to travel in a convoy between multiple locations of interest. As each location may require a different number of agents (e.g., resupply vehicles), these missions will require a mutable convoy formation structure that may be divided to meet operational needs at each location. While we assume we know the number of different required robotic agents at each location, allocating specific agents to a particular convoy and assigning that convoy to a specific location is non-trivial. Addressing this mission-level allocation problem is the focus of the top layer of our architecture, as shown in Figure 3.1. As such, we focus on the development of the task allocation layer of our architecture in this chapter.

We model this mission type by modifying the vehicle routing problem with multiple synchronizations (VRPMS) to enforce convoy constraints (VRPMS-CC). This centralized approach to organizing and routing convoys is represented as a graph-based routing problem and then solved as a mixed integer program. A solution of the VRPMS-CC forms convoys by ensuring that agents participating in the same convoy remain spatially and temporally coupled, traversing the same edge of the graph simultaneously. We demonstrate our approach through numerical studies, where we route up to six simulated agents through twenty convoying tasks, as well as on robotic hardware. These demonstrations motivate two further contributions to specialize our approach to robotic systems. These contributions

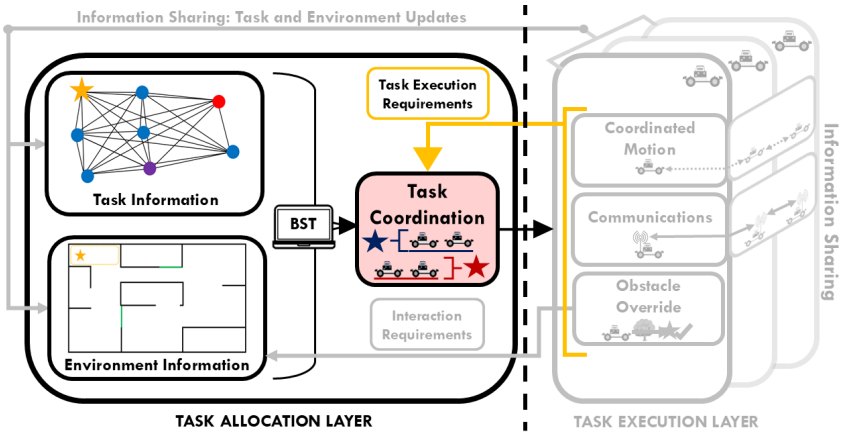


Figure 3.1: Task coordination forms the basis of the task allocation layer of our framework. We focus on developing the task allocation layer in this section that leverages the functional convoy task abstraction developed in Chapter 2.

are:

1. a warm-starting heuristic that improves solver times by up to eighty-nine percent and
2. an online multi-depot variant of the VRPMS-CC that responds to *a priori* unknown impassable environmental obstacles.

3.2 Introduction

For hundreds of years, convoy formations have served as the backbone of large economic and military transport operations [5]. Although prior work has addressed challenges in conducting robotic convoy operations, autonomously generating convoy formations (or "convoy details") and allocating routes to multiple robots remains a daunting challenge for complex missions [8]. In complex missions, the number of required robotic platforms may vary location-to-location (e.g., higher-risk areas), requiring multiple details or modifications to the convoy formation topology. Modeling these missions and developing formation-aware routing algorithms are the next steps toward solving this challenge.

The key challenge for formation-aware routing algorithms is synchronizing the routes of individual agents participating in a formation. Classically, routing problems construct each route as a sequence of tasks that an agent must visit, but few works require the synchronized action of agents when traveling between tasks. As the optimal allocation of routes to agents is a known NP-hard problem [29], adding additional synchronization constraints can make an already complex problem even harder. While multiple works enforce synchronization constraints between a pair of agents, convoy formation topologies contain more than two agents. These topologies add complexity to the convoy operation as they require the synchronization of multiple agents assigned to the formation.

To address this variable formation topology systematically, we introduce a set of "convoy constraints" to the vehicle routing problem with multiple synchronizations (VRPMS). These convoy constraints synchronize the motions of multiple agents into different-sized convoys between locations of interest. Our first contribution is this VRPMS variant, known as the vehicle routing problem with multiple synchronizations - convoy constraints (VRPMS-CC). The solution to an instance of the VRPMS-CC is an optimal set of routes for all participating agents. To our knowledge, this is the first work to employ VRPMS for convoy operations.

The convoy constraints inherently impose a spatial and temporal structure, effectively clustering agents into synchronized groups (treated as distinct depots). This structure enabled us to formulate a multi-depot vehicle routing problem (MDVRP-SV) heuristic, our second contribution. By leveraging this clustering, we can search a reduced space of the initial problem, albeit at the loss of guaranteed solution optimality. However, by leveraging

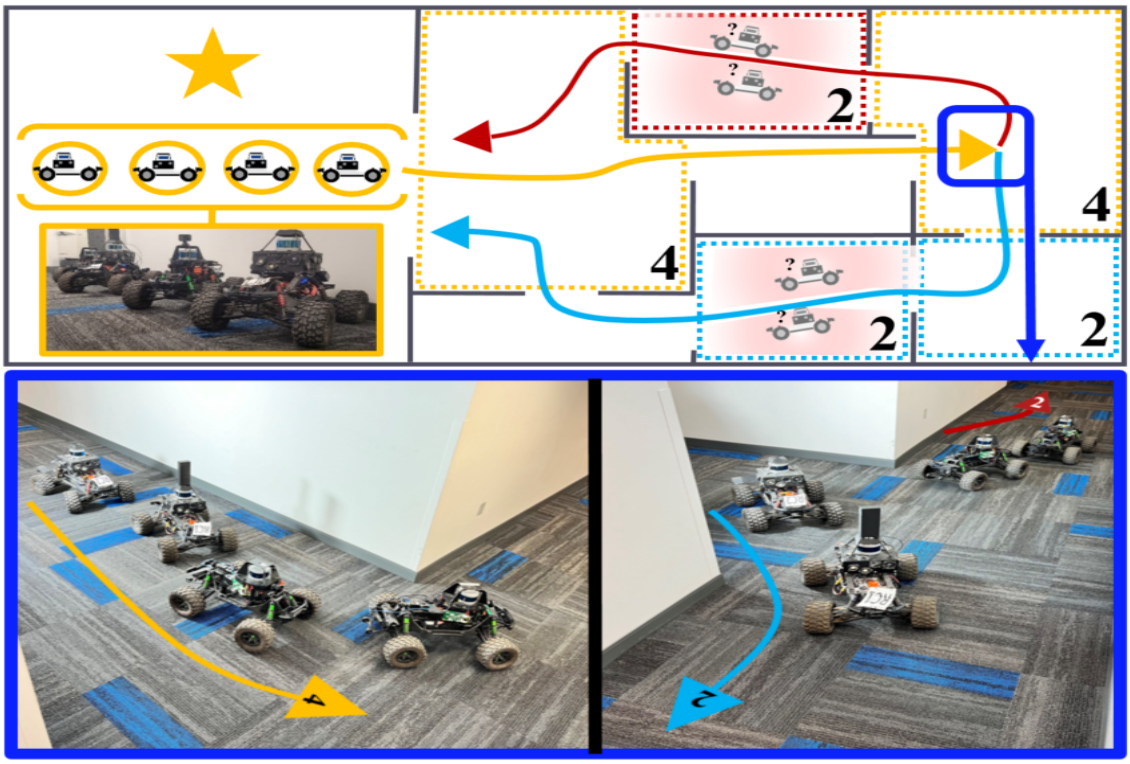


Figure 3.2: A team of four agents routing through a sample floor plan (top) where each room in the floor plan may require a different number of agents (black numeral) and the agents must travel as a convoy. The approach forms a single large convoy (gold) before dividing into two smaller convoys (red, blue), each containing two agents. The blue box depicts two points in time at the location of the division. The image at the earlier time (left) captures the arrival of the four-agent convoy, and the second image (right) depicts the convoy's division into two two-agent convoys. While not depicted in this figure, our allocation scheme can continue this decomposition, serving different locations with different needs. Note that specific mission requirements enforce this convoy decomposition strategy, but different mission parameters could relax these requirements, enabling compositional teaming as well.

this heuristic solution as a warm start, we can achieve a reduction of up to 89% in solver time.

Finally, these planned routes can face disruptions when *a priori* unknown impassable obstacles arise in multi-robot deployments. To address this, our final contribution is the Dynamic VRPMS-CC (DVRPMS-CC), an approach that dynamically resolves the VRPMS-CC in response to extroceptive stimuli reported by the robots. We demonstrate our contributions through hardware trials on our custom fleet (Fig. 3.2) of robotic agents, which are identical to those used in Chapter 2. This dynamic routing approach ensures

robust convoy operations in real-world environments.

3.3 Literature Review

The core of formation-aware routing approaches for robotic convoys resides in routing-based synchronization [30], where mathematical models enforce coordinated agent movement—the primary focus of this paper. However, to contextualize our approach, we will also examine multi-agent pathfinding (MAPF), market-based synchronization, multi-agent reinforcement learning (MARL), and dynamic grouping. These alternative methods often lack the en-route synchronization and dynamic formation management crucial for convoy operations.

3.3.1 Routing-based Synchronization

Routing problems often employ synchronization constraints within their mathematical models to couple the actions of agents. We focus on two primary types: movement (spatial) and operational (temporal) synchronization, both of which are relevant to convoy operations. Movement constraints coordinate agents along the same route, forming dynamic teams that work together. Research on these constraints is limited, with the tractor-trailer routing problem (TTRP) being a notable exception [31]. While the TTRP, which involves heterogeneous tractor and trailer agents, shares similarities with our problem, its tractor-trailer pairing differs from our flexible homogeneous convoy formations.

On the other hand, operational constraints synchronize agent arrival times, common to interlogistical applications such as the Traveling Salesman Problem with Drone (TSP-D) [32]. The TSP-D problem seeks to periodically synchronize the motion of a drone with a ground agent. However, unlike our convoy operations, the TSP-D allows for independent operation and reward collection. Column generation and branch-and-cut algorithms [33, 34] are commonly employed for exact solutions techniques, while heuristics such as local search and greedy techniques address larger instances [32, 33].

3.3.2 Synchronization in multi-agent pathfinding (MAPF)

Synchronization in MAPF literature is contained mainly by two works: 1) multi-agent teamwise-cooperative path finding (MA-TC-PF) [35] and 2) cooperative MAPF (Co-MAPF) [36]. These works seek to assign and plan collision-free paths for teams of agents to different goal locations. MA-TC-PF utilizes a modified conflict-based search (CBS) to optimize actions for pre-organized teams, which makes it unsuitable for our problem, where team formation is a core component. Alternatively, Co-MAPF performs simultaneous task

assignment and pathfinding using a modified CBS for synchronization, but limits this to agent pairs at meeting locations, similar to TSP-D solutions. Co-MAPF primarily focuses on intralogistical applications, such as operations in warehouses using mobile manipulators, and does not address en-route synchronization needed for convoy operations.

3.3.3 Market-based Synchronization

Market-based approaches utilize distributed auctioning protocols for decentralized task assignment, incorporating synchronization constraints within their auction mechanisms [37]. While potentially underperforming traditional routing methods [38], they offer scalability and inherent decentralization. Designing robust auction mechanisms poses challenges due to potential agent misrepresentation and prioritization of individual rewards at the team's expense. Similar to the Co-MAPF, market-based approaches often prioritize operational synchronicity at meeting points rather than continuous en-route synchronization, limiting their applicability in scenarios requiring a maintained formation.

3.3.4 MARL Synchronization and Dynamic Grouping

Cooperative MARL learns policies for agent coordination. While scalable and adaptive, these policies often lack guaranteed coordination and may converge to sub-optimal solutions. Given the training complexity, many MARL methods focus on solving sequential rather than combinatorial tasks [39]. Deep reinforcement learning applied to TSP-D [40] uses a hybrid attention long short-term memory model for coordination. However, its effectiveness with larger numbers of agents (only two are considered in [40]) or highly-coupled rewards is unclear. Furthermore, the learned TSP-D policies do not generalize well to basic TSPs, indicating strong problem constraint influence on policy performance. Dynamic grouping, a crowd simulation technique, coordinates simulated agents into dynamic sub-crowds based on relative movements of local agents [41]. This method assumes pre-defined task assignments, similar to MA-TC-PF, and lacks a mechanism to enforce required group sizes.

3.4 Problem Formulation

In a VRP, multiple vehicles (also referred to as "agents" or "platforms") are tasked with visiting a set of locations within an environment. The platforms usually start and end at a specific location known as a "depot." A classic extension to the VRP is the VRP with Time Windows (VRPTW)s [42], which incorporates two additional time-based constraints. These constraints are:

1. each location must be visited during a specified time window, and
2. each location must be visited for a set amount of time (known as the "service time").

The proposed VRPMS-CC model is a further extension of the VRPTW that adds the requirement that multiple platforms must visit a given location while traveling in a convoy. We define the number of required platforms as a location's "support value," which may vary between locations. In order to ensure that the agents travel as a single convoy between locations, we place two additional constraints on the platform's motion: C1) all platforms in a convoy that arrive at a location must depart from the same location, and C2) all platforms in a convoy must arrive at a location together (temporally). A reason for such a motion restriction includes the requirement that convoy elements travel as a formation to defend themselves [7].

For future work, we introduce the concept of "clustering rules" as a generalization of convoy motion constraints C1 and C2. These rules serve as mission-specific customizable constraints, enabling the VRPMS-CC to be tailored to specific problem contexts (e.g., relaxing C1 could liberate some routes from requiring convoy formation, while modifying both C1 and C2 could enable compositional team formation at different locations.)

3.5 Problem Modeling

In this section, we formulate the VRPMS-CC as a mixed integer linear program (MILP), in which a group of robotic platforms must be assigned to convoy details.

Consider a homogeneous fleet of robotic platforms, $\mathcal{K} = \{k_1, k_2, \ldots, k_{|\mathcal{K}|}\}$, which must complete a set of convoy details that end at locations: $\mathcal{N} = \{n_1, n_2, \ldots, n_{|\mathcal{N}|}\}$. To complete a convoy detail, a convoy containing $v_i \in \mathbb{Z}^+$, $v_i \leq |\mathcal{K}|$ platforms must be assigned to visit the location. We define v_i as the support value of the convoy task that ends at the location n_i and define the set $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{N}|}\}$. All platforms are required to begin and end their routes at a depot location, denoted d^- and d^+, respectively. Thus, for each robotic agent k, the action sequence $r_k = (l_0, l_1, l_2, \ldots, l_g, l_{g+1})$ with $l_0 = d^-$, $l_{g+1} = d^+$, and $L = \{l_1, \ldots, l_g\} \subseteq \mathcal{N}$ not only describes the motion of the platform, but also implicitly encodes the assignment of agent k to a convoy detail. To enforce convoy motion constraints, we propose a routing model that atomizes a convoy detail into elements that can be assigned to individual vehicles. Convoys are then implicitly modeled by synchronizing the movements of individual vehicles across each vehicle's action sequence.

The task-based model is generated using the location set $\mathcal{N}$ and the support value of each location. Consider a set $\mathcal{S}$ where the elements of $\mathcal{S}$ are sets that describe the atomized elements ("tasks") of a convoy detail. A "task set" for a convoy detail that ends

at location n_i is defined as: $\mathcal{S}_i = \{\{s_i^{(1)}, \ldots, s_i^{(v_i)}\}\}$. The same task representation is also made concerning the depot, with $v_{d^-} = v_{d^+} = v_{|\mathcal{K}|}$. This produces two depot task sets: $\mathcal{S}_{d^-} = D^-$ and $\mathcal{S}_{d^+} = D^+$. The set of all non-depot tasks is $\mathcal{S}_T = \{\bigcup_{i \in \{1,\ldots,|\mathcal{N}|\}} \mathcal{S}_i\}$ and the set of all tasks is $\mathcal{S} = \mathcal{S}_T \cup D^- \cup D^+$.

The task information above may be structured into a connected weighted symmetric task graph $G_t = (V_t, E_t)$, where the vertex set $V_t = \{s | s \in \mathcal{S}_i \in \mathcal{S}\}$ consists of all tasks contained in $\mathcal{S}$. The edge set $E_t = \{e = \{s_i, s_j\} = \{s_j, s_i\} | s_i \in \mathcal{S}_p, s_j \in \mathcal{S}_q, \mathcal{S}_p, \mathcal{S}_q \in \mathcal{S}, p \neq q\}$ has an associated traversal cost $c_{ij}^k \in \mathbb{R}^+ \, \forall \, (i,j) \in E_t$, which is incurred when a platform moves between tasks i and j. Additionally, associated with each convoy detail ending at location n_i (and thus all tasks in the corresponding $\mathcal{S}_i \in \mathcal{S}$) is a time window for completion $TW_i = [a_i, b_i]$ and required service time, st_i, which denotes how long a task takes to complete. Finally, the travel time between tasks $(i,j) \in E_t$ is defined as t_{ij}. Thus, an instance of the VRPMS-CC is uniquely defined by the graph $G_t = (V_t, E_t, c_{ij}^k, TW_i, st_i)$.

Our mixed integer programming model involves three types of decision variables. The first variable is common in three-index platform flow routing problem formulations. For all edges in the task graph $(i,j) \in E_t$ and platforms $k \in \mathcal{K}$, define a binary variable x_{ij}^k, which describes the edge-flow over the task graph for a platform (k).

$$x_{ij}^k = \begin{cases} 1 & \text{if } (i,j) \in E_t \text{ is traversed by platform } k \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The second type of variable is a binary flow variable capturing convoy flow across task sets

$$d_{pq} = \begin{cases} 1 & \text{if } (\mathcal{S}_p, \mathcal{S}_q) \in \mathcal{S}_T \text{ is traversed by a convoy} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The last type of variable is a timing variable $T_{ik}, i \in V_t$ for each subtask and platform that specifies the time when platform k begins subtask i.

We now present the **Vehicle Routing Problem with Multiple Synchronizations - Convoy Constraints (VRPMS-CC)**. This model incorporates the clustering rule, which stipulates that the number of platforms traveling in convoy between two locations must be equal to the support value at the successor location. The VRPMS-CC is formulated as:

$$\min_{x_{ij}^k, d_{ij}, T_{ik}} \quad J = \sum_{k \in \mathcal{K}} \sum_{(i,j) \in E_t} c_{ij}^k x_{ij}^k \tag{3.3}$$

$$\text{s.t.} \sum_{d \in D^-} \sum_{j \in \mathcal{S}_T} x_{dj}^k \leq 1 \quad \forall\, k \in \mathcal{K}, \tag{3.4a}$$

$$\sum_{d \in D^+} \sum_{i \in \mathcal{S}_T} x_{id}^k \leq 1 \quad \forall\, k \in \mathcal{K}, \tag{3.4b}$$

$$\sum_{i \in \mathcal{S}_\mathcal{T}} x_{ij}^k - \sum_{i \in \mathcal{S}_\mathcal{T}} x_{ji}^k = 0 \quad \forall\, j \in \mathcal{S}_\mathcal{T},\, k \in \mathcal{K}, \tag{3.4c}$$

$$\sum_{k \in \mathcal{K}} \sum_{i:(i,j \in E_t)} x_{ij}^k = 1 \quad \forall\, j \in \mathcal{S}_\mathcal{T} \tag{3.4d}$$

$$(T_{ik} + st_i + t_{ij} - T_{jk}) \leq M_t(1 - x_{ij}^k) \quad \forall k \in \mathcal{K}, (i,j) \in E_t, \tag{3.5a}$$

$$a_i \leq T_{ik} \leq b_i \quad \forall k \in \mathcal{K},\, i \in \mathcal{S}_T \tag{3.5b}$$

$$T_{ik} - T_{ik'} = 0 \quad \forall k, k' \in \mathcal{K}, \quad i \in \mathcal{S}_T, \tag{3.6}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{S}_i} \sum_{j \in \mathcal{S}_j} x_{ij}^k = v_j\, d_{pq} \quad \forall\, (\mathcal{S}_p, \mathcal{S}_q) \in \mathcal{S}_T, \tag{3.7a}$$

$$\sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{S}_m} \sum_{j \in \mathcal{S}_j} x_{mj}^k \leq M_R(1 - d_{pq}) \quad \forall (\mathcal{S}_p, \mathcal{S}_q) \in \mathcal{S}, \mathcal{S}_m \in \mathcal{S} \setminus (\mathcal{S}_p, \mathcal{S}_q) \tag{3.7b}$$

$$\sum_{\mathcal{S}_p \in \mathcal{S}} d_{pq} = 1 \quad \forall\, \mathcal{S}_q \in \mathcal{S}, \tag{3.7c}$$

$$x_{ij}^k \in \{0,1\}, \quad d_{pq} \in \{0,1\}, \quad T_{ik} \in \mathbb{R}^+. \tag{3.8}$$

Constraints (3.4a), (3.4b), and (3.4c) describe sequence constraints (platforms start at the initial depot, platforms end at the final depot, and platform route-flow constraints). Constraint (3.4d) ensures that each task is assigned to a single agent. Constraints (3.5a) and (3.5b) ensure feasibility with respect to the task time windows. Temporal synchronicity is provided by (3.6), which states that the start time of any given task is the same across all platforms in a convoy. Movement synchronicity is enforced by (3.7a), (3.7b), and (3.7c). Constraint (3.7a) ensures that v platforms are used to visit a location, (3.7b) ensures that

all platforms arrive at a successor task from the same predecessor task, and (3.7c) ensures that all platforms arrive only from one other task group. Hyperparameters M_t and M_R are two "Big-M" values that may be selected by finding the upper-bound of the left-hand side of constraints (3.5a) and (3.7b) as outlined in [42, 43].

3.6 Numerical Studies

We tested the model presented in Section 3.5 on modified variants of both Solomon's [44] and multi-agent pathfinding [45] (MAPF) benchmarks. These studies motivate the introduction of a warm-start routing heuristic (in Section 3.6.3) that empirically demonstrates a decrease in solver time.

3.6.1 Experimental Design

Each VRPMS-CC instance is uniquely defined by the graph $G_t = (V_t, E_t, c_{ij}^k, TW_i, st_i)$. We provide the information used to construct this graph to the solver in the form of an environment map, a list of locations, the support value at each location, and the timing information for each location. All instances considered in this chapter are solvable.

Environment Maps

Each VRPMS-CC instance was associated with an environmental map to introduce an idea of environmental obstacle avoidance into the study. These obstacles directly affect the cost associated with traveling between locations. We represent this environment map as a binary 2D occupancy grid that captures information regarding impassable obstacles. The "floor plan" studies were associated with environmental maps that are presented in this chapter. MAPF environments are referenced by name from the open-source MAPF benchmark [45]. All other trials are associated with environmental maps that do not include obstacles.

Task Graph Vertex and Edge Definitions

Graph vertices are populated by creating duplicate tasks at each location equal to the support value associated with that location. Locations were either provided (Solomon's benchmark) or randomly sampled from non-occupied cells (Floor Plan, MAPF). For Solomon's benchmark, we randomly select six to eight locations from [44]. Tests with designed location layouts will be denoted in the discussion of that test. As neither the MAPF benchmark nor the Solomon benchmark was initially constructed with the concept of support value, unless noted otherwise, a random support value between $S_i \in [2, K]$ was selected for each location.

Graph edges are defined between all vertices. Each edge is weighted by finding the shortest obstacle-free path between the locations (vertices). Vertices located at the same location have a relative distance of zero. (Note that constraints (3.7a), (3.7b) ensure that different agents are assigned tasks at the same location). All results are denoted with straight arrows, which represent the obstacle-free paths of the vehicles.

Timing information

Unless specified in the benchmark (i.e., Solomon's), locations are assigned time windows with the width of the mission duration. Estimated traversal times were computed by finding the shortest obstacle-free path distance between two locations and then dividing that path distance by an assumed average velocity. Unless indicated otherwise, all locations have a service time set at 10 [s].

Solver Information

Each VRPMS-CC instance was solved using the Gurobi Optimizer (Version 10.03) on a mobile workstation with an AMD Ryzen 7 4800H CPU with 8 cores and 16 threads. Each test was conducted twice, once with Gurobi's internal solver presolve accelerations and once without. As Gurobi's presolve accelerations decrease problem solve times by an order of magnitude (average: 98% across all Solomon trials), we provide results with and without the presolve accelerations. Such results are presented not only to demonstrate the computational complexity empirically, but also to provide insight into solver performance when the presolve is not available. For all trials, we set the solver time limit at 7200 [s] and an optimality gap of 0.01. If a solver did not resolve in 7200 [s], it will be marked as "Did Not Finish" or "DNF".

3.6.2 VRPMS-CC Numerical Studies

As a solution to an instance of the VRPMS-CC may contain complex convoy routing behaviors, we utilize the floor plan environment as an illustrative example. The individual robotic platform behaviors captured in the floor plan environment are also present in the tabulated solutions.

Demonstration: Floor Plan A Routing

Figure 3.3 depicts a patrol scenario in which five robotic agents must visit three locations of interest with different support values. The routes taken by each platform are shown in Fig. 3.3a, and the aggregate convoy routes are shown in Fig. 3.3b. The alignment of

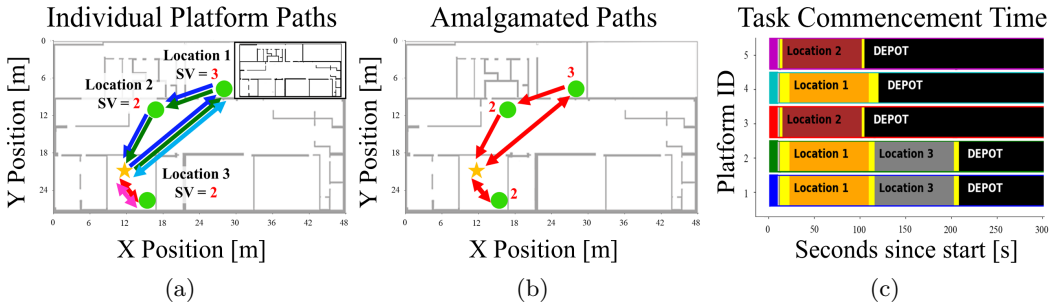


Figure 3.3: An example security patrol mission on "Floor Plan A" where a team of five robotic platforms must visit three locations of interest.

the task commencement times in Fig. 3.3c demonstrates the temporal synchronicity in the movements of the agents.

The agents start at the depot and are initially organized into two separate convoy details of sizes two and three. The size-two detail travels to Location 3 before returning to the depot (marked by a bidirectional arrow). The size-three detail first travels to Location 1 before splitting into a new detail of sizes one and two. The singular agent returns to the depot as it is not needed at Location 2. The size-two detail proceeds to service the task at Location 2 before returning to the depot. Compared to maintaining a static size-three detail, splitting the convoy details decreases the total distance traveled by 1.3%. The amount of savings increases for every robot that is prevented from making unnecessary trips.

Study: Solomon's benchmark

In this chapter, a non-capacitated version of Solomon's benchmark was used to evaluate the VRPMS-CC. Two variants of each trial were considered: one that maintains Solomon's original timings and one that replaces these timings with wide windows and service times of magnitude zero. While the first variant represents missions or operations with tight time schedules, we have found the second variant to be a representation of many robotic missions where such tight time window constraints may not naturally emerge. Each instance was conducted for a team of six robots with twenty-eight tasks.

The results of the studies demonstrate that the modeled approach correctly routes convoys in the presence of timing constraints. An interesting trend is that loose timing constraints yield longer solve times than tight timing constraints. Table 3.1 shows that both the presolve accelerated and the non-accelerated cases take advantage of the strict timing constraints to order the visitation of the convoys across the environments (e.g., the solve times for wide time windows in *C101* are greater in both columns). On average, wider

Table 3.1: Numerical Studies demonstrate that the VRPMS-CC instance can route convoys with and without timing constraints

Map Name	Solve Time: (Normal / Wide) (No Presolve) [s]	Solve Time: (Normal / Wide) (Presolve) [s]
C101	26.7 / **DNF**	1.9 / 43.8
C102	1222.7 / 6772.9	14.1 / 15.1
C201	168.8 / **DNF**	3.9 / 26.8
R101	39.8 / 1062.1	1.5 / 16.6
R102	426.4 / 529.5	12.6 / 18.1
R201	149.7 / 6731.7	2.8 / 19.6

time windows required an increase in solve time of 77.6% and 80.4% with and without the presolve, respectively. The observation inspires the use of wide time windows in all the following investigations in this chapter.

Study: MAPF Benchmark Performance

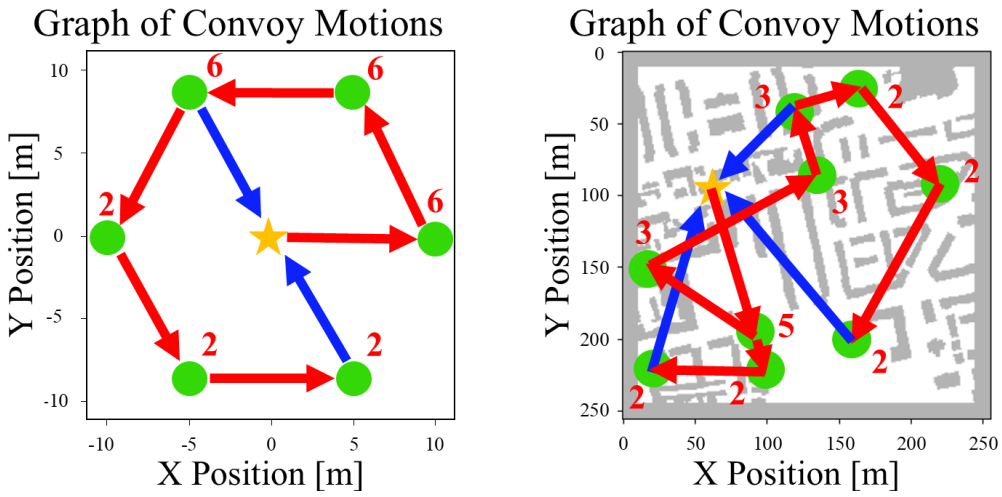
The MAPF benchmarks are tailored to represent environments in which robotic systems are expected to operate. The required obstacle avoidance behavior, considered in these maps, is included during the evaluation of the edge weights, as each edge is weighted by the obstacle-free path distance that a robotic platform would take if that edge were included in the route.

We observe from the MAPF tests that, when using the associated pre-solve techniques on problems of the size considered in the simulation section, an empirical worst-case performance of 45.2 [s] is achieved. For our application, which is similar to the DARPA Subterranean Challenge, missions take between 30-60 minutes [46]. Thus, this empirically observed worst-case performance is sufficient for task allocation. Compared to the dynamic convoy topologies determined by the VRPMS-CC, simple fixed convoy topologies equal to the largest support value found at any location demonstrate significant increases in total distance traveled by all agents. In the generated missions, the total convoy travel distance can be decreased by as much as 56% (*Maze_32_32_6*). This large decrease in total traveled distance arises from the structure of the instance, as only two agents are needed at all locations except for a single location, which requires the entire team. This result indicates that the set and relative magnitude of different support values highly influence the resulting convoy behaviors. This observation was also tested on the *Hex* environments, as the locations are held fixed while the support values are changed. Noticeably, trials where

Table 3.2: Tests on MAPF benchmarks demonstrate VRPMS-CC reduces total distance traveled by all agents in the system

Name (Robots/Tasks)	Route Length % Decrease	Solve Time [s] (No presolve)	Solve Time [s] (Presolve)
Berlin_0_256 (5/24)	-10.2%	2900.5	15.8
Boston_0_256 (6/22)	-15.2%	1315.8	10.2
Paris_1_256 (6/21)	-20.3%	315.6	7.7
Maze_32_32_2 (6/46)	0.0%	2430.1	31.1
Maze_32_32_6 (6/50)	-56.2%	2543.1	45.2
Hex_CLUSTER (6/24)	-28.6%	338.9	36.2
Hex_MIX (6/24)	-15.7%	7184.0	6.3
Hex_ALT (6/24)	-4.3%	1142.4	4.2

similar support values are clustered (*HEX_CLUSTER*) yield lower solve times (without presolve) than randomized support values (*HEX_ALT*). Figure 3.4 is a visual representation of the determined set of routes plotted on both the "*Hex*" (Fig. 3.4a) and "*Berlin_1_256*" (Fig. 3.4b) environments.



(a) (b)

Figure 3.4: Two solutions depicted considered on a "*Hex*" environment (Fig. 3.4a) and a MAPF environment (Fig. 3.4b). Support values for each location (green) are indicated in red next to the location. The gold star indicates the depot node. Convoy routes that return to the depot are indicated with blue arrows, while convoy routes that progress to further nodes are in red.

3.6.3 MDVRP-SV Routing Heuristic

It is clear from Table 3.2 that additional solver accelerations could be introduced to improve solver performance. Noticeably, finding an initial feasible solution proves difficult in many of the MAPF trials. If an initial feasible solution could be found, we hypothesize that the data would reflect a decrease in solver time of a magnitude similar to the time it took to find a feasible solution. We introduce a decomposition-based multi-depot warm-start heuristic to achieve this decrease in solver time.

The intuition for this heuristic arises from the structure of the convoy constraints. The constraints provide routing restrictions that may be exploited to form a valid set of initial motions for each convoy detail. Specifically, the clustering rules described in Section 3.4 suggest that only a single convoy can visit one location from any other location. The consequence of this clustering rule is that convoys of agents can only be split from a larger convoy and not combined with smaller convoys. Thus, a convoy is largest when it leaves the depot, and all agents in the convoy are assigned tasks at locations with the highest support values first.

Our approach takes the form of a decomposition-based heuristic in which a set of multi-depot vehicle routing problems (MDVRP) are solved sequentially in order of descending support value. While it is clear that requiring the solution of an MDVRP inside the MDVRP-SV heuristic procedure indicates the heuristic is itself NP-hard, the advantage of posing the MDVRP-SV heuristic on the location space rather than the task space is that the location space is a smaller space by construction. Effectively, this solves for the motion of entire formations, rather than any particular platform. This process is detailed using Python syntax in Algorithm 2 on the following page.

Algorithm 2 MDVRP-SV Heuristic

1: **Define:** MDVRP-SV($\mathcal{N}, \mathcal{V}, d^-, d^+$)
2: $L \leftarrow \emptyset$, $SV \leftarrow \emptyset$ $D \leftarrow d^-$, $N \leftarrow \emptyset$
3: $SV \leftarrow$ sorted(set($\mathcal{N}$), reverse=True)
4: **for** $sv \in SV$ **do**
5: **for** $v_i \in \mathcal{V}$ **do**
6: **if** $v_i = sv$ **then**
7: $N \leftarrow \mathcal{N}[i]$
8: **end if**
9: **end for**
10: $L \leftarrow$ VRP(N, D) ▷ Perform VRP on sv locations
11: L.pop() ▷ Remove Depot Return
12: $D \leftarrow \emptyset$, $N \leftarrow \emptyset, D \leftarrow L[-1]$
13: **end for**
14: $L \leftarrow d^+$

Given the set of locations, $\mathcal{N}$, the support value set, $\mathcal{V}$, and the initial depot, the heuristic first searches through the support value list to extract all unique support values (Line 3). Next, the heuristic iterates through the unique support value list L_v to build partial routes at each support value level. This procedure is completed by first storing all locations with a particular support value (Line 7) in a set N. Following this, a procedure that solves the multi-depot Vehicle Routing Problem (VRP in Line 10) on set N starting from one or more depots stored in D. The result of solving the VRP is used to populate the sequence of locations for each formation containing sv platforms to visit, denoted L. The last element in L is removed (Line 11) to extract the last location visited by the formation with sv platforms (Line 12). After cycling through all support values, the final locations in L are connected to the return depots d^+.

The heuristic itself is designed for instances of the VRPMS-CC that are not time-constrained, as the expectation is that the convoy movement constraints and not the timing constraints drive the solution of the problem. For a problem with $|SV|$ unique support layers with a max of V nodes at any layer, the heuristic can be solved $|SV|$ times using both Yang's [47] multi-depot multiple traveling salesman transformation and the Held-Karp Algorithm [48] to yield an exponential worst case complexity of $\mathcal{O}(|SV| \cdot (V^2 2^V))$. Note that the heuristic still retains its exponential complexity in the number of nodes. Alternatively, to overcome scalability challenges, the MDVRP-SV could be solved via heuristic or metaheuristic methods. Note that the heuristic is myopic to each layer, as the solved MDVRP does not ensure that the last location in the previous layer is necessarily closer to the first location in the next layer. Consequently, symmetrically placed locations

Table 3.3: VRPMS-CC Benchmarks demonstrate formation-based routing heuristic decreases solver time

Map Name	Solver Time / % Change (No presolve) [s]	Solver Time / % Change (Presolve) [s]
Berlin_0_256	1381.9 / -52.4 %	21.2 / 33.5 %
Boston_0_256	1264.6 / -51.4 %	4.3 / -43.2 %
Paris_1_256	319.3 / 1.2 %	6.5 / -15.5 %
Hex_CLUSTER	157.8 / -53.5 %	2.7 / -92.5 %
Hex_MIX	758.5 / >-89.4 %	5.8 / -7.5 %
Hex_ALT	251.3 / -78.0 %	4.1 / -2.4 %

in the previous layer can adversely affect the optimality of the heuristic solution.

The influence of the heuristic on the solver performance was demonstrated on both the MAPF benchmark and the Hexagon (*Hex*) environments. All warm-starts were solved in less than 0.02 [s]. The first solution produced by Algorithm 2 (suboptimal or not) was used. The results of the rerun numerical studies are presented in Table 3.3. Notably, almost all maps exhibit a marked decrease in solver time for tests both with and without presolve. The only exceptions are the non-presolve *Paris_1_256* and presolved *Berlin_0_256* cases. Both warm-starts provided in these tests are highly suboptimal (<40%). The greatest improvement arises from the *Hex_12* case, which achieves an improvement in the solver time of more than 89%. This performance improvement directly results from the warm-start, which provides the optimal solution.

3.7 Four-platform Team Hardware Trials

A significant concern regarding the use of operations research models is whether the solved problem instance accurately represents the modeled robotic system(s) or environment. We illustrate the validity of this concern by considering how an unmarked environmental feature (a closed door) yields an *a priori* unknown infeasible set of routes. We then address this concern by demonstrating that an agent's onboard sensing can augment the representation in order to populate an instance of the VRPMS-CC.

3.7.1 Hardware Trial Environment Overview

All hardware trials were carried out in the indoor multi-hallway environment depicted in Fig. 3.5. Of note are two terrain features (doors) that, in certain configurations (closed),

are impassable to the agents. The existence of these features is not denoted in the initially provided representation (occupancy map). The locations of these doors are highlighted in Fig. 3.5b (south) and Fig. 3.5c (north). Clutter present in the environment was not dense enough to completely obstruct the motion of any agent.

Hardware Trial Robotic Agent Autonomy System

The team of four robotic agents (Fig. 3.2) was tasked with completing the convoy operation. The multi-agent team utilized in this chapter was first described in Chapter 2. Further details can be found in Appendix A and in [49]. The route of each agent was determined by solving an instance of VRPMS-CC on the operator interface. This solution was then provided to an agent-based convoy formation controller developed in Chapter 2, which formed the agents into a convoy and conducted the mission. During operations, each agent provides map and odometry information back to the base station over a wireless communication network.

3.7.2 Dynamic VRPMS-CC (DVRPMS-CC)

If the initial environmental representation is inaccurate, the initial route allocation may be suboptimal or infeasible. In this work, the inaccurate representation produces an inaccurate measure of c_{ij}^k. As this representation does not have an indication of the impassable feature, we utilize reactive "dynamic" vehicle routing problem (DVRP) techniques to address the potential infeasibility [42].

Our extension to a Dynamic VRPMS-CC (DVRPMS-CC) introduces a process that addresses two aspects common to DVRPs. The first aspect is that (3.4a) must be modified such that the platforms start from the locations they currently occupy rather than the original depot. Redefining the starting depot set as $\mathcal{D}^- = \{d^{k_1^-}, d^{k_2^-}, \ldots, d^{k_k^-}\}$, this modification yields a new route that starts from the agent's initial position ($d^{k_k^-}$ for platform k). The second modification involves updating the environment representation to ensure accurate measures of c_{ij}^k. The agent's SLAM system provides real-time mapping capabilities that may be compared to the occupancy grid used to generate the routes initially. In the presence of an impassible obstacle, the local planner does not yield a viable forward path for the system. This information is provided to the base station, which can then construct a new VRPMS-CC instance. The base station receives a map update from the agents over the communication network and then uses the updated map to recompute c_{ij}^k and populate the new VRPMS-CC instance. Solving the new VRPMS-CC instance yields an updated set of routes for the agents.

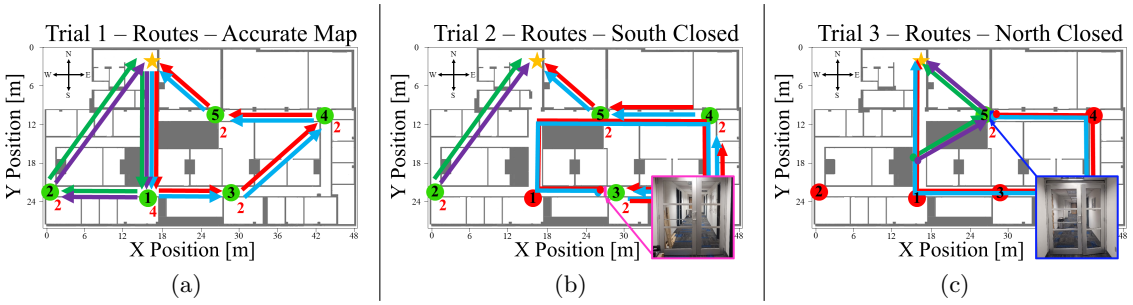


Figure 3.5: A diagram showing the final routes performed by each convoy in the hardware trials. Figure 3.5a depicts the convoy routes when no impassable terrain features are present. In Fig. 3.5b, the routes taken by the light blue and red agents are replanned once the doors (in pink) are observed and block forward progress. Fig. 3.5c demonstrates a similar scenario where the doors (in blue) are closed, requiring the green and purple agents to finish the last task.

3.7.3 Accurate Environment Representation

Consider an application of the VRPMS-CC framework to a patrol mission consisting of five locations (Fig. 3.5, in green), where one of the locations requires four robotic agents and all other locations require only two agents. Agents must visit all locations while minimizing the total distance traveled by all agents. The service time for each task is negligible, and the time window for task completion is the mission duration.

Given an accurate environmental representation, only a single instance of the VRPMS-CC must be solved on the base station. As shown in Fig. 3.5a, the agents first form a four-vehicle convoy formation that then splits into two two-vehicle convoy formations (green-purple and blue-red). The agents collectively traveled a total distance of 358.1 [m].

3.7.4 Inaccurate Environment Representation

If either the north or the south set of doors is closed in the test environment (Fig. 3.5b and 3.5c), the initial routes prove to be infeasible. They are either replanned (i.e., the same agents perform the task) or re-allocated (i.e., a different set of agents perform the task). In Trial 2, the southern doors obstruct the motion of the robotic agents toward Tasks 3-5, as shown in Fig. 3.5a. The DVRPMS-CC process outlined above produces the new set of routes shown in Fig. 3.5b. The total distance traveled by all vehicles increased by 46.7% as a result of the re-solve. In Trial 3, instead of the southern door being closed, the northern door was closed. The new routes re-allocate the task at Location 5 to the convoy that completed the task at Location 2, resulting in an increase in total distance traveled by all

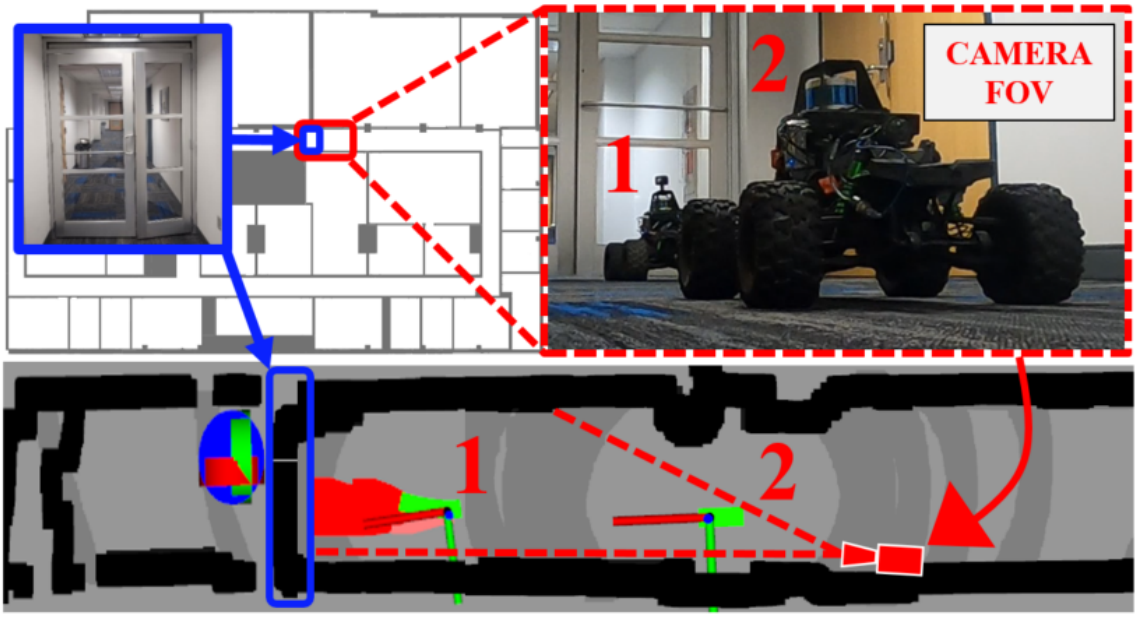


Figure 3.6: An image from Trial 3 depicting a convoy encountering the northern closed door (blue). The bottom figure shows the onboard occupancy map, reflecting that the front agent has no forward paths (red) due to the door.

vehicles of 48.3%. Note that the difference in total distances traveled for Trial 2 and Trial 3 is due to navigation around environmental clutter. An image from the trial is shown in Fig. 3.6

3.8 Conclusions

This VRPMS-CC utilizes the structure of routing problems, along with synchronicity constraints, to solve a multi-platform convoy coordination problem. The presented constraints enable a team of vehicles to coordinate their routes both spatially and temporally, allowing them to travel as a convoy. We utilize these synchronization constraints to define the VRPMS-CC, which is solved using off-the-shelf mixed integer programming solvers. Numerical studies and hardware results demonstrate the applicability of the model in both simulated and real-world cluttered environments, but the model inherits the challenges associated with scalability common to combinatorial search problems. We further contribute the MDVRP-SV heuristic, which improves solve times of off-the-shelf commercial solvers, and the DVRPMS-CC for online replanning when the initial environmental representation is inaccurate.

Although the results demonstrate the effective routing of a small team of robotic agents,

future work must incorporate additional realistic constraints and address solver performance limitations, particularly for large-scale problem instances. Investigating clustering rules that allow compositional teaming and constraints derived from the limitations of the communications system (e.g., range, drop-out) are critical priorities. In particular, the incorporation of information-transfer rendezvous points seems particularly relevant [50]. The performance of the heuristic implies that additional information regarding the problem structure (e.g., unique support value(s)) and the homogeneous nature of the agents (symmetry breaking) may lead to improved solve times of the VRPMS-CC. Furthermore, hybrid approaches that approximately solve the VRPMS-CC offline and use an alternative suboptimal method (e.g., reinforcement learning [40]) for online reactivity play to the strength of both approaches.

Part II

Convoy Formation-based Wireless Network Construction

Chapter 4

Automated Construction of Ad hoc Wireless Networks

Our approach concerning the use of a max-min tree to construct ad hoc wireless networks in an automated manner was first presented at the 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE).

Citation:

C. Noren, B. Vundurthy, N. Bagree, M. Travers, "A Max-min Tree Approach to the Automated Construction of Ad hoc Wireless Networks in Unknown Environments," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, Los Angeles, USA, August 2025

Distribution:

This work was in part supported by OUSD/R&E (The Under Secretary of Defense-Research and Engineering), National Defense Education Program (NDEP) / BA-1, Basic Research. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

4.1 Overview

RELIABLE communication networks are essential for the remote operation of automated teams of robotic agents. For unknown (no prior map) and communications-deprived (no existing communication infrastructure) environments, the robotic agents must construct the network as they move through the terrain. These constructed networks can then enable agents to transfer essential data to one another and share decision-making assets within the team. For our work in Chapter 3, this included updating the central optimizer with environmental data and passing re-solved plans back to the agents. We see that to support our bi-level task allocation and task execution framework, we need to ensure communications between the agents. We highlight this requirement in Figure 4.1.

We present a novel method for automated network construction tailored for mobile robotic teams that require communication with a central base station. Our key innovation is the introduction of a maximin spanning tree structure, which guarantees a minimum level of signal strength between nodes. By directly optimizing node placement based on signal-based metrics, rather than relying on geometric surrogates such as distance and visibility, we also achieve significant decreases in agent utilization while maintaining coverage for the traversed area. By using the robotic agents themselves as mobile repeaters in a communication network, each robotic agent can be individually assigned to prioritize network connectivity during critical operations. Numerical simulations on standard multi-agent pathfinding benchmarks demonstrate a reduction of up to 36% in the number of required

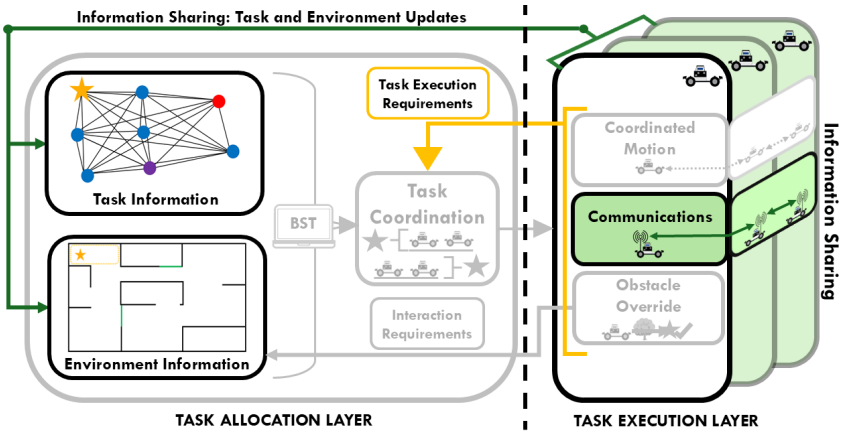


Figure 4.1: Communicating information between agents and the central solver is an essential capability required for the DVRPMS-CC. In this chapter, we focus on developing communication network construction behaviors to ensure effective communication in communication-deprived environments.

nodes compared to existing techniques. Furthermore, our method guarantees robust network connectivity in dynamic environments, outperforming strongest-neighbor approaches that are vulnerable to link disruptions. Lastly, hardware tests confirm the robustness of our method in challenging scenarios encountered in real-world deployments.

4.2 Introduction

Many complex missions in dangerous environments, e.g., wildfire firefighting, battlefield triage, and search-and-rescue, will be revolutionized by the use of coordinated multi-agent robotic systems. Currently, high-performing multi-agent coordination relies on robotic agents that reliably exchange information with one another and with other decision-makers. However, many environments lack the communication infrastructure to support agent communications. Recent efforts have enabled robotic systems to construct this network during operations. Yet, ensuring a minimum quality of service across participants in the multi-agent system during network construction remains challenging. We propose that by augmenting network construction techniques with communication-strength-aware tree search on the network, a quality-assured ad hoc robotic network topology can be established to support multi-robot operations in hazardous environments.

Abandoned buildings or subterranean caverns are examples of potentially hazardous environments with little to no existing communication infrastructure. Exploration of these environments was the focus of the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge (SubT) [46]. During exploration, vehicles were often expected to maintain contact with a central base station to communicate information back to a set of human operators. Multiple SubT performer teams approached this communication challenge by creating a mobile ad hoc wireless mesh network (MANET) by deploying "relay nodes" throughout the environment they traversed. These relay nodes can establish communication throughout the environment; however, node placement strategies may result in variable communication quality or a high number of dropped nodes.

We introduce an automated MANET construction technique for a small team of agents that aims to minimize the number of dropped nodes while ensuring quality communications (see Fig. 4.2). As the network grows, whether through the deployment of additional nodes by different robots or simply due to extended operation, multiple communication paths to the base station emerge. To ensure robust connectivity and maximize the minimum link quality across the diverse paths, we present a complete algorithm to compute a maximin metric-based spanning tree, which explicitly quantifies the connection quality of the MANET between an agent (or formation) and a base station. To our knowledge, no previous robotic

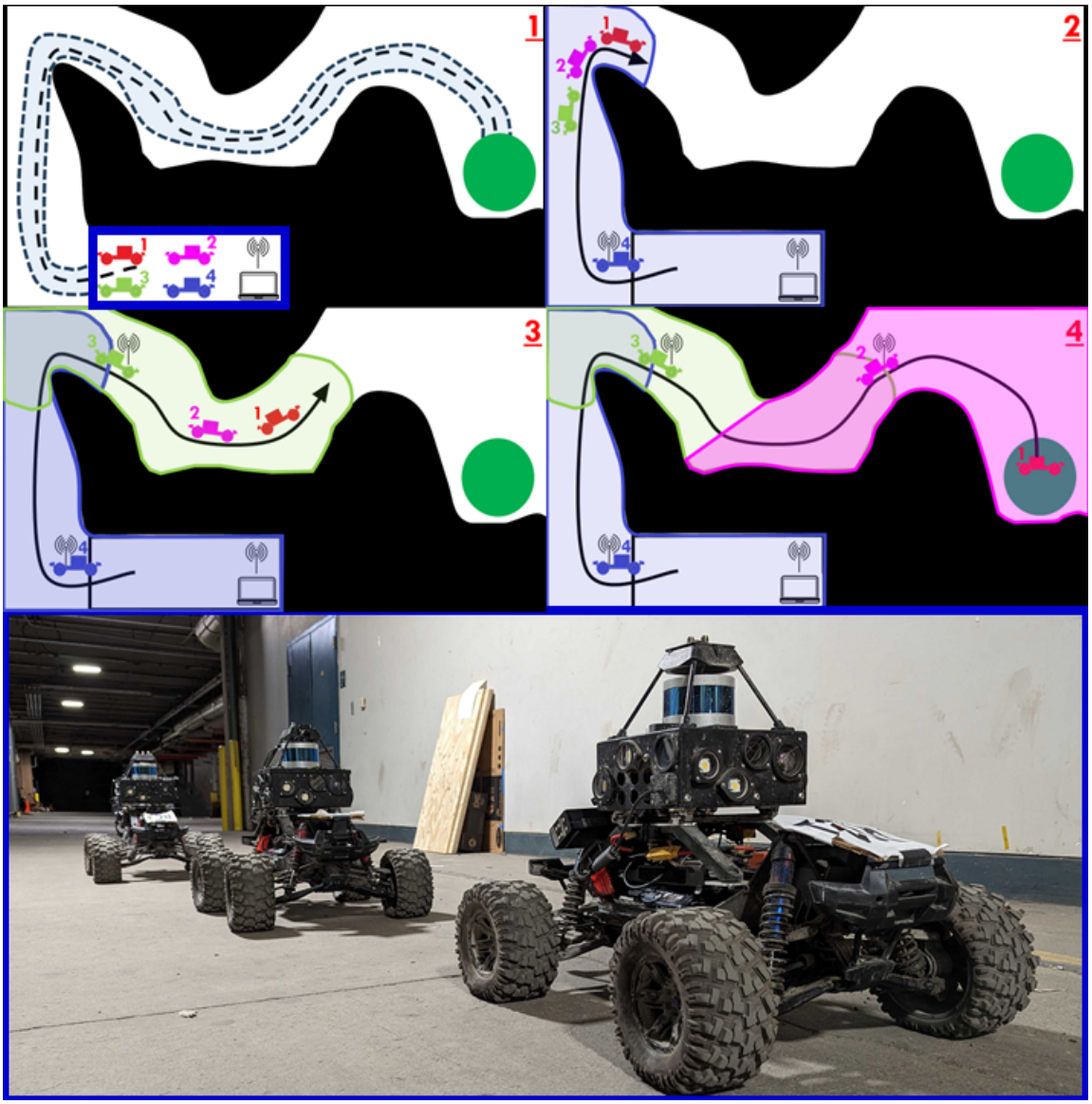


Figure 4.2: A four-agent robotic convoy builds a communications network in a communications-deprived environment starting from a base station (black computer) and ending at a goal location (green circle). To ensure all agents remain in communication with the base station, the agents leave the convoy to become communications nodes as the convoy reaches the communications boundary. As the convoy leaves the communications area supported by the base station (black), Agent 4 leaves the convoy to act as a communications node (pane 2), extending the communications-accessible area further. In subsequent panes, Agent 3 and Agent 2 similarly leave the convoy, enabling Agent 1 to reach the goal location with communication.

MANET construction technique enables this capability.

Furthermore, the maximin tree provides a reliable method for determining the connectivity of any node to the base station, even when one or more intermediate nodes stop responding. This demonstrates its resilience to node failures, a key aspect of dynamic network environments. We demonstrate the effectiveness of our maximin spanning tree in improving communication quality through simulations on standard Multi-Agent Path Finding benchmark maps [45] and physical experiments involving formations of robotic agents. Therefore, to address the challenges of reliable MANET construction for teams of mobile robotic agents in dynamic environments, we contribute:

1. a quality-assured maximin tree-based communication construction algorithm for MANET generation;
2. a communication relay deployment behavior for network construction in *a priori* unknown environments.

4.3 Related Works

Given the lack of available communications infrastructure in the DARPA SubT Challenge, multiple performer teams employed network construction techniques to ensure communication. These techniques would construct a "communication backbone" that supports robotic operations by "dropping" communication nodes, thereby extending the effective coverage of a wireless communication network [51]. For Team CERBERUS, a human operator was primarily responsible for determining the drop points [52]. For Team Explorer, this "dropping" behavior was primarily controlled by distance-based limits and line-of-sight (LOS) requirements between nodes [20]. Given their simplicity, geometric communications models are commonly used in robotic systems, where physical distance and LOS take precedence over communications-based signal strength metrics [53]. However, SubT Teams also considered communication-based metrics, including a Radio Signal Strength Indicator (RSSI) threshold, as one of the determining factors for deploying both mobile and stationary communication nodes [54, 55]. As an alternative to RSSI, Team CoSTAR utilized the signal-to-noise ratio (SNR) in conjunction with other environmental and communication factors [56].

The aforementioned approaches and metrics investigated in the DARPA SubT Challenge broadly capture aspects of the sensor coverage problem. For known environments, computational geometry approaches that frame the sensor coverage problem as a variant of the Art Gallery Problem (AGP) have proven extremely successful [55]. To address the computational complexity of the AGP, suboptimal polynomial-time approximations, such

as polygonal decomposition or partitioning, are frequently employed [57]. Many variants of the AGP place additional restrictions on the coverage model used in the AGP, including limited range [57], range fading [58], or k-visibility through boundaries (i.e., the problem of k-transmitters) [59]. In accordance with the AGP, these approaches often seek to provide coverage of polygonal areas and may rely on geometry-based communication models to derive the optimal sensor placements. Such models lie in contrast to a recent model presented by [53], which specifies that only certain areas of the state space must be made "communication-accessible." Such an approach provides the unique ability to represent realistic (i.e., non-polygonal) environments that robotic systems operate in, and often more closely aligns with the robotic operations (i.e., non-coverage problems).

Finally, the relay placement problem does not exist far outside of previous work in the communications-aware motion planning space. Many of these works focus on the development of control or planning algorithms that place requirements on maintaining specific distances (e.g., coverage area, visibility) or on metrics of network connectivity [60, 61]. These works are often structured around a fixed network topology and check for connectivity quality or reachability using a tree-based analysis. However, these works do not address the crucial aspect of constructing extensible network topologies that guarantee a minimum communication strength criterion [62].

4.4 Problem Definition

In this chapter, we consider a variant of the relay placement problem detailed in [53]. The objective of the relay placement problem is to determine the minimum set of communication relay locations required to form a valid network topology that covers a specified area of interest within the environment. In order to solve the problem, we require: 1) a defined area of interest, 2) a coverage model, which describes the conditions for communications coverage, and 3) a network model, which states the set of constraints required to form a valid network topology. The area of interest (denoted by $\mathcal{I}$) in the relay placement model represents an area that is to be made "communications-accessible" to the agents trying to traverse or explore the environment [53].

Consider a planar environment $\mathcal{W} \subset \mathbb{R}^2$ that is divided into object-free space ($\mathcal{F}$) and object-occupied space. All elements of the environment can be delineated into one of the two spaces through the use of an occupancy function $F : \mathcal{W} \to \{0, 1\}$. Here, "0" denotes that a spatial element is contained in the object-free space. Using this categorization, we can define $\mathcal{F} = \{x \in \mathcal{W} : F(x) = 0\}$. Our defined area of interest, $\mathcal{I}$, is a subset of the object-free space $\mathcal{F}$. Specifically, it is the region within which robotic agents will traverse for

mission objectives, necessitating uninterrupted communication with the base station. For convenience, we define another indicator function $I : \mathcal{F} \to \{0, 1\}$, which maps elements of the object-free space to the area of interest (a value of "1" indicates an area must be covered). This indicator function enables us to define the area of interest $\mathcal{I} = \{x \in \mathcal{F} : I(x) = 1\}$, such as the light blue region around the planned path in pane 1 of Fig. 4.2. Note that $\mathcal{W}, \mathcal{F}, \mathcal{I}$ are connected spaces.

To maintain communication coverage, we consider a communication criterion inspired by our field experiences in the DARPA SubT Challenge [20]. Unlike [53], which considers an ℓ_2-boundedness and a visibility constraint between a pair of relays to maintain connectivity, our communication criterion depends directly on the measured communications strength between a pair of relays. For a total of N deployable relays, the relay set $\mathcal{R} = \{r_1, \ldots, r_N\}$ is defined as a set of 3-tuples: $r_i = (i, x_i, y_i) \in \mathbb{N} \times \mathbb{R} \times \mathbb{R}$, where i is the relay index and $p_i = (x_i, y_i) \in \mathbb{R}^2$ is the relay position. To measure the communication strength between relays in the relay set, we define a function, $\mathrm{C}_{\mathrm{val}}(r_i, r_j) \to \mathbb{R}^+$, that returns a measure of communication strength (e.g., SNR, RSSI) between nodes $r_i \in \mathcal{R}$, and $r_j \in \mathcal{R}$. If $\mathrm{C}_{\mathrm{val}}(r_i, r_j) = 0$, then relays r_i and r_j are not connected.

We consider a pair of relays r_i and r_j to be mutually covered by each other if the observed communication strength, $\mathrm{C}_{\mathrm{val}}(r_i, r_j)$, is greater than a threshold value: $\mathrm{c}_{\mathrm{thresh}} \in \mathbb{R}^+$: $(\mathrm{C}_{\mathrm{val}}(r_i, r_j) > \mathrm{c}_{\mathrm{thresh}})$. For convenience, we extend our definition of coverage to include positional arguments as well: $\mathrm{C}_{\mathrm{val}}(p, r_j) \to \mathbb{R}^+$, which captures the signal strength between a (potentially fictitious) relay placed at location $p \in \mathbb{R}^2$ and a relay $r_j \in \mathcal{R}$.

We then construct a network from the relays in the relay set $\mathcal{R}$ by placing the relays at different locations in the object-free space. Our goal is to create a valid network topology $\mathcal{N} = \{r_1, \ldots, r_n\},\ n \leq N$ to cover space $\mathcal{I}$. Space $\mathcal{I}$ is covered if $\forall\, x \in \mathcal{I}, \exists\ r \in \mathcal{N}$: $\mathrm{C}_{\mathrm{val}}(x, r) > \mathrm{c}_{\mathrm{thresh}}$. Finally, we define a valid network topology as one that is "connected." Simply put, a network is connected if, for all pairs of relays $r_i \in \mathcal{N}, r_j \in \mathcal{N}$, there exists a progression of relays starting from r_i and ending at r_j such that relays are sequentially mutually covered.

In this chapter, we are interested in enabling communications between a starting position, x_s, and a goal location, x_g, in the environment. We define the optimal planar planning problem over a state space $\mathcal{W}$ with permissible states $\mathcal{F}$, requiring $x_s \in \mathcal{F}$ and $x_g \in \mathcal{F}$. We assume that there is at least one relay at x_s. We define $\sigma : [0, 1] \to \mathcal{W}$ as a sequence of states (a path) from the set of paths Σ. The optimal planning problem is therefore to find a path σ^* that minimizes a cost function $s : \Sigma \to \mathbb{R}^+$ and connects x_s to x_g. We define the condition for optimality in a general manner, as our approach is not preconditioned on any specific measure. We define the optimal path as follows:

$$\sigma^* = \arg \min_{\sigma \in \Sigma} \{s(\sigma) |\ \sigma(0) = x_s, \sigma(1) = x_g, \forall t \in [0,1], \sigma(t) \in \mathcal{F}\}. \tag{4.1}$$

In a known environment, where a robotic team aims to travel between two points while maintaining continuous communication with the base station, the area of interest $\mathcal{I}$ can be defined as the optimal path σ^* connecting these points. While selecting $\mathcal{I} = \sigma^*$ does not necessarily minimize the number of communication nodes required, it guarantees connectivity between the base station and the goal position. Minimizing the number of communication nodes in a known environment is a variant of the minimum covering set problem, which is NP-hard and falls outside the scope of this thesis [63].

In this chapter, we do not assume any prior knowledge of the environment's structure, specifically the decomposition of the workspace $\mathcal{W}$ into free and occupied space. Thus, given a motion planning policy $\pi(x_s)$ that yields a feasible path σ between x_s and x_g, we propose a network construction algorithm that procedurally covers the area of interest $\mathcal{I}$. By employing a placement strategy that ensures a connected network topology, we enable the robotic agents to traverse the environment while maintaining continuous communication with the relay at x_s.

4.5 Methodology Overview

A communication network topology can be modeled as a simple weighted undirected graph $G = (V, E)$. In this model, the vertices V of this graph represent physical assets participating in a network topology $\mathcal{N}$, including deployed relays and any communications-enabled agents. These assets are collectively referred to as "communication nodes." While we define any given communication node as $v_n \in V$, we split the set of vertices V into two unique sets: 1) the set of mobile "non-deployed" agents P and 2) the set of static "deployed" nodes S (i.e., $V = P \cup S$). For a team of p robotic agents $P = \{p_1, p_2, \dots p_p\}$ and N available nodes in the communication network, we define $\mathcal{J}_N = \{1, \dots, N\}$. Here, $n \in \mathcal{J}_N$ represents a specific communication node index. We also overload the definition of $v_n \in V$ to be equivalent to the physical location of $p_n \in \mathbb{R}^2$. The communication graph is initialized with v_1 as the only starting node, which often represents a central base station (see Fig. 4.2).

The edges of the communication graph have an associated positive weight, $e(v_i, v_j) \rightarrow \mathbb{R}^+$, $v_i, v_j \in V$, which represents a communication link between the vertices. For example, in Team Explorer's original approach and Zoula and Faigl [53], a positive distance function $\mathrm{d}(v_i, v_j) : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$, which measured the physical ℓ_2 distance between two communication nodes in the graph, was utilized for $e(\cdot)$. Although physical distance may be used as a surrogate measure of communication strength, recent advances in communication technology

have enabled the direct use of communication strength metrics (e.g., RSSI, SNR) for relay placement. For ease of notation, we denote all communication strength metrics between two vertices (v_i, v_j) using our communication coverage model notation: $\mathrm{C_{val}}(v_i, v_j) \rightarrow \mathbb{R}^+, v_i, v_j \in V$. Note that in this chapter, as $e(\cdot)$ may take the form of $d(\cdot)$ or $\mathrm{C_{val}}(\cdot)$, the particular use will be indicated as required.

4.5.1 Maximin Communications Graph Spanning Tree

To ensure that communications are maintained between the base station and all assets, we design a reactive node placement behavior that deploys a communications node in response to an imminent loss of communication coverage. To ensure the connectivity of all assets $v_n \in V$ to the base station (v_1), we require the existence of at least one nodal path in the communications graph G that starts at the base station v_1 and ends at every asset v_n where all relays are progressively sequentially mutually covered. By defining a path through G between the base station v_1 and any vertex $v_n \in V$ as $\pi(v_1, v_n) = v_1, \ldots, v, \ldots, v_n,\ v \in V, v_1 \neq v_n$, we impose the condition $\mathrm{C_{val}}(v_i, v_{i+1}) \geq \mathrm{c_{thresh}}\ \forall\, (v_i, v_{i+1}) \in \pi(v_1, v_n),\ \forall\ v_n \in V$ where the pairing (v_i, v_{i+1}) represents every pair in the sequence of nodes defined by the path π. This condition generates a tree structure T on G, i.e., $T \subseteq G$, with vertex v_1 as the root vertex of the tree. Given this structure, we choose to simplify the path notation between node v_1 and v_n into a single argument (i.e., $\pi(v_1, v_n) = \pi(v_n)$).

To enforce the single-path communications coverage constraint to all mobile agents, we pose T as a minimum spanning tree where the edge weights capture the strongest "weakest" link (i.e., the widest-path) on a path between the root vertex and a non-root vertex in the tree. We refer to this metric as the "maximin communications metric." For each non-root vertex, v_n, connected by a path $\pi(v_n)$ to the root vertex v_1, a value $\mathrm{C_{met}}(v_1, v_n) \in \mathbb{R}^+$ is calculated using Algorithm 3. This is done to find the maximum minimum edge value $\mathrm{C_{val}}(\cdot)$ for all nodal paths between v_1 and v_n in the communication graph G. By definition, the spanning tree T spans all vertices included in the communications graph. Thus, by ensuring $\mathrm{C_{met}}(v_1, v_n) \geq \mathrm{c_{thresh}},\ v_n \in V$, we ensure that there is at least one path between v_1 and v_n such that the minimum edge weight is observed for all edges in the tree.

We utilize several helper data structures to construct the maximin tree. The first structure is a real-valued n-tuple, $\mathrm{CV} \in \mathbb{R}^N$, which represents the minimum communication value experienced between the n^{th} communication node and the root node v_1. Next, we define another n-tuple, $\mathrm{SPT} \in \{0, 1\}^N$, which tracks the inclusion of vertices in the spanning tree. Finally, $(\mathrm{PATH}, \mathrm{TEMP}) \in \mathbb{Z}^{+,N}$ tracks the structure of the tree. All tuple indices correspond to node indices, with index 1 representing the central base station.

We additionally introduce a supporting routine, $\mathrm{MCV}(\mathrm{CV}, \mathrm{SPT}, V)$ that parses CV

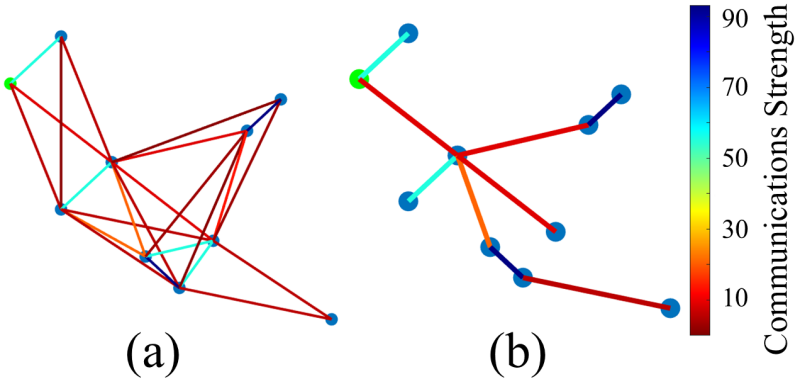


Figure 4.3: A communications graph (G) on the left and its corresponding maximin tree (T) on the right. The edge colors demonstrate the relative strength of the inter-vertex communications connection.

for the maximum value of a node not already in the tree and returns that node's index or -1 if all nodes are already included in the tree. Algorithm 3 exists in the family of single-source shortest path algorithms (making it a complete algorithm). With a naive array implementation, the time complexity of the presented algorithm is $O(|V|^2)$. This logic takes the set V and root node v_1 as input where $1 \in \mathcal{J}_N$, determines the maximin tree for the graph, and then publishes a message identifying the vertices with a $\mathrm{C_{met}}(\cdot)$ below $\mathrm{c_{thresh}}$. For convenience, define: $\mathrm{MIN}(\cdot) = \mathrm{CV}[y] \leftarrow \min(\mathrm{CV}[x], \mathrm{C_{val}}(v_x, v_y))$.

By running Algorithm 3, the value of $\mathrm{C_{val}}(\cdot)$ may be monitored to ensure communication capabilities. An example of the maximin spanning communications tree for a communications graph is shown in Fig. 4.3. In Fig. 4.3a, the vertices in the communications graph are demonstrated in both green (v_1) and blue ($v \in (V \setminus \{v_1\})$), with edges demonstrating connection strengths greater than $\mathrm{c_{thresh}}$. The edge colors reflect varying communications strengths, from low strength in red to high strength in blue. Using Algorithm 3, a spanning tree structure can be imposed on the graph presented in Fig. 4.3a, which is shown in Fig. 4.3b.

4.5.2 Communication Relay Deployment Behavior

In this chapter, we limit each agent of the multi-agent team to act as at most one communication node (as opposed to the carrier agents in [20]). Once the maximin communications spanning tree is found, the communication relay deployment behavior in Algorithm 4 periodically checks to ensure that no agent has a $\mathrm{C_{met}}$ value below $\mathrm{c_{thresh}}$. Note that if multiple agents (e.g., $i, j \in P$) have a $\mathrm{C_{met}}$ value below $\mathrm{c_{thresh}}$, and if $\mathrm{C_{val}}(i, j) > \mathrm{c_{thresh}}$, then before both agents deploy nodes, only the agent with the lowest ID (e.g., $i < j$ implies

Algorithm 3 Maximin Communication Spanning Tree

Define MCST(v_1, V)
Input v_1, V ▷ Index 1 is the assumed root
Output CV
Require: CV $= \{0\}^N$, CV[1] $= \infty$, SPT $= \{0\}^N$
Require: PATH $= \{1\}^N$, TEMP $= \{1\}^N$
for $i \in \mathcal{J}_N$ **do**
  if $\sum_{j \in \mathcal{J}_N, i \neq j} \mathrm{C_{val}}(v_i, v_j) = 0$ **then**
    SPT$[i] = 1$, CV$[i] = 0$, PATH$[i] = i$
  else ▷ if not disconnected
    x = MCV(CV, SPT, V)
    if $x \neq -1$ **then** ▷ i.e., still have nodes in tree
      SPT$[x] = 1$, PATH$[i]$ = TEMP$[i]$
      for $y \in \mathcal{J}_N$ **do**
        if ($\mathrm{C_{val}}(v_x, v_y) > 0$, SPT$[y] = 0$, and
          CV$[y] <$ MIN$(\cdot)$) **then**
          CV$[y]$ = MIN$(\cdot)$, TEMP$[y] = x$
        end if
      end for
    else ▷ if all nodes already in the tree
      for $y \in \mathcal{J}_N$ **do**
        if SPT$[y] = 0$ **then**
          SPT$[y] = 1$, CV$[y] = 0$,
          PATH$[y] = y$ ▷ Set self as parent
        end if
      end for
    end if
  end if
end for

Algorithm 4 Communication Relay Deployment Behavior

Input P, G, v_1, σ, t

while $p_p \neq \sigma(1)\ \forall\ p \in P$ **do** ▷ while no agent at x_g
  $\mathrm{C_{met}} \leftarrow \mathrm{MCST}(v_1, V(G))$
  for $k \in P$ **do**
    if $\mathrm{C_{met}}[k] < \mathrm{c_{thresh}}$ **then**
      agent k deploys a communication relay
    end if
  end for
  Move along σ ▷ for t seconds
end while

agent i) deploys a relay. This ensures that multiple agents do not redundantly deploy nodes at the same location.

Thus, the experienced automated behavior is that an agent stops to become a stationary communication repeater (relay) if its experienced $\mathrm{c_{thresh}}$ value falls too low. In practice, the "true" minimum value of $\mathrm{c_{thresh}}$ is buffered by some amount, $\delta \in \mathbb{R}^+$, to conservatively approximate a communications boundary. This ensures that the system does not experience an unexpected dropout or cause oscillatory behaviors near the communications boundary (i.e., $\mathrm{c'_{thresh}} = \mathrm{c_{thresh}} + \delta$). The magnitude of δ is highly dependent on the system hardware and the rate t at which Algorithm 4 is checked.

4.6 Comparative Simulations

To evaluate the performance of our proposed maximin communication-metric tree construction technique, we conducted simulations within Multi-Agent Path Finding (MAPF) benchmark environments [45]. These benchmarks are employed solely to generate realistic communication strength layouts, enabling the simulation of communication graphs without relying on prior environmental knowledge. Crucially, from a communication perspective, we are still operating within unknown environments. We begin by comparing our maximin approach with the network construction method used by Team Explorer, demonstrating that our method yields a reduction in the number of deployed nodes. Furthermore, our simulations indicate that the maximin tree's ability to condense multiple nodal paths into a single representative value provides a more informed assessment than a myopic approach.

4.6.1 Simulation Environment

In cluttered environments, the effectiveness of communication networks is significantly impacted by distance and obstacles. To model this, we adopt an inverse square law for signal attenuation, approximating signal strength as inversely proportional to the square of the distance with an additional linear decline through the obstacles. The communication model used in this simulation section is given as

$$\mathrm{C}_{\mathrm{val}}(r_1, r_2) = c_1 \cdot \frac{1}{\overline{r_1 r_2}^2} - c_2 \cdot d_{obst}.$$

Note that this model is adopted solely for simulation purposes and is not utilized to pre-process the environment for communication node placement. The algorithm's performance with a real communications system will be demonstrated in the subsequent hardware trials section. The simulations in this section assume a value of $c_1 = 100$ for obstacle-free space and an additional constant $c_2 = 20$ when considering obstacles. For example, a robot 2 units away from the source with an obstacle that is 0.5 units thick in between experiences a signal strength of $25 - 10 = 15$ units.

4.6.2 Visibility vs. Communications-Metric Graph Construction

This section compares the performance of two communication network construction approaches: the visibility-based method used by Team Explorer in SubT [20] (Algorithm 5) and our maximin communications-metric-based construction approach (Algorithm 4). Team Explorer's approach adds nodes to the graph G depending on their relative distance and visibility to previously deployed nodes, as detailed in Algorithm 5. To check visibility, we define a procedure, $\mathrm{LOS}(v, p), v \in V$, which returns "True" if p is within the line of sight of any other node in V and otherwise returns "False." Furthermore, two hyperparameters: $\mathrm{d}_{\mathrm{LOS}} \in \mathbb{R}^+$ and $\mathrm{d}_{\mathrm{NLOS}} \in \mathbb{R}^+$, were defined to capture the maximum allowable distance (e.g., in an ℓ_2 sense) for a node to be placed with respect to any previous node in G. The first parameter, $\mathrm{d}_{\mathrm{LOS}}$, represents the furthest allowable distance an agent p is allowed to be from any other vertex $v \in V(G)$ while $\mathrm{LOS}(v, p)$ is "True." Similarly, $\mathrm{d}_{\mathrm{NLOS}}$, represents the furthest allowable distance an agent p is allowed to be from any other vertex $v \in V(G)$ if $\mathrm{LOS}(v, p)$ is "False." Team Explorer's procedure for adding nodes to the communication network topology is presented in Algorithm 5.

We evaluate their effectiveness by navigating a robot from a random start to a goal location. Along the path, communication nodes are deployed whenever the signal strength falls below a threshold ($\mathrm{c}_{\mathrm{thresh}}$) of 10. A key difference lies in how signal attenuation is modeled. To ensure connectivity in Algorithm 5, we model a visibility-based approach that

Algorithm 5 Explorer Communication Graph Construction

Input N, P, $\mathrm{d}_{\mathrm{LOS}}$, $\mathrm{d}_{\mathrm{NLOS}}$
Require: $N \geq 1$, $\{v_1\} = S$
while $n \leq N$ **do** ▷ Exit if run out of nodes
  for $p \in P$ **do**
    $s = \arg\min \mathrm{d}(s_i, p), s_i \in S$ ▷ Find closest node
    if LOS(s, p), and $\mathrm{d}(s, p) > \mathrm{d}_{\mathrm{LOS}}$ **then**
      $S = S \cup p$, $P = P \setminus \{p\}$, $n = n + 1$
      "Establish Node" ▷ Drop Node for SubT
    else if $\mathrm{d}(s, p) > \mathrm{d}_{\mathrm{NLOS}}$ **then**
      $S = S \cup p$, $P = P \setminus \{p\}$, $n = n + 1$
      "Establish Node" ▷ Drop Node for SubT
    end if
  end for
end while

assumes an immediate and complete signal loss (infinite attenuation, $\mathrm{d}_{\mathrm{NLOS}} = 0$) when the line of sight between the communications nodes is broken.

To evaluate the performance difference, we conducted extensive simulations across a diverse set of maps from the MAPF benchmark dataset. For each map, we randomly generated 100 different start and goal locations for the robot, ensuring a comprehensive assessment across varying environmental configurations and path requirements. These experiments measure the number of nodes required to maintain communication between the start and goal points, with fewer nodes indicating a more efficient approach. The results are presented via Fig. 4.4 where the node construction approach from Algorithm 4 (in blue) consistently outperforms (has fewer nodes) the approach from Algorithm 5 (in red). The largest decrease in required nodes was observed in the *maze-128-128-1* environment, where the average reduction in deployed nodes across all trials was 36%.

4.6.3 Efficacy of Maximin Communication Spanning Tree

Algorithm 3 optimizes the maximin communication metric across nodal paths, ensuring the weakest link to the base station exceeds a threshold for higher network quality. While intuitively beneficial for practical robotic applications, we found no prior architectures offering comparable network quality guarantees. To evaluate the efficacy of our algorithm, we introduce a baseline approach: the strongest neighbor communication metric (Algorithm 6). This method assumes an ideal, disruption-free network, where connectivity to the strongest neighbor directly implies connectivity to the base station. Such a simplified strongest-neighbor strategy reflects common practices observed in the SubT Challenge, providing a

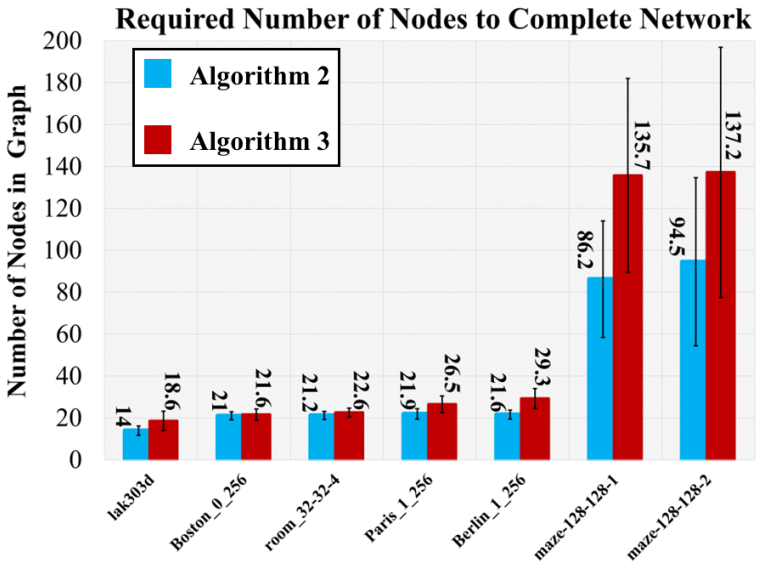


Figure 4.4: A summary of network construction tests demonstrating the effect of communication metrics on network growth. Both metrics demonstrate the capability to construct the network through all maps, but dropping line-of-sight constraints and utilizing communication-based metrics decreases total node usage.

comparative benchmark for our maximin algorithm.

To illustrate the algorithmic differences, consider Fig. 4.5a, which depicts already deployed nodes (blue) and a base station (pink). We examine a point (blue star, which represents v^*) along the path between the red and green dots, which in turn represent x_s and x_g, respectively. In this instance, Algorithm 3 computes $\mathrm{C}_{\mathrm{met}}(\cdot) > \mathrm{c}_{\mathrm{thresh}}$ indicating sufficient connectivity for v^* to the base station. In contrast to Algorithm 3's use of $\mathrm{C}_{\mathrm{met}}(\cdot)$ to determine connectivity, all intermediate links exceed the required threshold, correctly indicating sufficient connectivity for v^*. In contrast, Algorithm 6 relies on the strongest neighbor link (e.g., node v_a). In Fig. 4.5a, this also results in a correct connectivity assessment, which we denote as a true positive.

However, Fig. 4.5b shows a potential failure scenario. Here, three nodes, including v_a, experience intermittent connection issues, rendering them nonfunctional. Consequently, the star node's strongest link shifts to v_b, which still exceeds the threshold. Algorithm 6, in this case, incorrectly infers base station connectivity via v_b, resulting in a false positive when compared to the true connectivity determined by Algorithm 3. This example highlights the vulnerability of the strongest neighbor approach to link disruptions, which the maximin tree is robust against.

To rigorously evaluate the performance of Algorithm 3 and Algorithm 6, we conducted

Algorithm 6 Strongest Connection Metric

Input v^*, G, c_{thresh}
Output IfConnected
Require: MAXCV = 0
1: add v^* to G
2: MAXCV = $\max(C_{val}(v^*, v_i),\ \forall\ i \in \mathcal{J}_N)$
3: **if** MAXCV > c_{thresh} **then**
4: IfConnected = True
5: **else**
6: IfConnected = False
7: **end if**

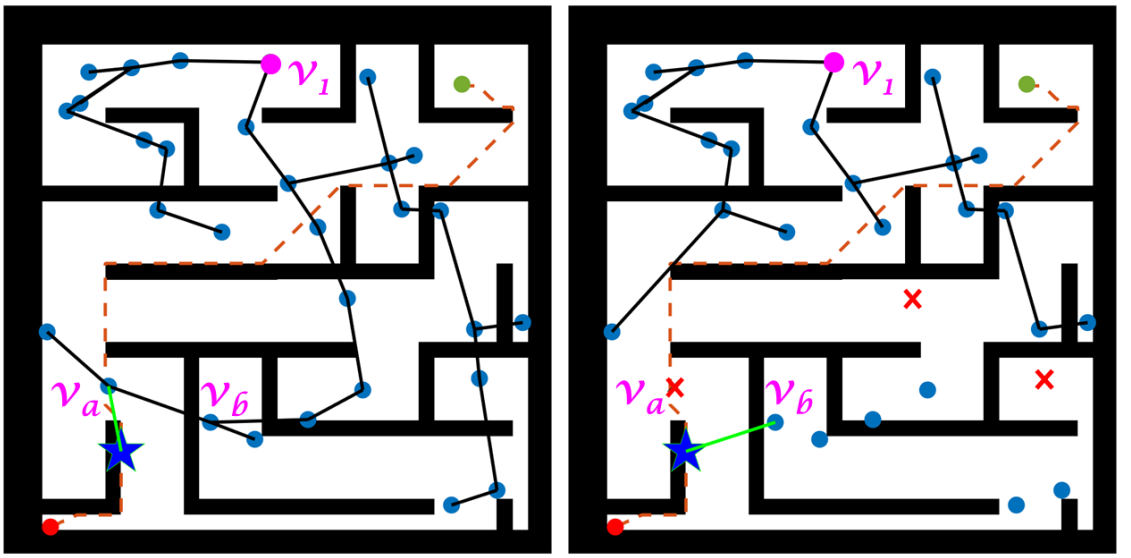


Figure 4.5: Comparison of Algorithm 3 and Algorithm 6.

an extensive study across multiple MAPF benchmark maps, each with randomized node configurations. Specifically, we generated 100 random pairs of start and goal locations (x_s and x_g) within each map and tested both algorithms at discrete points between them. Each environment is seeded with 30 relays that form a valid, connected network. We simulated network disruptions by randomly disconnecting 3 of the 30 nodes. We also randomly select a base node from the remaining set of non-disconnected relays. This methodology allowed us to assess the effectiveness of the algorithms across a diverse range of dynamic network conditions. We quantified performance by calculating the rate of false positives produced by Algorithm 6 relative to the total positives identified by Algorithm 3. This metric represents

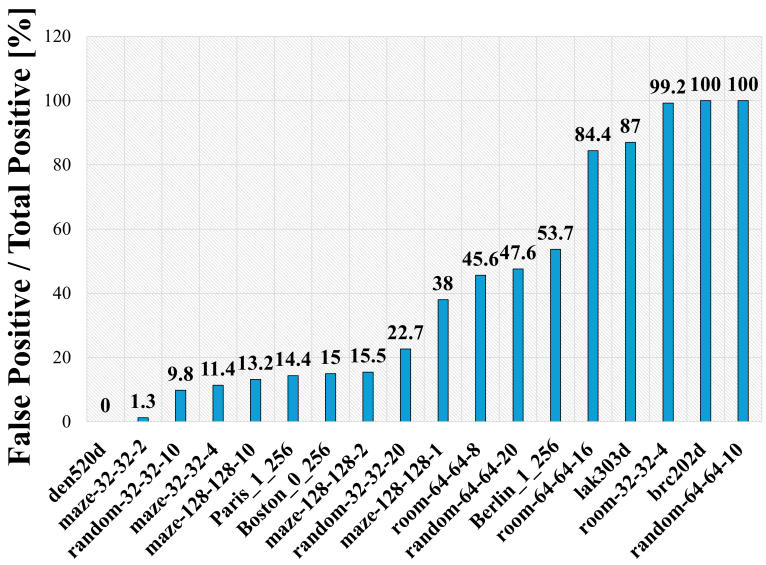


Figure 4.6: Multiple MAPF environments demonstrate advantages in using Algorithm 3 over Algorithm 6.

the percentage of instances where Algorithm 6 yields an incorrect connectivity assessment.

The results of this study are presented in Fig. 4.6. Notably, in the *den520d* map, Algorithm 6 exhibited zero errors, highlighting its potential for significantly faster computation compared to Algorithm 3 in static environments. This observation aligns with the algorithm's adoption in prior work, which has used static network assumptions. However, across the broader dataset, we observed a substantial increase in false positives, reaching up to 100% in some cases. This dramatic decline in performance highlights the need for a base-station-centric, critical strength-aware communication tree, as outlined in Algorithm 3, to support robust decision-making in dynamic and mission-critical scenarios.

4.7 Hardware Trials

The network construction algorithms presented in Section 4.5 were tested on a convoy of robotic platforms in the context of an automated patrol mission through a hospital-like building without any available network infrastructure. The testing environment is illustrated in Fig. 4.7, which depicts the building's floor plan as mapped by the agents during the hardware trials. Figure 4.7 shows that this environment consists mainly of two long corridors (viewpoints 1 and 3), a sharp turn (viewpoint 2), and an exit point to the exterior of the test facility (viewpoint 4). In all experiments, the convoy started at the operator base station near viewpoint 1. The path σ was implicitly constructed using a frontier exploration

algorithm governed by a greedy heuristic to minimize the total distance traveled [20, 64].

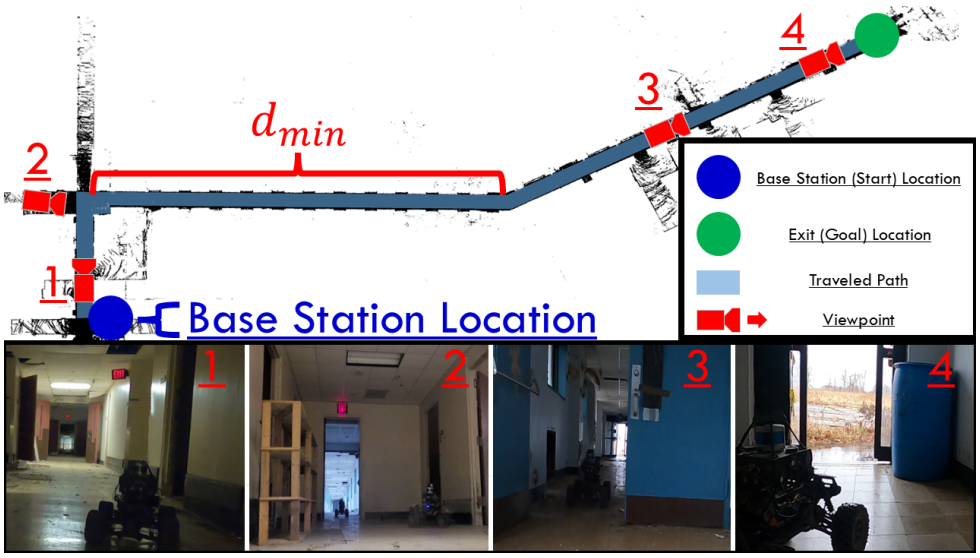


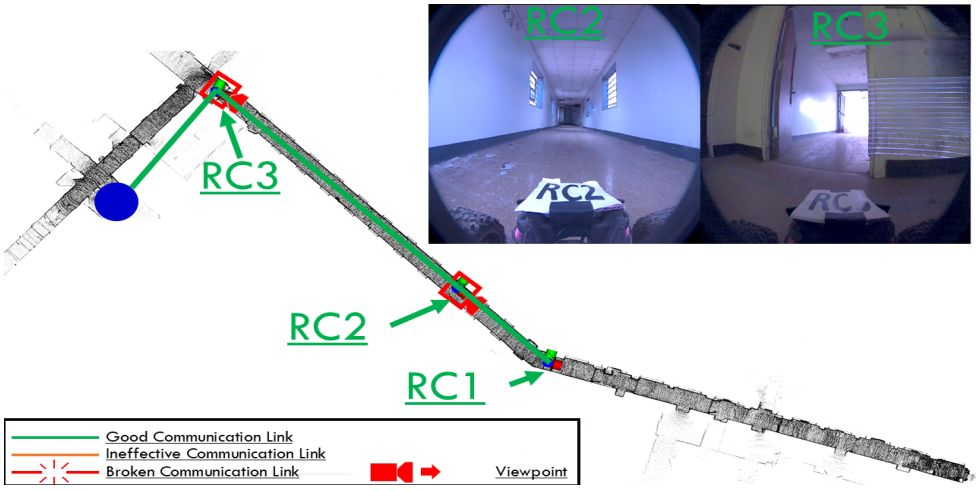
Figure 4.7: Hardware Trial Environmental Overview

The convoy consisted of three mobile robotic agents (Fig. 4.2) that were all equipped with a communications radio. Each agent's radio enabled it to communicate with all other agents or the base station, provided the corresponding radio was also in range. Constant communication between the base station and the convoy was ensured by procedurally building a communications network using Algorithm 4 and Algorithm 5. Further discussion of the mobile agents can be found in Appendix A.

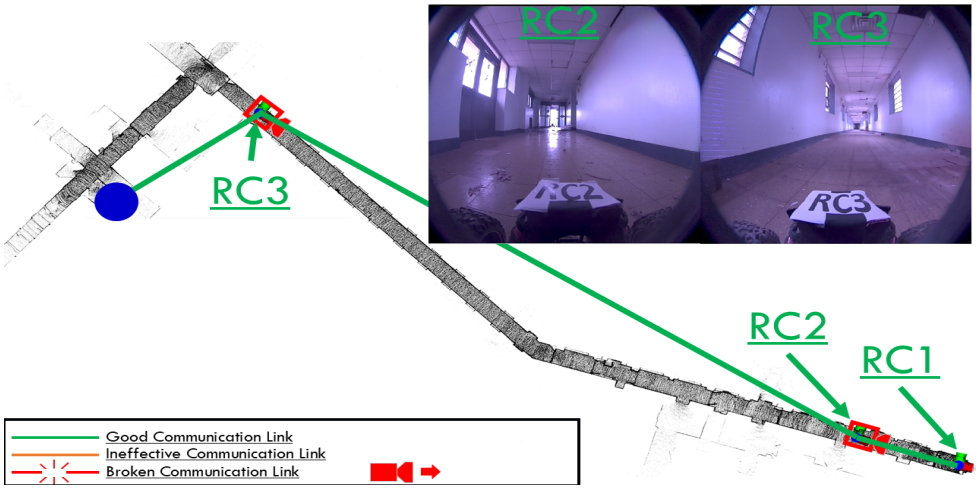
4.7.1 Visibility vs. Communications-Metric Graph Construction

The results of the comparative simulations reflect that distance-based metrics may perform poorly in complex environments with high densities of line-of-sight blocking terrain features. For the given environment in this set of experiments, a singular limit could be observed such that the multi-agent team of robots would have at least one platform reaching the egress point without LOS constraints. This limit is denoted d_{min} (shown in Fig. 4.7), and was measured to be at least 75 [m] (thus, $d_{LOS} = d_{NLOS} = d_{min} = 75$ [m] for the given tests). Any value above 75 [m] would allow the agents to reach the goal. However, given a suboptimal parameter choice for d_{LOS} or d_{NLOS}, the mobile platform team was unable to reach the exit point of the facility. This is reflected in Fig. 4.8a. Here, a suboptimal choice was selected ($d_{LOS} = 65$ [m], $d_{NLOS} = 0$ [m]) which caused platform "RC1" and platform "RC3" to stop at points where the system lost line-of-sight to the previous nodes. Note that line-of-sight to the base station relay was available in the starting hallway. In contrast,

leveraging Algorithm 3 with a communication-based metric (selected as falling below a "signal strength" threshold) provides a different spatial distribution of deployed platforms. Note that the selection of the communication-based metric also required empirical testing to determine the hyperparameter threshold, but did allow for the dropping of the line-of-sight



(a) Test I: Ending state in Algorithm 5 evaluation



(b) Test II: Ending state in Algorithm 4 evaluation

Figure 4.8: The testing environment for all hardware experiments. Note that each camera view taken from the multi-agent team is also labeled on the map. The closest red arrow is an approximate position for the image, and the image was taken in the pointing direction of the arrow's head. Hardware evaluations comparing Algorithm 5 and Algorithm 3. Figure Fig. 4.8a reflects the end positions for the agents using Algorithm 5 and a distance-based metric. Figure Fig. 4.8b reflects the end positions for the agents using Algorithm 3 and a communications-based metric.

constraint, which is advantageous in complex geometries.

4.7.2 Failed Node Robustness Test

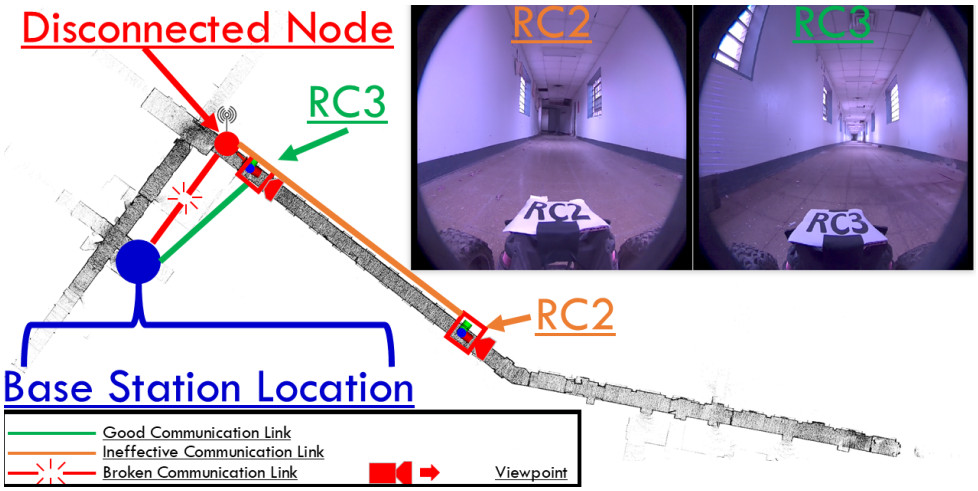


Figure 4.9: The presence of the disconnected node in the environment affects each agent differently. Agent "RC2" does not realize that it cannot communicate with the base station and continues out of range. This contrasts with agent "RC3," which recognizes the communication failure and stops.

In addition to the performance tests, additional robustness tests were performed to handle disconnected or "failed" nodes. Here, a failed node does not communicate with the base station but can communicate with the agents in the convoy. On one agent, "RC3," the maximin communications spanning tree is used to construct the network (Algorithm 4) while "RC2" only uses Team Explorer's strategy (Algorithm 5).

The position of the failed node in the environment is shown in Fig. 4.9 and is labeled "RC1." The placement of this node satisfies both the distance and LOS constraints required for Algorithm 5. Without the maximin communications spanning tree, "RC2" does not stop until the position marked as "RC2" in Fig. 4.9, as this is where the communication range (determined by d_{LOS}) is reached. When "RC2" moves beyond the "RC1" position, communication with the agent began to fail and became intermittent. The post-trial analysis shows that the agent which stopped at the "RC2" position had almost half the "signal strength" needed to communicate with the base station (c_{thresh}). However, with the presence of the maximin communications spanning tree, "RC3" observed this drop in signal strength near the "RC1" position, causing it to deploy as a communications node. That threshold was reached at the position listed as "RC3" in Fig. 4.9. Thus, the presence of

the maximin communications spanning tree restrained platform "RC3" from leaving the communications-accessible area.

4.8 Conclusions

This chapter presents a maximin communication spanning tree approach for ensuring communication between a multi-agent robotic team and a base station. We demonstrate the approach in both simulation and hardware to generate network topologies in complex environments, and show how the maximin communications spanning tree provides additional robustness to disconnected or "failed" nodes in the environment. Immediate future works relate primarily to two areas: 1) map-predictive communication network construction and 2) the development of communication-quality-constrained exploration-based planning and control algorithms for robotic convoying. While the problem definition discussed in Section 4.4 dismissed the effectiveness of minimum set coverage approaches for robotic systems, the use of machine learning to predict environment layout and communications strengths would be a vital step towards generating more optimal relay placements. Works such as Tatum [55] initiated this effort, but more powerful environmental prediction algorithms (e.g., MapEx [65]) could provide a significant improvement over Tatum's baseline. Furthermore, communication-quality-constrained motion planning and control for exploration have been proposed in prior works, but their realization on hardware remains to be seen. Specifically, utilizing perception systems which predict drops in communication quality (e.g., observing known-communication blocking materials or geometries) and then enforcing constraints in the reachable space of a low-level controller provides a unique online benefit towards ensuring the system never leaves communication range.

Chapter 5

Communication Network Construction Behaviors for Robotic Convoying

The communication network construction behaviors discussed in this chapter was presented at the 2025 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)

Citation:

C. Noren, S. Chaudhary, B. Shirose, B. Vundurthy, M. Travers, "Communication Network Construction Behaviors for Robotic Convoying," in *Proceedings of the 2025 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, NDIA, Novi, USA, August 2025

Distribution:

This work was in part supported by OUSD/R&E (The Under Secretary of Defense-Research and Engineering), National Defense Education Program (NDEP) / BA-1, Basic Research. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

5.1 Introduction

ROBOTIC teams have repeatedly demonstrated the capability to effectively search dangerous or difficult-to-reach areas. Such results were most prominently demonstrated during the Defense Advanced Research Projects Agency (DARPA) Subterranean (SubT) Challenge [46]. Given that many of the DARPA SubT test environments lacked communications infrastructure, performer teams often simultaneously completed search and rescue tasks while building a mobile wireless ad-hoc network (MANET). The constructed MANET was often dependent on "drop nodes" –communication relays that were physically "dropped" from robotic platforms– as they explored the environment. In many applications, including dismounted operations, the need for communication between robotic agents remains, but relying solely on drop-node-based strategies has several undesirable features. Such approaches can be time-consuming for human operators, as the operators may need to clean up or collect the drop nodes, and modifying the network topology can be challenging once the nodes are deployed. In this chapter, we build on our SubT experience by truly observing the "*mobile*" name of a MANET. We introduce and integrate a series of MANET construction and modification behaviors for teams of robotic agents that travel in a convoy formation into our framework (see Figure 5.1). We aim to begin a discussion on how to develop automated communications-aware robotic systems that can operate in communications-deprived environments without relying on human operators (e.g., the warfighter, disaster response team members, etc.). We argue that operating as a MANET is a core underlying technology for multi-agent operations, and we demonstrate

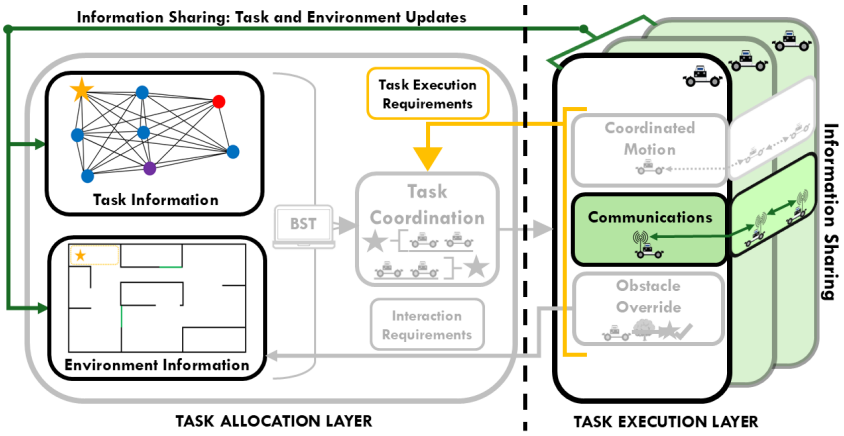


Figure 5.1: In Chapter 4, we developed an ad hoc wireless network construction technique. In this chapter, we focus on further refining and testing those communication network construction behaviors for formations of agents.

the application of such technologies to multi-robotic-agent formation movement problems in cluttered environments.

We consider missions where the robotic agents are required to move throughout an environment while remaining in constant contact with a central base station. Both the base station and the robotic agents are equipped with wireless communication capabilities (nodes) that allow the base station and the agents to communicate with each other at a distance under nominal conditions. Agents move through the environment as part of a formation, typically a convoy, to provide mutual support to one another during their traversal of the environment. As agents move through the environment, the quality of the inter-agent communications is affected by objects in the environment and the distance between the agents. We require that connections between the base station and a multi-agent formation maintain a certain level of signal strength to ensure constant communication between the agents and the base station.

To achieve this operating concept, we introduce two behaviors: 1) the MANET construction technique for formations, and 2) a set of individual agent behaviors to reconnect the network topology if a node in the network fails.

5.2 Related Work

As noted in Section 4.3, we build upon designs originating from the DARPA Subterranean Challenge [46]. As the "Systems" SubT performer teams were required to complete a challenge with similar requirements to our work (i.e., operating in communications-deprived environments), we first provide a brief overview of the different systems developed by each of the SubT performer teams before surveying other relevant communications-aware multi-agent systems literature.

In particular, we would like to highlight the approaches taken by Team CERBERUS [52], Team CoSTAR [56, 66], Team NCTU [54], and Team Explorer [20]. The solutions designed by these teams relied on both inter-agent and agent-operator communication to explore the environment and locate objects of interest. The developed communication solution resulted in teams constructing a "communication backbone" that would allow information sharing (e.g., environmental maps, odometry information, detected objects, etc.) with the other agents in the team [51]. Each of the aforementioned teams relied on both robotic agents and a set of "dropped" communication nodes to ensure that the agents in the system remained in communication with the base station. Certain teams, such as Team CERBERUS, relied on a "Human Supervisor" to command agents to drop communication nodes. This decision was attributed to the team's observations regarding challenges with

ascertaining consistent measures of communication performance in SubT environments [52]. Team CoSTAR designed a robotic behavior that considered both environmental factors, relative line-of-sight (LOS), coverage, and network conditions to determine whether a drop should occur [66]. Team NCTU's solution evolved over the course of the DARPA SubT challenge. The original approach designed by Team NCTU, consisting solely of dropped communications nodes, was augmented with mobile communication nodes mounted to small wheeled robotic platforms. The mobile nodes could be moved in response to poor communication environments, producing a flexible approach that was not dependent on large numbers of dropped nodes [54]. Team Explorer designed a set of distance-based heuristics for both LOS and non-line-of-sight (NLOS) operations, which, when combined with a map prediction algorithm, could be used to minimize the number of dropped nodes [20, 55]. Our proposed system takes lessons learned from each of the SubT performer teams: human supervision (CERBERUS), network-conditioned-based drops (CoSTAR), mobile communications nodes (NCTU), and NLOS (Explorer) operations in order to improve overall system performance.

The design of network construction algorithms is highly dependent on the placement of communication relays throughout the environment. This placement is influenced not only by the environment itself, but also by the current known information about the environment and the communications system. Although both our operating conditions and the environments in which SubT agents performed were *a priori* unknown, *a priori* known environments give rise to computational geometry techniques for sensor placement strategies in wireless sensor networks [67]. Specific techniques frame the problem as a variant of the Art Gallery Problem: a visibility-based approach to predicting sensor placement. Solutions to the Art Gallery Problem minimize the number of sensors (guards) required to cover a (classically) polygonal area [57]. Different sensor models modify the expected visibility of the guards, resulting in varying solutions to these optimization problems. Common modifications include limiting the visible range of any placed sensor [57], distance-based visibility fading [58], and visibility through a limited number of walls (k-transmitter problem) [59]. Although some of these geometric approaches translate to robotic systems, the quality of the information that travels through the communication system also comes into play.

The different performer teams considered multiple measures of wireless communications quality for use in the DARPA SubT Challenge. Certain teams (CERBERUS) side-stepped the issue entirely by relying on human operators, noting that common radio signal strength indicators (RSSI) were often not reliable in subterranean environments [68, 69]. However, other teams (NCTU and Explorer) relied on RSSI to determine the locations of the node drops [20, 54]. Finally, utilizing signal-to-noise (SNR) has been considered in multiple

works adjacent to Team CoSTAR [56], primarily with a focus on planetary exploration. Predicting these communications metrics is challenging, with recent work relying on machine learning techniques to predict the strengths of a 5G signal in indoor environments [70]. Such techniques can be helpful if direct access to SNR values is not provided by the communication system or if the expected SNR value at a specific position in the environment needs to be predicted.

Finally, while many of the approaches deployed in the DARPA Subterranean Challenge demonstrate the requirement to maintain communication with a central communication node, this requirement is often relaxed to reflect the mission context. If agents can operate independently or with limited oversight, this could allow for operations with intermittent connection between the agents or between the agents and a base station or human operator. The acceptability of intermittent communication yields a fundamentally different problem from those previously considered. Although the application considered in this chapter focuses on developing robotic agents that directly support dismounted operators when required, in the interest of completeness, we also wish to detail several recent works that enable intermittent communications. Wang et al. [71] demonstrates a unique solution approach that develops a communication signal map online. In particular, an uncrewed aerial vehicle learns a signal-to-interference-plus-noise map while avoiding an adversarial communication jammer. Although such an approach may be extensible to a communication relay node placement problem, the presented formulation is predicated on a fixed communications environment and allowable finite-time communication service interruptions to facilitate exploration in the communication signal space. Similarly, Woosley et al. [72] uses a Gaussian process to model information entropy and communication signal strength for simultaneous exploration and information collection, but again does not require the exploring agents to maintain communication with a central base station or operator. Finally, we would also like to highlight an approach that requires information transfer between agents but does so by intermittently sharing maps at scheduled rendezvous locations. This approach enables agents to share information without needing a relay network to maintain continuous connectivity with a base station [50]. This method works well to explore an unknown area quickly, but does so at an increased operational independence of the agents themselves (i.e., the agents may not be able to communicate with each other until a rendezvous occurs).

5.3 Technical Approach

In order to ensure communications during a robotic mission, we propose a graph-based mobile ad hoc network (MANET) construction system inspired by our experience in the

Figure 5.2: An operator commanded "peel-off" in an urban environment.

DARPA Subterranean Challenge. We build on these experiences by replacing dropped "communication nodes" with a formation of cooperative ground agents.

5.3.1 System Overview

We demonstrate our approach on the developed robotic convoying system first discussed in Part I and further detailed in Appendix A. The team consists of multiple heterogeneous robotic platforms (agents) and an operator interface. As the agents themselves have been discussed in the previous chapters, we will focus primarily on the operator interface. The operator interface displays information about each robotic platform (e.g., position, current waypoint, relative experienced radio signal strength) and serves as the primary interface for sending commands to the agents. For robotic behaviors that require centralized decision-making, the operator interface also serves as the central point to coordinate the actions of each platform. For convenience, we may abbreviate the operator interface as "BST" ("base station") in the remainder of this chapter.

Operating Concept: Automated Peel-Off

We desire a methodology to ensure that the convoy of agents remains in communication with the base station throughout the operation. Initially, we relied on an operator to manually "peel-off" an agent from the convoy when the convoy was about to leave communication range. Measurements of relative inter-agent communication strength (e.g., RSSI or SNR) were displayed on the operator interface to assist the operator in determining "peel-off"

locations. Triggering a peel-off would cause the last agent in the convoy to leave the convoy and stop. When the agent stops, it acts as a **communication node**, extending the effective communication boundary by relaying information and commands between the operator interface and all robotic convoys or agents in communication range. In order to minimize the number of network connections, convoys are treated as a single "composite" entity, where all information and commands to a convoy are filtered by and passed through the lead agent of the convoy. The lead agent then distributes commands to the remaining convoy agents until the agent leaves the convoy or the convoy is split by the operator. An example of such a peel-off is shown in Figure 5.2. We thus define an automated **peel-off behavior** as a behavior executed by an agent participating in a convoy that causes the agent to 1) leave the formation, 2) come to a complete stop, and 3) relay communications to other assets in the system. We modified this manual peel-off to be callable by the lead agent of an autonomous convoy. This enables the lead agent to direct the following agents to act as communication nodes if a set criterion is met. We describe this behavior as an "automated peel-off" behavior and define the criterion for initiating a peel-off in the following section.

5.3.2 Network Construction Behavior

Determining when to "peel-off" agents in an automated manner is fundamentally connected to the connectivity and interactions between the system assets. Graph data structures provide a convenient means of representing the interrelatedness of assets in our system. In our communications systems, this interrelatedness of assets exists in at least two representative aspects. There is an inherent coupling between 1) the physical (spatial) representation and 2) the signal strength representation of the communication network. For a MANET, the "mobile" nature of a robotic system makes this coupling even more prevalent. As such, we pose our network construction behavior as a graph analysis problem in which the network topology changes as the robotic agents navigate through the environment.

The network topology is constructed from a set of n communications-enabled assets, including: 1) the operator interface, 2) existing friendly environmental communication infrastructure (if any), 3) individual robotic agents, and 4) formations/teams of robotic agents. For simplicity, we enumerate the total list of assets as $\mathcal{C} = \{c_0, c_1, c_2, \ldots, c_{n-1}, c_n\}$ with index set $\mathcal{I} = \{0, 1, 2, \ldots, n-1, n\}$. We define the set of agents in the asset list as $\mathcal{A} \subset \mathcal{I}$ with corresponding index set $\mathcal{I}_a \subset \mathcal{I}$, and, for simplicity, denote asset c_0 as the operator interface. As these assets represent physical entities (e.g., a robotic agent), we define a mapping $p : \mathcal{C} \to \mathbb{R}^n$, which takes an asset index and returns its physical location with respect to a predetermined common reference frame (e.g., the coordinate frame of the lead vehicle).

We represent the relationships between the assets using a pair of undirected weighted graphs. The first graph, $G_d = (V, E_d, w_d)$, captures the physical distances between the assets (i.e., the relative spatial positions). In this graph, each vertex $v_i \in V \subseteq \mathcal{C}, i \in \mathcal{I}$ represents an asset. The edges connecting all vertices are assigned a weight equivalent to the ℓ_2-distance between agents (i.e., $w_d = d(v_i, v_j) = \ell_2(p(v_i), p(v_j)),\ v_{i,j} \in V$). Note that while we overload the notation of $p(\cdot)$, we intend for its meaning to remain the same.

Each asset may also be represented in the network logical layer topology, which describes the communication connections and relative signal strength that the asset has with other assets. We name this representation a "**communications graph**" and define it as: $G_c = (V, E_c, w_c)$. In this definition, the vertex set (V) remains consistent with G_d (i.e., vertices represent assets), but the definition of the edge set does not. In the communications graph, the edges represent the ability for two assets to communicate with each other. These edges are weighted by a corresponding weighting function $w_c = \text{COMM}(v_i, v_j),\ v_{i,j} \in V$. As discussed in Section 5.2, this COMM$(\cdot)$ function can represent several different indicators for signal strength. Finally, note that both graphs are dynamic in-so-much that the relationships between the agents may change during operation (e.g., agent motion through the environment).

Although the communications graph captures the relative communications strengths between assets, it does not inherently indicate *when* an agent should be "peel-off" from a convoy. We propose that the establishment of a communication node (i.e., a "peel-off") should occur in response to a potential loss of communication between the agents and the base station. We model the proposed mechanism as a restriction on the allowable edge weights assumed in the communications graph. Specifically, for every agent i in the agent list $\mathcal{I}_A$, there must exist at least one path in the communications graph that starts at the operator interface (v_0) and ends at the agent (v_i) where all edge weights maintain a value greater than a minimum allowable edge weight. If we define this minimum allowable edge weight as C_{THRESH} and a path in the communications graph as a set of edges between two different vertices ($\pi(v_i, v_j),\ v_i \in V, v_j \in V, v_i \neq v_j$), then we seek to ensure that $w_c(e) \geq \text{C}_{\text{THRESH}}\, \forall\, e \in \pi(v_0, v_i),\ \forall\, i \in \mathcal{I}_A$. For notational simplicity, we drop the dependence on v_0 and choose to represent the paths between the operator interface and an agent i using a single argument (i.e., $\pi(v_0, v_i) = \pi(v_i)$).

To ensure this restriction is observed, we construct a maximin spanning tree, $T \subseteq G_c$, on the communications graph rooted at the operator interface (v_0). The tree is constructed such that the edge weights associated with edges included in tree T maximize the strongest "weakest" link between a parent node and its child node in the tree. Concretely: for each non-root vertex, v_i, connected by edge (v_i, v_j) to its parent node, v_j, an associated edge weight

$\text{CMET}(v_i, v_j) \in \mathbb{R}^+$ is chosen such that $\text{CMET}(v_i, v_j)$ represents the strongest "weakest" link for v_i in the communications graph. As the spanning tree T spans all vertices included in communications graph, and by ensuring that $\text{CMET}(v_0, v_i) \geq \text{C}_{\text{THRESH}}$, $v_i \in V \in T$, we ensure that at least one path between v_0 and v_i exists such that the minimum edge weight is observed for all edges in the tree.

We describe any communication nodes that represent the operator interface or an agent that has performed a peel-off behavior as a **central node**. For ease of reference, we collate these central nodes into a set, denoted as CN. Note that, initially, $CN \leftarrow \{v_0\}$ as the list of central nodes always includes the node representing the operator interface. Thus, these central nodes represent our version of the vital "communications backbone" described by many DARPA SubT works [51]. Maintaining a connection to at least one central node guarantees a communication pathway to the operator interface.

Thus, we can define the network topology $\mathcal{N}$ of the system by considering elements of the communications graph. The network topology itself consists of a "communications backbone" of central nodes and the remaining mobile agents. There exists an effective communications boundary $\mathcal{B}(\mathcal{N})$ created by all the assets in the system. The strength of the system's communications within this boundary, as observed by the system assets, is not less than C_{THRESH}. That is, $\mathcal{B}$ forms a level set $L = \{x \in \mathbb{R}^n \mid \mathcal{B}(x) = \text{C}_{\text{THRESH}}\}$, where $x \in \mathbb{R}^n$ denotes a spatial location in the environment. This communication boundary delineates the limit of the MANET coverage area, crossing which will result in the loss of communication for the agent that crosses the boundary. The automated peel-off behavior described above enables agents to, methodologically, extend the communications boundary $\mathcal{B}$ by tasking an agent to act as a communication node. This communication node extends the current communication boundary, thus ensuring that the remaining agents remain in contact with the operator interface with a minimum communication strength C_{THRESH}.

5.3.3 Network Repair Behavior

During real-world operations, robotic systems are prone to failure (e.g., agent onboard power loss). If such a failure occurred, the resulting communications graph may include an edge with an insufficient edge weight (i.e., below C_{THRESH}). In order to improve system robustness against such failures, we propose a communication network repair behavior that repositions agents to reestablish communication with the root node of the maximin communication tree.

We adopt an approach that repositions at least one agent back to the communications boundary of the failed node. The proposed communications network repair behavior is described in Algorithm 8. The algorithm runs locally on each agent (v_a) and is activated

Algorithm 7 Closest Central Node (CCN)

Require: CN_L, $\mathbb{V}$, x
Ensure: $x_{cn} \in CN_L$ ▷ Closest central node
1: $i = \arg\min(d(x, x_i) \forall\, i \in CN_L, i \notin \mathbb{V})$
2: **return** $x_{cn} = CN_L[i]$ ▷ get i from CN_L

when the signal strength between the agent and the agent's parent node (v_p) in the maximin communications tree falls below the threshold value (i.e., $\text{CMET}(v_a, v_p) < \text{C}_{\text{THRESH}}$). Note that this implies that central nodes may also move to repair the network if their signal strength falls below C_{THRESH}. Given that Algorithm 8 runs on each agent, the motion of an agent acting as a central node could activate the recovery behaviors of any agents dependent on v_a, causing a potentially large number of nodes to reposition in response to a singular node failure. We recommend further study of repair methodologies that minimize the number of agents impacted by the repair behavior (further discussed in Section 5.6).

Before providing an overview of the algorithm, we outline the underlying assumptions of our approach. We first assume that the root node of the maximin communications tree (i.e., the base station) does not fail and that all nodes have the same communications capabilities. To enable an agent to return to the communications boundary of its parent central node, each agent stores the locations of all central nodes in the network. Define the set of central node locations as $CN_L = \{x_0^{cn}, ..., x_n^{cn}\}$, where $x_i^{cn} \in \mathbb{R}^n$ represents the location of central node i (e.g., $x_i^{cn} \in \mathbb{R}^2$ for a planar environment) and x_0 represents the position of the root node. In the event of a central node failure, these locations represent the most likely locations to reestablish the network. Finally, we assume that all locations in CN_L are reachable by all robotic agents from any location in the environment.

For convenience, we define a subroutine (Algorithm 7) to find the central node closest to robotic agent a. Define agent a's position as $x_a \in \mathbb{R}^n$ and the set of central nodes visited by agent a as $\mathbb{V} \subset CN_L$. We first find the index of the central node closest (based on Euclidean distance) to a position x (e.g., agent x_a) that is yet to be visited (Line 1), and then return that node's location (Line 2).

Given these assumptions and Algorithm 7, we propose the Network Repair Behavior in Algorithm 8. We first target the parent node of the agent in the maximin communications tree (Line 2), which is guaranteed to exist via our earlier assumptions. We then check if we are "close enough" (less than a threshold, d_{min}) to the target node's location, x_g, on Line 3, and continue moving towards x_g if we are not (Line 9). If the distance between the target node x_g (initially, x_p, (Line 1)) and agent a becomes less than d_{min} and $\text{CMET}(v_a, v_p) < \text{C}_{\text{THRESH}}$, we assume the current central node location is not a suitable location to repair the network,

Algorithm 8 Network Repair Behavior

Require: x_a, x_p, CN_L, t, $\text{COMM}(v_a, v_p)$
1: Initialize $\mathbb{V} \leftarrow \emptyset$, $x_g = x_p$
2: **while** $\text{COMM}(v_a, v_p) < \text{C}_{\text{THRESH}}$ **do**
3: **if** $d(x_a, x_g) < d_{min}$ **then**
4: **if** $d(x_a, x_0) < d_{min}$ **then**
5: **break** ▷ reached root node
6: **end if**
7: $\mathbb{V} \leftarrow \mathbb{V} \cup \{v_p\}$
8: $x_g \leftarrow \text{CCN}(CN_L, \mathbb{V}, x_a)$
9: $v_p = CN[g]$ ▷ update v_p to v_t
10: **end if**
11: Move towards x_g ▷ for t seconds
12: **end while**

and we must target a different central node. We then check on Line 4 to ensure we are not at the root node (and terminate the algorithm if we are), before adding node v_p to the list of visited central nodes (Line 6). Following this, we then get the next closest central node position (Line 7) and update the next closest central node (Line 8). Consequently, the agent sequentially explores all these locations until it either establishes communication or all locations have been explored (i.e., $\text{CMET}(v_i, v_j) > \text{C}_{\text{THRESH}}$ or $\mathbb{V} = CN_L$).

An important point to note is that the algorithm is guaranteed to recover communications as long as a path exists to the root node of the maximin tree. This is due to the fact that the agent will ultimately reach the location of the base station (root node, x_0) if it cannot regain communication elsewhere.

5.4 Numerical Simulations

The proposed communications network construction techniques and recovery behaviors were first implemented in simulation. Numerical simulations were conducted in environments representative of various real-world scenarios, including buildings, cities, and natural environments. Unless noted otherwise, none of the test environments includes existing communications infrastructure that the robotic team could utilize during its movement through the environment.

We first demonstrate our network construction capabilities on a set of maps from a Multi-Agent Pathfinding (MAPF) benchmark [45]. These results illustrate the evolution of network topology in complex environments, utilizing a communications model to guide the placement of the communication nodes. We also demonstrate our network repair behavior

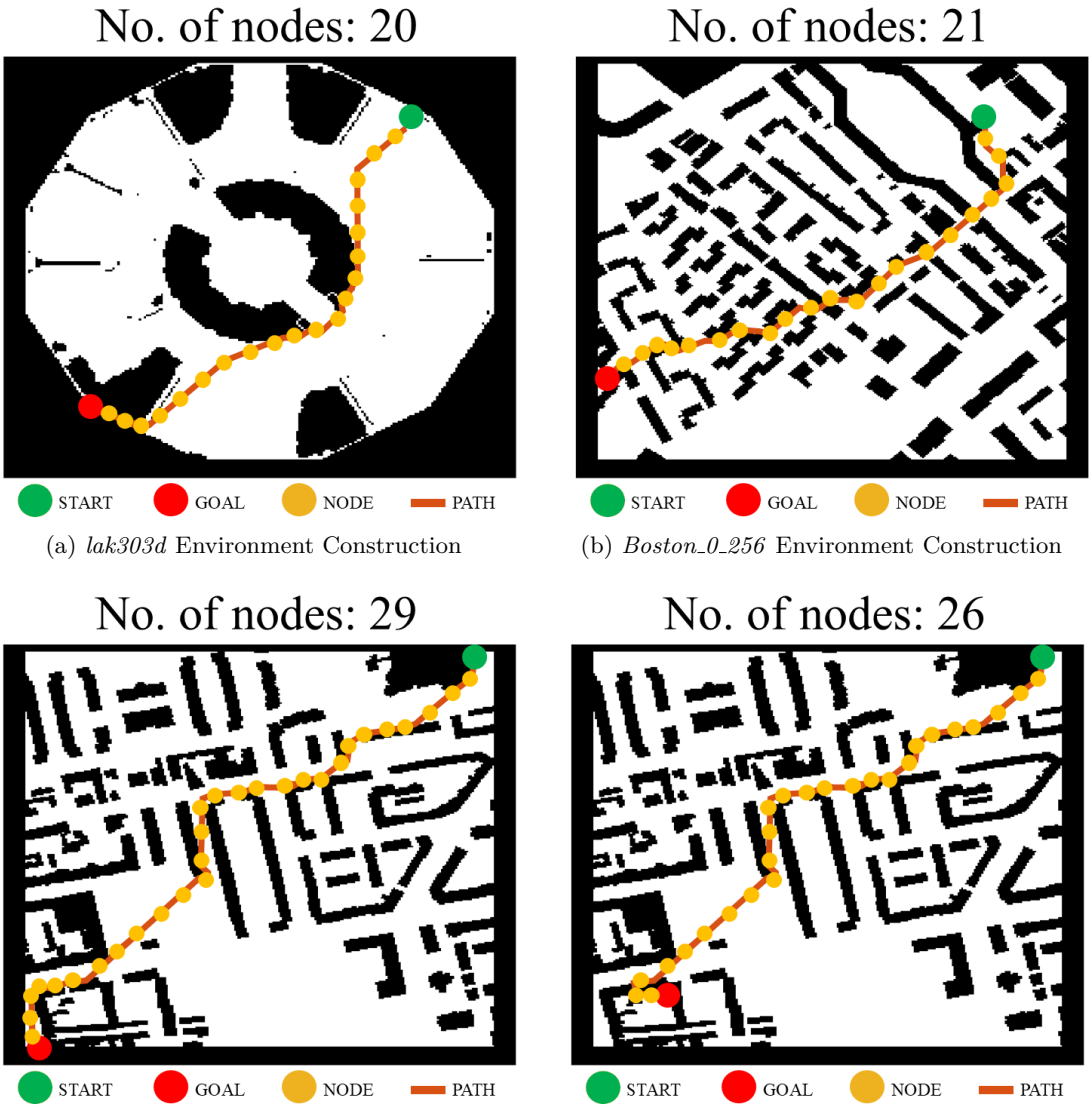
in simulation. The network repair simulation environments are modeled in Gazebo. These studies reflect how communication node failure is addressed using Algorithm 8.

5.4.1 Network Construction Studies

We first simulate a series of missions that require a team of robotic agents to travel through different environments with no existing communications infrastructure. The objective of this set of tests is to determine the required number of robotic agents needed to construct a communications network between an initial start location and a goal location. The environments are drawn from Stern's multi-agent pathfinding benchmark [45]. Each environment is represented as an occupancy map, where the measure of occupancy denotes whether a robotic agent may traverse a cell in the occupancy map. Although the network construction algorithms described in Section 5.3 are not dependent on an *a priori* possession of each environment's occupancy map, each map is assumed to be known beforehand to compute the shortest path between the start and goal states (e.g., via Dijkstra's Algorithm). Given this shortest path between the start and goal states, a communications model is then propagated from an agent (or the base station at the initial start state) along the path. Every robot deployed as a communication node then acts as a further communication source in this communication model, enlarging the communication-accessible area for the other robotic agents.

Although robotic agents are incapable of traversing through occupied cells, in these studies, we assume that the communications model is capable of such obstacle penetration. We represent our communications system using a communication model with distance fading, wherein the communication signal strength is inversely proportional to the distance between the source and receiver. This model is further augmented with additional signal attenuation arising from occupied cells. This additional signal attenuation is assumed at a rate of 20 [cell^{-1}] (e.g., a 20 [unit] reduction in signal strength every 1 [cell] encountered).

We include several reference images for three maps, including: 1) *lak303d*, 2) *Boston_0_256*, and 3) *Berlin_1_256*, to depict the constructed network topologies. For each map, we generate one hundred different start-goal configurations on the same occupancy map. These results are shown in Fig. 5.3.



(a) *lak303d* Environment Construction

(b) *Boston_0_256* Environment Construction

(c) *Berlin_1_256* Environment Construction

(d) *Berlin_1_256* Environment Re-optimization

Figure 5.3: An overview of different constructed networks from trials completed on common MAPF benchmark environments.

Each starting position is denoted with a green circle, and each ending position is denoted with a red circle. Using the communication model above, communication nodes are deployed whenever the signal strength falls below the threshold of 10 (i.e., $C_{THRESH} = 10$). Each communication node that is deployed is represented by a gold circle. The path along which the team travels is shown in orange.

Each environment demonstrates a varying amount of object clutter that the robotic

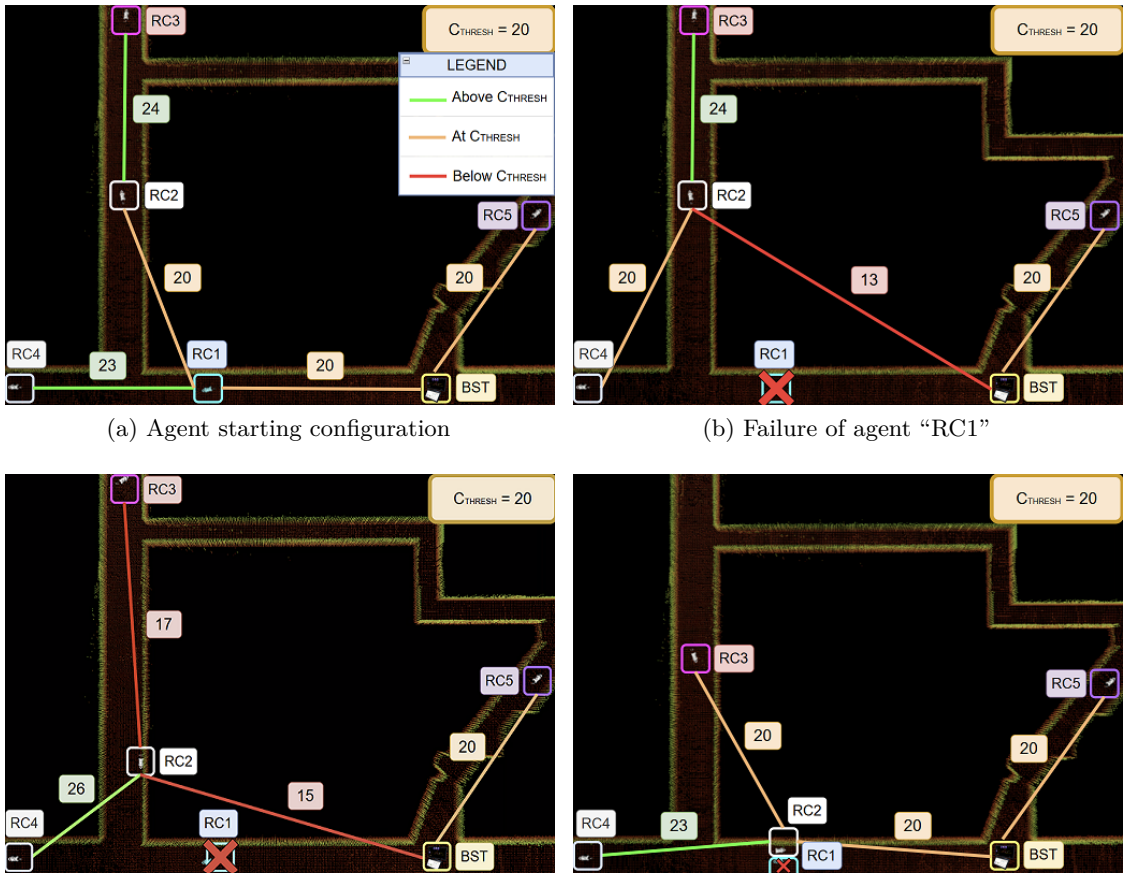
agents must navigate. Large numbers of deployed nodes appear to arise in environments with sharp corners and on objects that must be traversed around that continuously block line-of-sight (*Berlin_1_256*, Fig. 5.3c). We also observe that tight corridors, such as those near the goal location in Fig. 5.3a, typically require a large number of communication nodes to ensure communication quality along the route.

Remark (Re-optimization)**.** *We additionally include a re-optimization of the nodal positions considered in Figure 5.3c to a new configuration in Figure 5.3d. The configuration change is motivated by the new goal location that the agents are required to visit. The result reflects a difference in the required number of nodes (29 to 26), but the remaining agents largely remain in their original positions.*

5.4.2 Network Repair Studies

This section demonstrates the efficacy of Algorithm 8 in different simulation environments. As in the previous simulation section, we adopt a scaled inverse model for our communications system. The communication metric, COMM$(\cdot)$, was defined to be inversely proportional to distance and scaled by a scaling factor k. This scaling factor is a tunable parameter that changes the effective range of the communication model, with a larger scaling factor corresponding to a larger communication boundary. For example, for an agent 10 [m] away from a communications signal source with $k = 500$, our model would predict a signal strength of $\frac{1}{10} \times 500 = 50$. For each test demonstrated in this section, the value of the scaling factor and the threshold C_{THRESH} are specified.

We first present a scenario with five robotic agents and a base station (the root node) in an indoor environment. The results of this test are shown in Figure 5.4. For this test, $k = 500$ and $C_{\text{THRESH}} = 20$. Figure 5.4a shows both the starting configuration of the agents in the communications graph alongside the COMM$(\cdot)$ values for each node with its parent. In this scenario, as "RC1," "RC2," and "RC5" have already peeled-off, the agents are included in the list of central nodes. To test the network repair behavior, we simulate the failure of "RC1" which causes the agent to lose connectivity with all other agents and the base station. As seen in Fig. 5.4b, this sudden failure of "RC1" causes the CMET$(\cdot)$ value of "RC2" to drop below C_{THRESH}. Given the logic of Algorithm 8, the network repair behavior is activated in Agent "RC2," compelling "RC2" to move towards the next closest central node (in this case, "RC1"). As seen in Fig. 5.4d, "RC2" takes the place of the failed "RC1" agent, effectively repairing the network. However, as seen in Fig. 5.4c, as agent "RC2" moves the CMET$(\cdot)$ value of "RC3" decreases below C_{THRESH}. This, in turn, triggers the network repair behavior in "RC3," causing "RC3" to take the previous position of "RC2."



(a) Agent starting configuration

(b) Failure of agent "RC1"

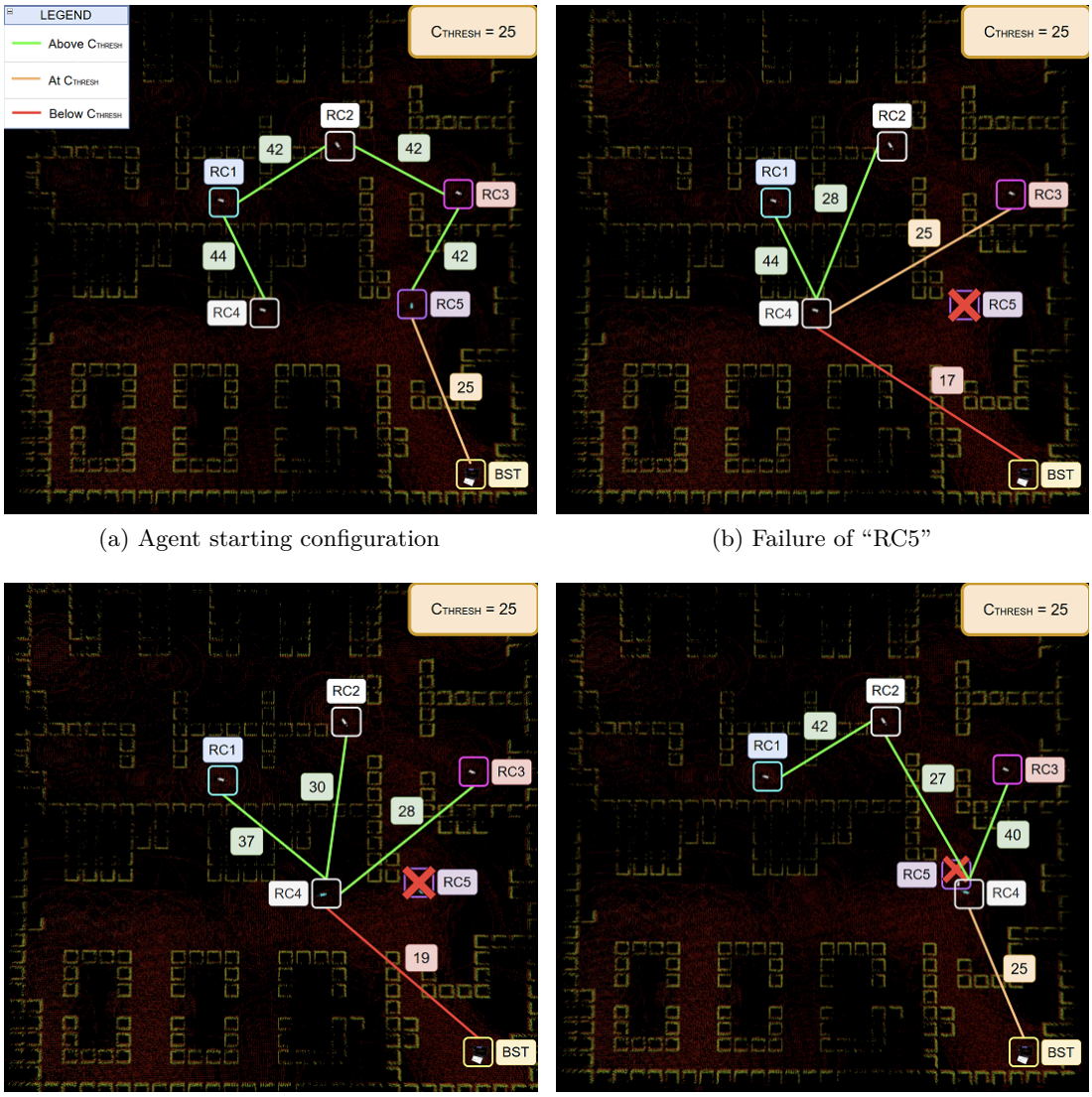
(c) Agent "RC2" moves to repair the network

(d) Agent "RC3" goes below C_{THRESH} & repairs network

Figure 5.4: The first simulation study demonstrates the network repair behavior described in Algorithm 8. The study demonstrates the cascading effect of the network repairing process on multiple agents ("RC2" and "RC3").

While this experiment contains only five agents, it demonstrates the expected cascading effect associated with the network repair behavior on all agents influenced by the failure of a central node.

The second simulation study is shown in Fig. 5.5. For this test, $k = 1000$ and $C_{THRESH} = 25$. As in the previous simulation experiment, Fig. 5.5a shows the initial configuration of the communications graph along with the COMM$(\cdot)$ values for each node with its parent. In this study, "RC5" acts as the only non-root central node (i.e., excluding the base station) in the network. As such, we simulate the failure of agent "RC5" to investigate the system's response. Figure 5.5b demonstrates the effect of the failure of "RC5." The failure of "RC5" triggers the network repair behavior in "RC4," causing it to take the place of "RC5" and



(a) Agent starting configuration

(b) Failure of "RC5"

(c) Agent "RC4" moves to its closest central node ("RC5")

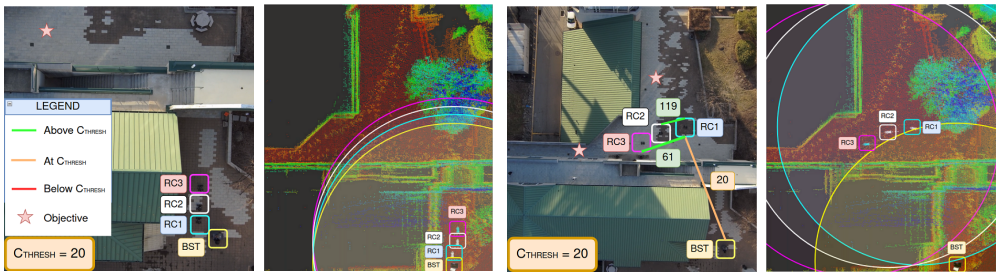
(d) Agent "RC4" repairs the network

Figure 5.5: The second simulation study demonstrates how the network repairing process is only initiated by agents affected by the communication dropout ("RC4").

repair the network (Fig. 5.5c). Note that none of the other robots were affected by the failure of "RC5" as "RC4" moved to repair the network, enabling the other agents to continue their tasks. This experiment demonstrates the effectiveness of the network repair behavior in scenarios where only a single communication node fails.

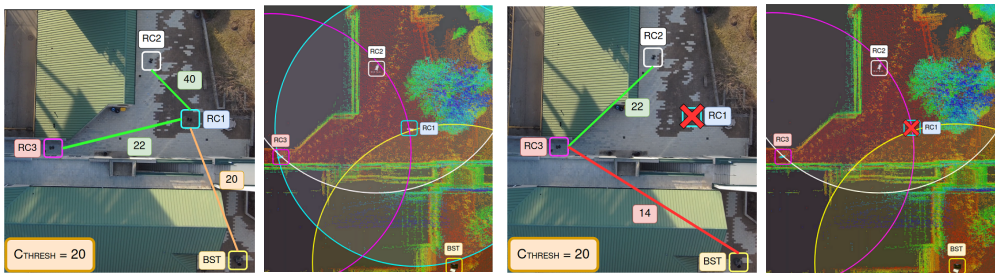
5.5 Hardware Trials

This section details a set of hardware trials that were conducted on a team of wheeled robotic platforms. We demonstrate both our network construction technique and network repair behavior on the small robotic team. We conclude the section with a discussion on computation time.



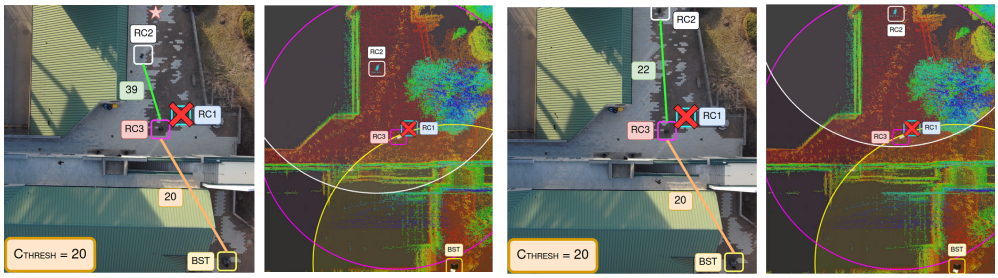
(a) Agent starting configuration

(b) "RC1" peels off as central node



(c) "RC2" & "RC3" continue to different waypoints

(d) "RC1" fails disrupting "RC2"



(e) "RC3" returns to repair the network

(f) "RC2" continues on its mission

Figure 5.6: A system hardware test for the network construction and network repair behaviors. Each image is composed of a full-color aerial image on the left and a corresponding RViz visualization on the right. The RViz screen displays the base station (BST), the agents, the environmental point cloud as mapped by the agents, and the communication boundaries for each node in the communication graph.

5.5.1 Convoy & Network Repair Trials

We choose to demonstrate both the network construction behavior and the network repair behavior on a three-agent robotic team. Fig. 5.6 shows the time lapse of a mission in a city environment using the hardware described in Section 5.3.1. For this hardware trial, $k = 500$ and $C_{THRESH} = 20$.

In Fig. 5.6, each real-world image is accompanied by the corresponding point cloud map –visualized in RViz– that is seen by the operator at the base station. The RViz images illustrate how each node contributes to expanding the communications boundary, enabling the team to achieve objectives that were initially beyond the reach of the communications system. The subfigures also demonstrate the progression of the network construction throughout the mission. Specifically, the different images illustrate how the agents modify their behavior to function as communication nodes, thereby ensuring communication with the base station.

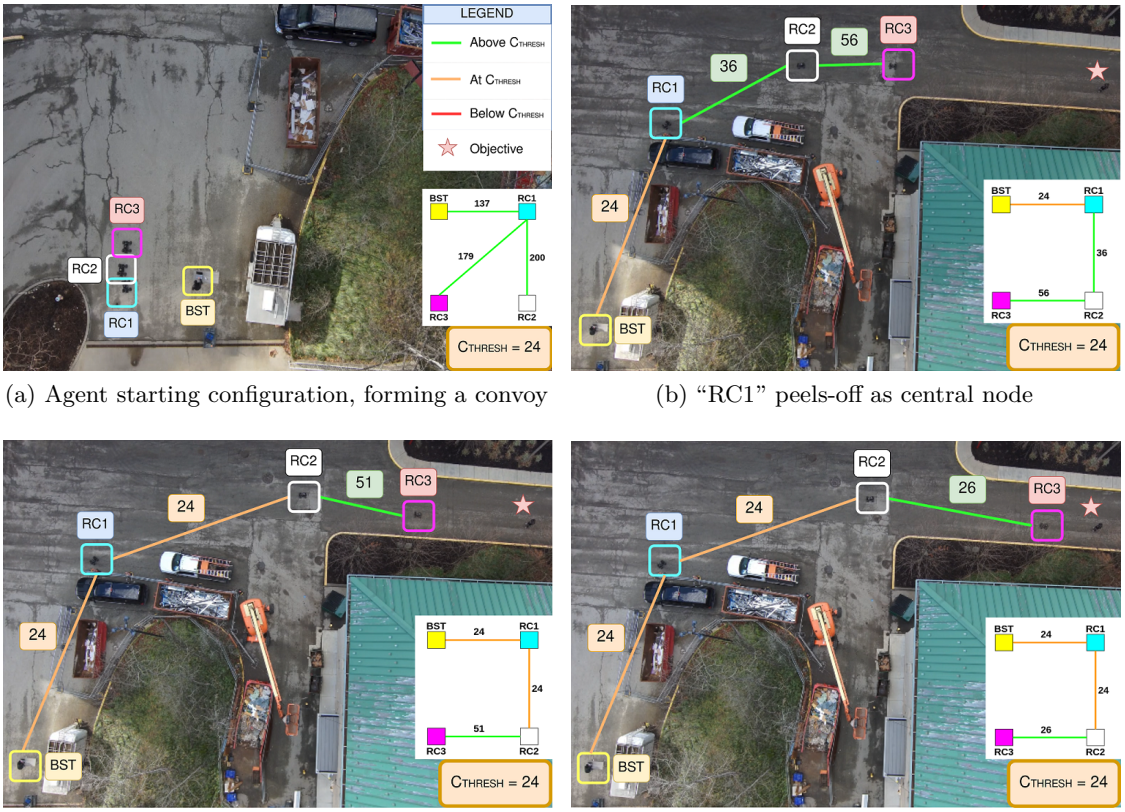
Figure 5.6a shows the agents starting as a convoy (formation) in the vicinity of the base station. The agents are tasked with achieving the waypoints marked by the pink stars in Fig. 5.6. The behaviors and formation structure associated with the agent convoy are further detailed in Chapter 2 and in our previous work [49]. As the mission progresses, "RC1" "peels-off" as it reaches the communications boundary created by the base station and establishes itself as a central node (Fig. 5.6b). This enables "RC2" and "RC3" to continue on the mission, as they can now use "RC1", which is stationary, to relay messages to the base station. "RC1" effectively "extends" the communications boundary, enabling "RC2" and "RC3" to reach the first waypoint.

Agents "RC2" and "RC3" then encounter a fork in the road that requires the two agents to travel in different directions to achieve two new waypoints. The agents diverge to explore each route, as shown in Fig. 5.6c.

To demonstrate the network repair behavior (Algorithm 8) during active operations, we then trigger a failure in "RC1" that causes it to drop from the communications network. This is shown in Fig. 5.6d. The failure of "RC1" causes "RC2" and "RC3" to leave the communications-accessible area, as the signal strength experienced by both robots drops below C_{THRESH}. The network repair behavior causes "RC3" to replace "RC1," enabling "RC2" to continue its mission (Fig. 5.6e & Fig. 5.6f). This experiment demonstrates the functionality of both the network construction and network repair behaviors in real-time on robotic hardware for a real-world mission.

5.5.2 Network Construction Trials

This section discusses the results of the network construction trials using both the "automated peel-off" behavior described in Section 5.3.1 and the maximin criterion described in Section 5.3.2.



(a) Agent starting configuration, forming a convoy

(b) "RC1" peels-off as central node

(c) "RC2" peels-off as central node

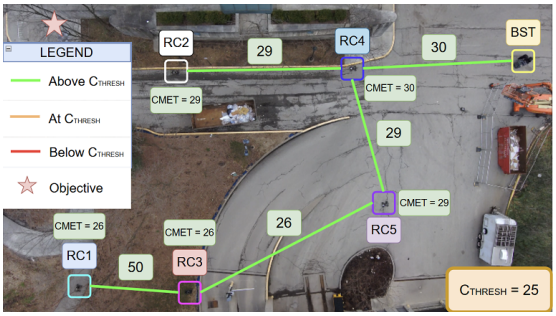
(d) "RC3" reaches objective

Figure 5.7: A depiction of the automated peel-off behavior showing how robots in a convoy "peel-off" and act as relays to enable mission completion. This behavior forms a crucial part of the network construction technique. Each image showcases the communication graph representing the network topology on the bottom right, and the physical layer representation of the same tree overlaid on each image.

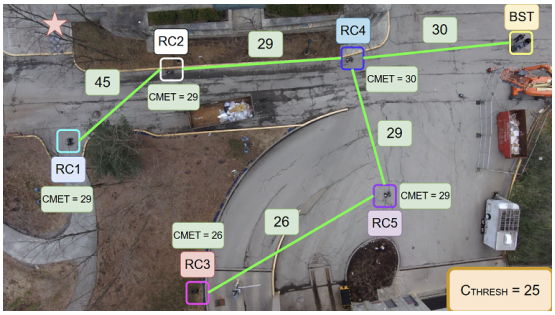
Fig. 5.7 demonstrates the "automated peel-off" behavior for a convoy of robots. For this hardware trial, we set $k = 500$ and $C_{THRESH} = 24$. Fig. 5.7a shows the convoy starting in the vicinity of the base station. Next, Fig. 5.7b shows the convoy progressing towards the objective (represented as a pink star). As the convoy progresses to the objective, "RC1" peels-off as its $COMM(\cdot) = C_{THRESH}$. Fig. 5.7c then shows the convoy's progression until it reaches the communications boundary, causing "RC2" to peel-off. Finally, Fig. 5.7d shows

"RC3" reaching the objective, using "RC1" and "RC2" as communication relays.

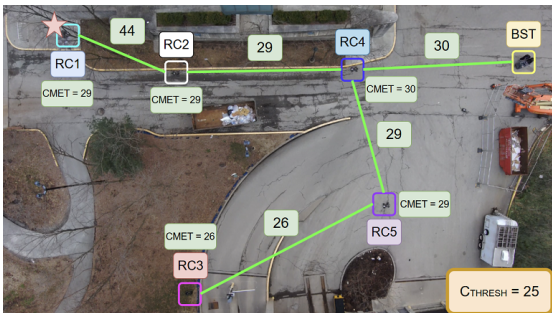
The behavior of the maximin spanning tree is demonstrated in Fig. 5.8. For this hardware trial, $k = 500$ and $C_{THRESH} = 25$. Each image shows the COMM(·) values between pairs of nodes and the CMET(·) value for each node given by the maximin tree. Fig. 5.8c and Fig. 5.8d show the two different outcomes of the experiment, with the former being when all nodes are active and the latter being when "RC4" fails.



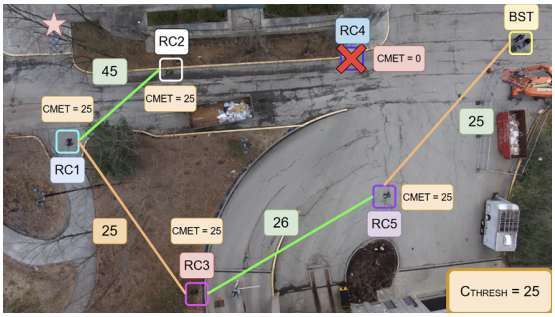
(a) Agent starting configuration



(b) "RC1" connects to BST via "RC2"



(c) "RC2" reaches objective



(d) "RC4" drops out, causing "RC1" to reach C_{THRESH} and stop

Figure 5.8: Demonstration of the maximin criterion for constructing the tree. This criterion considers weak links in the chain of relays (represented by the CMET(·) values for each node), and not just the strongest connection. Fig. 5.8d shows "RC1" stopping despite having a strong direct connection with "RC2". This is because the connection between "RC2" and "RC5" is weak, causing "RC2" to connect to "RC1" instead. Note that links with zero signal strength have not been shown.

In Fig. 5.8a, "RC1" connects to the base station via "RC3." As "RC1" moves towards its objective, it finds a better connection through "RC2" as seen in Fig. 5.8b before reaching its objective as seen in Fig. 5.8c. However, if "RC4," which acts as the link between "RC2" and the base station, fails, it weakens the connection between "RC2" and the base station. This causes "RC1" to connect to "RC3" instead, causing COMM(·) = C_{THRESH}, triggering

a peel-off. This causes "RC1" to fail to reach its objective, which is reflected in Fig. 5.8d.

The difference between the two cases is evident in Fig. 5.8b and Fig. 5.8d. In Fig. 5.8b the CMET(·) value of "RC2" is 29 whereas it is 26 for "RC3." Hence, as "RC1" approaches "RC2," it connects via "RC2" because its CMET(·) value is stronger than the "RC3" CMET(·) value. However, in Fig. 5.8d the failure of "RC4" causes "RC2" to connect to "RC1" instead, as the "RC1" CMET(·) value is 25, which is higher than any of the other possible values for "RC2." As a result, "RC1" remains connected to "RC3" and peels-off.

5.6 Conclusions

In this chapter, we have demonstrated how collaborative mobile robotic teams can be utilized to create a mobile ad-hoc network (MANET) that enables communications in communications-deprived environments. After demonstrating the ability to form a network during a mobile robotic operation, we then showed a network recovery and repair behavior that we have developed to address challenges associated with real-world operations. While the presented hardware demonstrations demonstrate the technical competence of the system, multiple avenues for improvement can be explored to enhance the robustness and capabilities of the system.

Our experiments have demonstrated that accurate communication and signal strength models can be utilized to inform the design of a communications network construction technique. Realistic models of a robot's communications systems are often highly coupled to the hardware used in the system. As simple geometric models are presented in this chapter, the robotics and communications community should continue to strive to develop increasingly accurate models of signal and communications strength. Specifically, incorporating a more accurate model of the communications system that accounts for object interference could help identify better locations during the re-optimization process.

Second, but along the same line of reasoning, the lack of an available environmental map before system operation requires the presented approach to be reactive in nature. By this, we mean that agents must monitor the strength of the communication signal and then "peel-off" as a reaction to poor signal strength. Developing a system that predicts the environment map from a partial observation of the environment could be utilized to decrease the required number of agents by allowing the agents to predict the edge of the communication boundary before reaching it. Finally, developing an adaptive construction policy that enables the system to estimate a communication model online could provide additional benefits if environmental conditions cause the "true" communications model to be out of distribution with the deployed model.

The network repair experiments suggest that a more sophisticated graph optimization or search approach should be employed to determine which agent is responsible for repairing the network. The presented approach has high agent utilization, which is neither reactive to nor predictive of environmental threats. It is clear that exploitative adversarial strategies could be employed to compromise a team of robotic agents using the network repair strategy presented in this chapter. Instead of requiring agents to reach the location of the parent node in the maximin communications tree, a possible avenue for future research could include reorganizing the communications graph topology by employing the computational geometry techniques discussed in Section 5.2 on the already explored map. Alternatively, relaxing the constraint on continual network connectivity could enable the use of techniques such as intermediate or "repair" rendezvous points similar to those discussed in [50]. This can result in a more effective agent utilization with minimal changes to the network topology. Furthermore, our approach relies on the A* path planner to find a feasible path back into the communications boundary. However, there can be scenarios where the planner does not find a valid path because of unobserved regions of the environment. This limitation may also be overcome by using the same aforementioned map prediction algorithm.

Part III

Convoy Interactions with Environmental Obstacles

Chapter 6

Interaction-aware Control for Robotic Vegetation Override

The interaction-aware control system discussed in this chapter was presented in both the Journal of Terramechanics and at the International Society of Terrain Vehicle Systems (ISTVS) 16th European-African Regional Conference.

Citation:

C. Noren, B. Vundurthy, S. Scherer, and M. Travers. "Interaction-aware Control for Robotic Vegetation Override in Off-road Environments," in *2025 Journal of Terramechanics*, vol. 117, pp. 2127-2133, February 2025

Citation:

C. Noren, B. Vundurthy, S. Scherer, and M. Travers. "Trajectory Optimization for Vegetation Override in Off-road Driving," in *Proceedings of the 16th European-African Regional Conference of the ISTVS*, Lublin, Poland, October 2023

Distribution:

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) (#HR001121S0004). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

6.1 Overview

> When [I am] going through vegetation and driving through or over it, [I] always have it in the back of my head how I am going to hit it. Typically, that is head-on with the front bumper, as that will be the strongest point on the vehicle and gives me the best leverage to run over [the vegetation] and go through it.
>
> *Ryan Arciero, Professional Off-road Driver*

The execution of off-road vehicle operations is influenced by the decision of whether to take a deliberate action to "avoid" or "strike" environmental objects [14]. As seen in the quote above, when striking vegetation, professional off-road racing drivers carefully internally model and allow for certain collisions to ensure safe travel off-road. It stands to reason that in order to emulate the high performance of professional drivers in off-road conditions, robotic vehicles will also need to consider the same decisions about collisions. However, there is a significant imbalance of work in the development of the "avoid" and "strike" actions for robotic platforms, with the majority of work focusing on the "avoid" action. We believe that this imbalance has led to a capability gap in robotic off-road operations that require vegetation override. Specifically, we claim that robot control policies that rely solely on avoiding environmental objects exist within a paradigm that both inadequately represents and models the challenges associated with off-road terrains and environmental objects.

The goal of this chapter is to develop an interaction-aware control system that models the

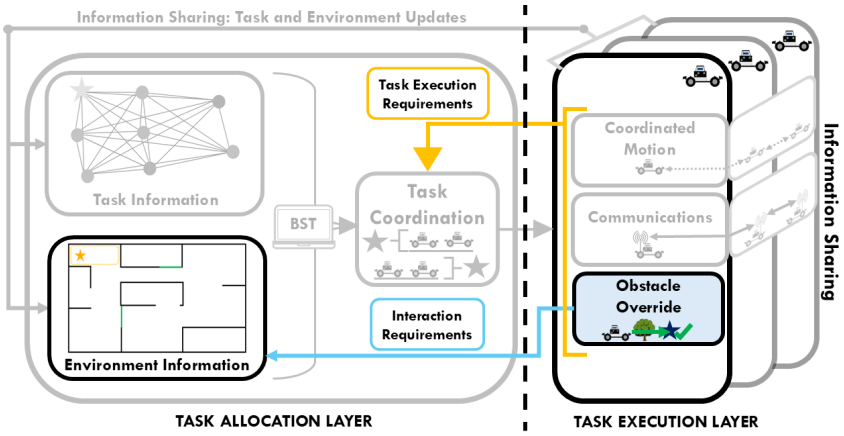


Figure 6.1: In this chapter, we focus on developing an interaction controller for vegetation override. We draw inspiration from Ryan Arciero's quote, which suggests that professional racing drivers interact with vegetation in order to achieve their goals.

effects of object interactions on the motion of a robotic platform. In particular, we focus this development in the task execution layer of our architecture (see Fig. 6.1). This system must provide a principled methodology for reasoning about interactions with vegetative objectives (i.e., "striking") and accurately capture the system's motion when such interactions occur. The system accomplishes this by:

1. **efficiently modeling vegetation interactions** through the use of existing low-computation-cost vegetation override models,
2. **generating dynamically-feasible interaction trajectories** through the imposition of collocation constraints on the vehicle's motion model, and
3. **operating in real-time** to enable the maneuver of a nonholonomic wheeled robotic platform through a cluttered off-road environment.

6.2 Introduction

For tasks such as on-road [73] and on-trail robotic operations, robotic vehicles generally only demonstrate the need for avoidance capabilities. Requirements reaching back to the Defense Advanced Research Projects Agency (DARPA) Grand and Urban Challenge(s) [74, 75] reflect the need for certain areal or object avoidance behaviors. Thus, many methods have evolved in the field robotics, planning, and control domains to address such requirements. Expressing object avoidance as a general constraint in the state space for both state-based planning [76] and for optimal-control-based approaches [77, 78] is well-established [73]. Furthermore, the use of penalty-based methods for object avoidance [79] and avoidance-guaranteed "proof-by-construction" techniques has seen renewed attention in recent years [80, 81]. However, the quote above highlights the need to traverse objects, and thus, only taking avoidance actions or forbidding contact may not accurately reflect expert driving behavior.

Although works that develop the "strike" action set are fewer in number, the field robotics, planning, and control communities have all evolved different means to reason about collisions. The approaches from the field robotics community, such as classifying objects as "strikeable" via visual measures of object geometry or through the use of virtual and physical bumpers [82], draw from the practical needs of robotic platform operation. High-level planning methods, such as those described by Rybansky [14], develop collision rules that depend on the vehicle configuration and environmental structure to govern expectations (for example, maximum traversal speed through an area) on vehicle operations. Outside of treating contacts as disturbances, control approaches generally fall into two

categories: through-contact models and hybrid-contact models. Through-contact models attempt to capture the non-smooth physics of the interaction and directly evaluate those physics during the determination of the control [83, 84, 85, 86]. Hybrid-contact models develop rules-based functional mappings to model the changes of state associated with a discrete event [87] such as a collision. These methods are commonly used in direct collocation trajectory optimization problems for legged locomotion [88]. Unfortunately, many of the techniques that enable contact for a "strike" action primarily focus on the geometric representations of the terrain or object and do not capture all the necessary aspects for traversing or "overriding" an object.

Due to the prevalence and need to interact with vegetation in the operating environment, the off-road mobility and cross-country movement communities have a long history of modeling such nongeometric requirements for vegetation with low computational requirements. The U.S. Army Engineer Waterways Experiment Station [16] and the U.S. Army Engineer Research and Development Center [89] have developed models for the override of different post and vegetation classes. These studies contribute to larger mobility models, such as the North Atlantic Treaty Organization (NATO) Reference Mobility Model (NRMM) [18], which are used to analyze the capabilities of both crewed vehicles and robotic platforms [17]. The NRMM continues to see advancements and refinements to develop higher-fidelity representations of vegetative objects, with recent work focusing on improving override-force modeling using a robotic test platform [90]. Yet, advancements in the vegetation-interaction modeling domain are not solely captured within the NRMM. Rybansky [15] performed a significant study that conducted several vegetation overrides with different classes of vehicles. However, Rybansky's mobility models remain confined to the domain of vehicle mobility analysis and were not used in an online capacity for reasoning in robotic platform operations.

Finally, work in the traversability-prediction domain for off-road robotic driving has also demonstrated the capability to implicitly represent vegetative objects to a robotic system. While many off-road driving datasets include multi-modal sensory data depicting vegetative objects [91], other datasets may include vegetation interaction data itself [92] or label the traversability of vegetative objects by considering the vegetation models from the NRMM or Mason et al. [89] in the labeling process [93]. Implicit (model-free) representations of vegetative objects can then be captured in the learned off-road mobility and traversability policies for off-road operations [94, 95, 96, 97]. In particular, online learning methodologies can leverage observed proprioceptive data [94] or human demonstrations [95] to adapt the system's behavior online in response to environmental stimuli; however, these works do not often directly model collisions.

In particular, the current state-of-the-art for off-road robotic systems that perform traversal with collisions utilizes haptic feedback to update the robotic system's confidence in its environmental obstacle model online [98]. Undoubtedly, when model confidence is low, such a learning-based approach provides a robust foundation for constructing a model representation online. However, the approach does not leverage pre-existing collision models, which enable aggressive maneuvers that exploit awareness of the model during online operation.

6.3 Method (Algorithm) Overview

This section begins with a description of direct collocation methods for trajectory optimization and then discusses vegetation and vehicle modeling. The proposed trajectory optimization control technique is then motivated by the description and models.

6.3.1 Direct Trajectory Optimization Techniques

Trajectory optimization problems determine an optimal state and control sequence that minimize an objective function [88]. Equation (6.1) shows an objective function of the trajectory optimization problem where the boundary term is not explicitly dependent on time

$$J(t_0, t_f, \mathbf{x}(t), \mathbf{u}(t)) = J_f(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} w(\mathbf{x}(\tau), \mathbf{u}(\tau))d\tau, \tag{6.1}$$

and where decision variables $t_0, t_f, \mathbf{x}(t), \mathbf{u}(t)$ are, in order: the initial time, the final time, and the state and control trajectories. Terms $J_f(\cdot)$ and $w(\cdot)$ are the boundary and integral cost terms. The objective function is then minimized in a mathematical program

$$\begin{aligned}
\min_{t_0,\, t_f,\, \mathbf{x}(t),\, \mathbf{u}(t)} \quad & J(t_0, t_f, \mathbf{x}(t), \mathbf{u}(t)), && \text{(6.2a)}\\
\text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), && \text{(6.2b)}\\
& \mathbf{h}(t, \mathbf{x}(t), \mathbf{u}(t)) \leq 0, && \text{(6.2c)}\\
& \mathbf{g}(t_0, t_f, \mathbf{x}(t_0), \mathbf{x}(t_f)) \leq 0, && \text{(6.2d)}
\end{aligned}$$

where (6.2b) are the system dynamics and (6.2c) and (6.2d) represent general path and boundary constraints, respectively.

Direct trajectory optimization techniques approximately solve the mathematical program

for the trajectory optimization proposed in (6.2) by discretizing and transcribing the problem into a more general nonlinear program. A standard-form nonlinear program is

$$\min_{\mathbf{z}} \quad R(\mathbf{z}), \tag{6.3a}$$

$$\text{subject to} \quad c(\mathbf{z}) = \mathbf{0}, \tag{6.3b}$$

$$d(\mathbf{z}) \leq \mathbf{0}\ , \tag{6.3c}$$

where all functions are assumed to be at least $\mathcal{C}^2$ smooth. Direct collocation methods represent continuous-time trajectories with a spline of N time-parameterized polynomial segments; thus, the methods discretize the continuous-time trajectory via the $N+1$ knot points. To ensure that the solution to the program posed in (6.2) is feasible with respect to the system dynamics as described in (6.2b), an integral form of the dynamics is approximated through numerical quadrature. Through the use of different quadrature rules, classically trapezoidal or Simpson quadrature, different approximating polynomials are recovered. These approximate integrals are thus posed as collocation constraints in (6.3b). Additional integral (6.2c) and boundary constraints (6.2d) are posed as constraints in (6.3b) and (6.3c). Finally, a discrete-time representation of the objective function in (6.1) must be posed for the nonlinear program (through approximations such as quadrature) and via an augmented state variable z containing all the decision variables at the knot points.

6.3.2 Vegetation Override Models

The vegetation override models developed by the off-road mobility and cross-country movement communities, including those by Blackmon and Randolph [16] and Mason et al. [89], abstract complex collision interactions into useful low-computational-cost approximations. These approximations generally characterize the required force, work, or velocity a vehicle needs to override a subset of vegetation, given some parameterization of the vehicle or environment.

2012 Mason Override Model

Mason et al. [89] presents a model for vertically embedded objects in the ground. These objects consist of posts and small trees. Equations are introduced in Mason et al. [89] to capture the necessary override force for post-like objects, which were then validated primarily through pull tests. The model in [89] is mainly characterized by vehicle mass and geometry (pushbar height), vegetation and emplacement geometry, soil parameters, and a

series of regression coefficients. Mason et al. [89] then relates a series of energy expenditure and traversal velocity equations from collected data and provides a force model. Equation 10 from [89] describes the minimum velocity at which the vehicle is required to travel in order to override a post, v_{over}, and is given as

$$v_{over} = \sqrt{\frac{2k\alpha\gamma_d D L_t}{m(h + 0.5 * L_t)}}, \tag{6.4}$$

where L_t is the burial depth, m is the vehicle mass, h is the height from the ground surface at which the override force is applied, D is the post diameter, γ_d is the dry density of soil, and α and k are empirical factors derived as described in Mason et al. [89]. As seen in (6.4), weather conditions directly influence the required minimum velocity for override, given that a decrease in the dry density of soil yields a lower override velocity.

1968 Blackmon Override Model

The model presented in Blackmon and Randolph [16] was derived from a series of vegetation override tests conducted on different vegetation types in various environments. Blackmon and Randolph [16] provide unique regressions for force and energy expenditure for these different vegetation types, including singular coniferous and hardwood trees, arrays of multiple trees struck in unison, and "clumps" of bamboo. From continuous measurements of pushbar force, drivetrain metrics, distance traveled, time, measurements of the impacted trees, and characterizations of the aftermath of the collision, Blackmon and Randolph [16] construct a model primarily parameterized by the geometry of individual or multiple trees (for example, the radius of a tree or the clump diameter).

In this chapter, the authors consider only experiments that require the override of a single tree or post. While Blackmon and Randolph [16] provide additional override models, this simplification to a single class of vegetation was drawn from limitations in the perception of the necessary characterizing features for arrays of trees. Thus, the methodologies presented herein are not limited in scope to single standing trees, aside from the limits discussed in the original [16] manuscript itself. Given this simplification, equations B10-B12 from [16], which describe the force and work required to override a single standing tree, are of particular interest. These equations largely take the form

$$F_h = K_f d_s^3, \tag{6.5}$$

and

$$W = K_w d_s^3, \tag{6.6}$$

where F_h is the horizontal pushbar force, W is the work required to fail a single standing tree, d_s is the stem diameter, and K_f, K_w are constants that are dependent on vehicle geometry (e.g., pushbar height).

Note that one advantage of Blackmon's model over Mason's model for use onboard robotic systems is that the model is dependent solely on a visibly measurable quantity (the stem diameter). This single-parameter dependency yields a significantly lower requirement for robotic sensing capabilities and *a priori* soil characterization, likely at the expense of model fidelity.

The measure of work produced from Blackmon's models may then be combined with additional vehicle information (e.g., the operating mass) to generate a suitable v_{over} for a sensed piece of vegetation. The equivalent relations to determine this v_{over} may be calculated in the manner discussed in Mason et al. [89]. This allows for either model to be used in the presented trajectory optimization techniques.

6.3.3 Vehicle Modeling

Figure 6.2: The modified Polaris RZR UTV

Although the mathematical outline provided above can be applied to multiple dynamical systems, the platform used in this chapter is shown in Figure 6.2. The platform is a modified Polaris RZR utility task vehicle (UTV) that can travel up to 20 [m/s] in cluttered off-road terrains. The vehicle is ruggedized to withstand collisions and is equipped with an onboard sensor suite that contains monocular and stereo cameras, as well as multiple Light Detection

Property	Unit	Magnitude
L_T	[m]	3.785
L	[m]	2.972
L_f	[m]	1.412
L_{nose}	[m]	0.915
h_{nose}	[m]	0.533
w	[m]	1.828
m	[kg]	901.0

Table 6.1: Platform Parameters

and Ranging (LiDAR) sensors. A nonlinear bicycle model was used to represent the vehicle dynamics. The vehicle state was modeled as: $\mathbf{x} = [p_x, p_y, \psi, \delta, v]$. In order, the elements of this state vector are the vehicle: x-position, y-position, heading, steering angle, and velocity. The model of the vehicle controls includes acceleration and steering rate: $\mathbf{u} = [a, \dot{\delta}]$. The continuous-time vehicle dynamics are

$$\begin{aligned} \dot{p}_x &= v * \cos\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right), \\ \dot{p}_y &= v * \sin\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right), \\ \dot{\psi} &= \frac{v}{L} \cos\left(\operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right) * \tan(\delta), \\ \dot{\delta} &= \dot{\delta}, \\ \dot{v} &= a, \end{aligned} \tag{6.7}$$

and the parameter values may be found in Table 6.1. The vehicle's total length is L_T, wheelbase length is L, front axle to center of mass distance is L_f, front axle to nose distance is L_{nose}, lower bull-bar strut height is h_{nose}, width is w, and the mass of the vehicle is m.

6.3.4 Direct Collocation for Vegetation Override

A central aspect of the collision models presented in the previous section is that each model captures the loss of kinetic energy due to the collision, corresponding to the failure of the vegetation or post. However, the structure of the trajectory optimization problem and the nonlinear program in (6.2) and (6.3) require at least $\mathcal{C}^2$ smooth functions to represent the dynamics in the collocation constraints accurately. Thus, to incorporate the previously mentioned vegetation models in the trajectory optimization, the impact dynamics must be accounted for. From the formulation, the collocation constraints are evaluated at specific

collocation points, which need not be the same as the knot points described earlier. This realization gives rise to the idea of introducing a pointwise-in-time discontinuity in the form of a functional map, commonly referred to as a "jump" map, between the two states that represent the collision. This concept exists at the center of hybrid-contact representations of collisions and multi-phase direct collocation methods, and is discussed in more detail in [87] and [88]. These approaches have not been implemented using the aforementioned vegetation models.

To capture the effects of the collision at a knot point, a mapping must be defined to transition the state at the time of collision, $\mathbf{x}(t_{col})$, to a new post-collision state. That knot point representing the pre-collision state, $\mathbf{x_{col}} = \mathbf{x}(t_{col})$, is mapped to the next knot point via the mapping

$$\mathbf{x_{k+1}} = \mathbf{f_{col}}(\mathbf{x_k}), \tag{6.8}$$

where subscript k is a general indexing of the knot points, instead of via the selected quadrature rule. While the technique is compatible with the different collision models as described in the vegetation modeling section, this work represented the collision as a loss of velocity at the point of collision. More specifically,

$$\mathbf{x_{k+1}} = \mathbf{f_{col}}(\mathbf{x_k}) = \mathbf{x_k} - [0; 0; 0; 0; v_{over}], \tag{6.9}$$

where v_{over} could be defined (for example, via Mason et al. [89]'s model) in (6.4), where the last element of $\mathbf{x_k}$ corresponds to the velocity in (6.7). However, enforcing such a mapping at the point of collision does not solely account for the effects of the collision. For the simple jump map in (6.9), collisions could propel the system in reverse at low speeds, so either a guard function or a lower bound on the minimal allowable velocity must be enforced to ensure compliance with physical laws. A natural requirement to ensure the vegetation is overridden is to enforce that the velocity reaches at least v_{over} at the time of collision. Just as the velocity constraint is imposed at the point of collision, additional constraints or allowances may be associated with the knot point $\mathbf{x_{col}}$.

Unfortunately, specifying the time of contact or the contact sequence is a non-trivial matter. For the posed collision problem, although the time of collision is unknown, the point in space at which the contact occurs is known. Fixed time-stepping methods that discretize the problem posed in (6.2) with a constant time step are challenging to use, as specifying which particular knot point (if any) will represent $\mathbf{x_{col}}$ is equivalent to knowing the specified contact sequence at best and may result in an unsolvable problem at worst. However, by incorporating the time step used during the determination of the collocation constraints into the decision variables and allowing the solver to determine a unique time

step for each segment, the knot point that specifies the collision takes on new meaning. Instead of representing a specific time of collision, the knot point $\mathbf{x_{col}}$ can now be enforced as a specific state at the point of collision through an equality constraint

$$\mathbf{x_{col}} = \mathbf{x_{obj}}, \tag{6.10}$$

where $\mathbf{x_{obj}}$ is a positional representation of the object in the state space. Equation (6.10) is then added as a general equality constraint in (6.3b). Selecting the specific index of $\mathbf{x_{col}}$ is a method hyperparameter. Associated with this additional time decision variable is the need to provide bounds on the sizes of the time steps. Simple inequality bounds may be posed as $h_{low} \leq h \leq h_{high}$ and $h_{low} > 0$. This constraint on the time step lower bound ensures that the solver does not take unphysical actions.

Finally, if no feasible trajectories are found, such as when the required velocity to override a piece of vegetation is not obtainable before collision, the platform is commanded to remain stationary or to take an emergency avoidance action. While no uncertainty in the position of the object was considered in this chapter, uncertainty in the measurements of the vegetation was addressed by taking the maximum v_{over} as prescribed by the vegetation model during the time the vegetation was observed.

6.3.5 Algorithm Overview

For a given vegetation model (such as Mason et al.'s 2012 post-override model), a set of vehicle dynamics, and the interaction model capturing the "jump map" in the prior section, an operational logic can be implemented on a robotic platform to override vegetation-like objects in the environment. The approach is outlined in Figure 6.3. Note that this approach assumes that any vegetation considered for override may be overridden if the override velocity can be reached from the initial state, given the dynamics (6.2b).

This logic is demonstrated in an example case motivated by the presented experiments. Given the vehicle's starting location (position "A" in Figure 6.3), a target (goal) location (position "B" in Figure 6.3) and a single intervening piece of vegetation (located at x_t in Figure 6.3), the vehicle computes the override velocity (v_{over}) from the selected override model. Different characteristics of the observed piece of vegetation may imply that different override velocities are required at the point of collision ($\mathbf{x_{col}}$). For the example figure, assume that the vegetation located at x_t has an override velocity that scales with trunk size. For the given trunk size, the required override velocity is v_2, outlined in purple. In order to be successfully traversed, the vehicle must reach a velocity of at least v_2 at the point of collision. For vegetation with smaller trunk sizes (for example, vegetation that requires an

override velocity v_1 corresponding to the green line), a smaller velocity may be achievable given the initial velocity of the vehicle. For vegetation that requires a large override velocity, the required velocity ($v_3 > v_{col}$, in red) may not be overridden safely, as the required velocity can never be reached. The vehicle then executes the corresponding action set determined by the trajectory optimization technique that corresponds to the velocity profile that achieves a velocity equal to or higher than the override velocity (v_2 in this example instance). In the example, either velocity trajectory that takes a velocity higher than the override velocity is thus valid, and a further selection of which velocity is taken depends on other design choices. For example, an emphasis on an added margin of safety may prefer the velocity trajectory corresponding to v_{col} while a preferred minimum-kinetic-energy approach may take the velocity trajectory corresponding to v_2. In this chapter, we select the trajectory that minimizes the point-wise ℓ_2-distance evaluated at the knot points between a reference velocity (nominal travel speed) and the trajectory.

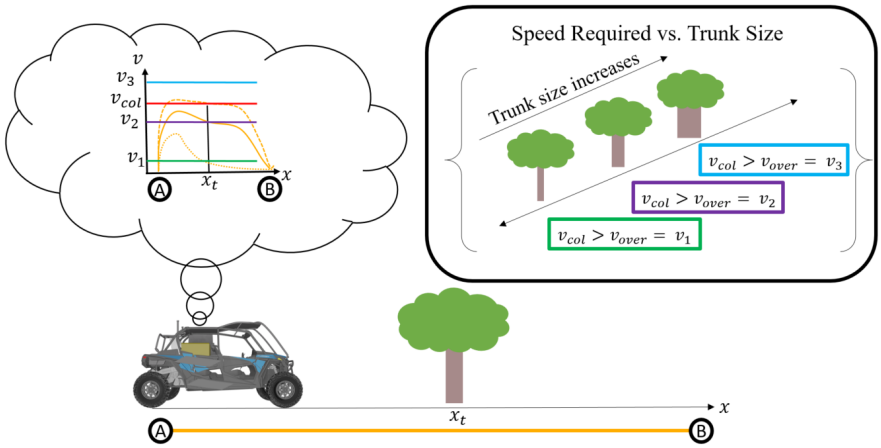


Figure 6.3: Control logic diagram of a vehicle performing a vegetation override in the presented control framework

6.4 Simulated Vegetation Override

Before running on the vehicle hardware, a series of tests were conducted in simulation. For all simulation tests, the Mason et al. [89] model was utilized, with modifications limited to the vehicle and post parameters (L_t, m, h, D).

An example override for a 31.75 [mm] post that is embedded 0.3048 [m] into the ground is shown in Figure 6.4. The post is placed 20 [m] in front of the robot in the x-direction. In the figure, the top plot represents distance from the goal, and the bottom represents the

simulated velocity. Of particular interest in this simulation is the collision that occurs at the 20 [m] mark, where the velocity drops substantially due to the collision with the simulated post. In the velocity graph included in Figure 6.4, this drop in velocity is labeled v_{over}. To decrease the time needed to solve the trajectory optimization problem, the problem was initialized by dissecting the trajectory into m segments, where m is the number of expected collisions plus one. For a scenario with a singular object that must be overridden, such as in Figure 6.4, the entire trajectory that is generated is coupled together at the collision points. This is shown by the two colored areas in Figure 6.4. The bifurcation of the trajectory requires that an additional boundary be generated for each section. For the presented scenario, the initial candidate solution was then constructed by linearly interpolating between the start position and $\mathbf{x_{obj}}$, and then between $\mathbf{x_{obj}}$ and the end position. While additional optimization could be performed, for the problems considered in this chapter, the initialization dropped the number of solver iterations by around a third. Note that the one exception to this statement is in the initialization of the control trajectory, in which the initialization held constant values.

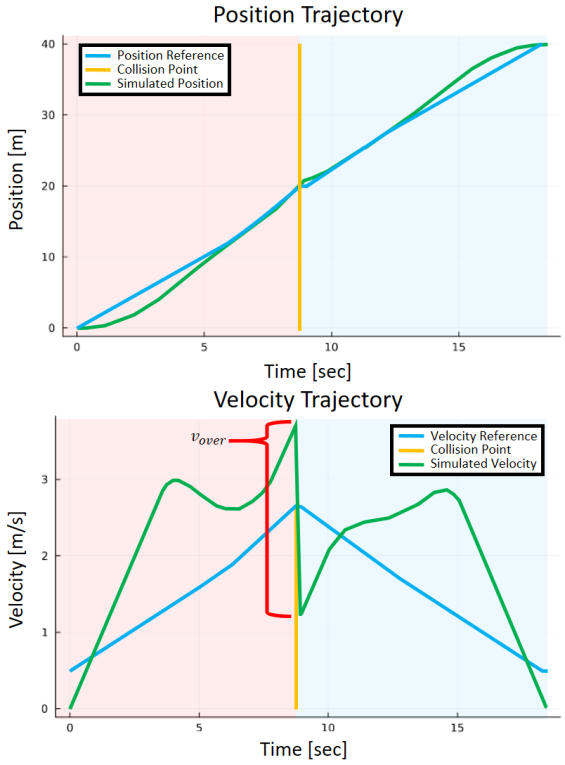


Figure 6.4: Simulated vegetation override for 31.75 [mm] post. The controller achieves an actual velocity (green) above the required threshold.

6.5 Vegetation Override Hardware Results

Three hardware results are presented in this section. The first two results utilize post-override models from Mason et al. [89], and the third result employs the longitudinal override model for a single standing tree, as described in Blackmon and Randolph [16]. In the experiments, the vehicle was commanded to travel at a nominal speed of 5 [m/s] for the first two experiments and 3 [m/s] in the final experiment. A model predictive controller, similar to the unconstrained version presented in Jianyu et al. [77], was implemented as a tracking controller. The results presented in Figure 6.6, Figure 6.8, and Figure 6.10 demonstrate that the tracking controller follows the trajectory as determined by the trajectory optimization controller.

The reported velocity for each run was calculated using a feature-based Simultaneous Localization and Mapping (SLAM) methodology known as "Super Odometry" [99]. Super Odometry fuses multiple sensing modalities, including LiDAR, Inertial Measurement Unit(s) (IMU), and Global Navigation Satellite System (GNSS) to simultaneously provide a registered pointcloud map of the environment and an estimate of the system odometry. An XSens MTI-630 AHRS IMU ("XSens", one onboard) and Carnegie Robotics Duro GNSS ("Duro", one onboard) were fused in the SLAM setup to generate a state estimate with a position accuracy greater than 0.4 [m] and a velocity accuracy greater than 0.08 [m/s]. While operations in Experiment 1 and Experiment 2 were conducted in environments where GPS could sufficiently localize the robotic system, in regions with heavier canopy cover, such as in Experiment 3, localization from GPS data alone may be difficult. However, the large number of unique environmental features (e.g., tree trunks) captured by the onboard LiDAR provided sufficient environmental characterization to localize the robotic system. Initial concerns that matched features existed primarily on the overridden object proved to be unfounded in sufficiently dense forest environments.

6.5.1 Experiment 1 - Straight Line Test

The first experiment that was conducted on hardware was a straight-line test. The object that the platform collided with was a 31.75 [mm] pine dowel rod that stood 0.914 [m] above the ground and was embedded around 0.305 [m] below the ground. The soil surrounding the dowel rod was compacted by hand. It had not rained for more than a week, and soil conditions were dry, even at a depth of 0.305 [m]. See Mason et al. [89] for further discussion on the influence of weather conditions on required minimum override velocity. The minimum required override velocity computed using Mason et al.'s 2012 model was 2.7118 [m/s]. The post was placed around 11.70 [m] from the front of the robotic platform. The red arrow in

Figure 6.6 marks the position of the object as seen by the robotic platform's perception system. In Figure 6.6, the vehicle began its trajectory at viewpoint zero. GPS had the vehicle localized at the center of the blue circle in Figure 6.6 at the time of collision. The vehicle's end goal is depicted as a red square.

In order to constrain the platform into overriding the vegetation, "keep-out zones" were enforced around the vehicle. These zones are all areas shown in bright pink in Figure 6.6. The gray regions in Figure 6.6 indicate all the surrounding objects that are above a height limit of 0.5 [m]. The planned vehicle trajectory is shown in green. Platform viewpoints are provided in Figure 6.6, which are captured along the executed trajectory. A subset of key viewpoints corresponds to the numbers located on the obstacle and collision map subfigure in Figure 6.6. Note that the vegetation in each photo is highlighted with a bounding box for better visibility of the extremities of the vegetation.

The vehicle reached a speed greater than the necessary threshold in order to impact the vegetation. The collision with the pine dowel rod occurred at around 4.5 [m/s]. The registered collision time was earlier than expected, but this is likely due to the fact that the point of time of collision is calculated when the vehicle first makes contact with the vegetation, and the vehicle models used in the collision planning did not account for the length of the vehicle's nose. The vehicle suffered no damage during the test, but the post failed almost completely. A pair of figures depicting the post before and after the experiment

(a) (b)

Figure 6.5: Two images of the 31.75 [mm] post used in the first experiment. Figure 6.5a demonstrates the condition of the object before being struck by the vehicle. Figure 6.5b demonstrates the condition of the failed object after being struck by the vehicle.

are shown in Figure 6.5.

After the collision, the post remained at an angle, with the end of the post that was suspended in mid-air sitting around 0.20 [m] over the ground surface. The post had displaced some soil, as evident in the post-collision subfigure in Figure 6.6. While the experimenters compacted the soils before the tests, this displaced soil may be evidence of a lack of strong compaction at the surface. The post failed roughly 0.271 [m] from its bottom-most point, a little below the ground plane.

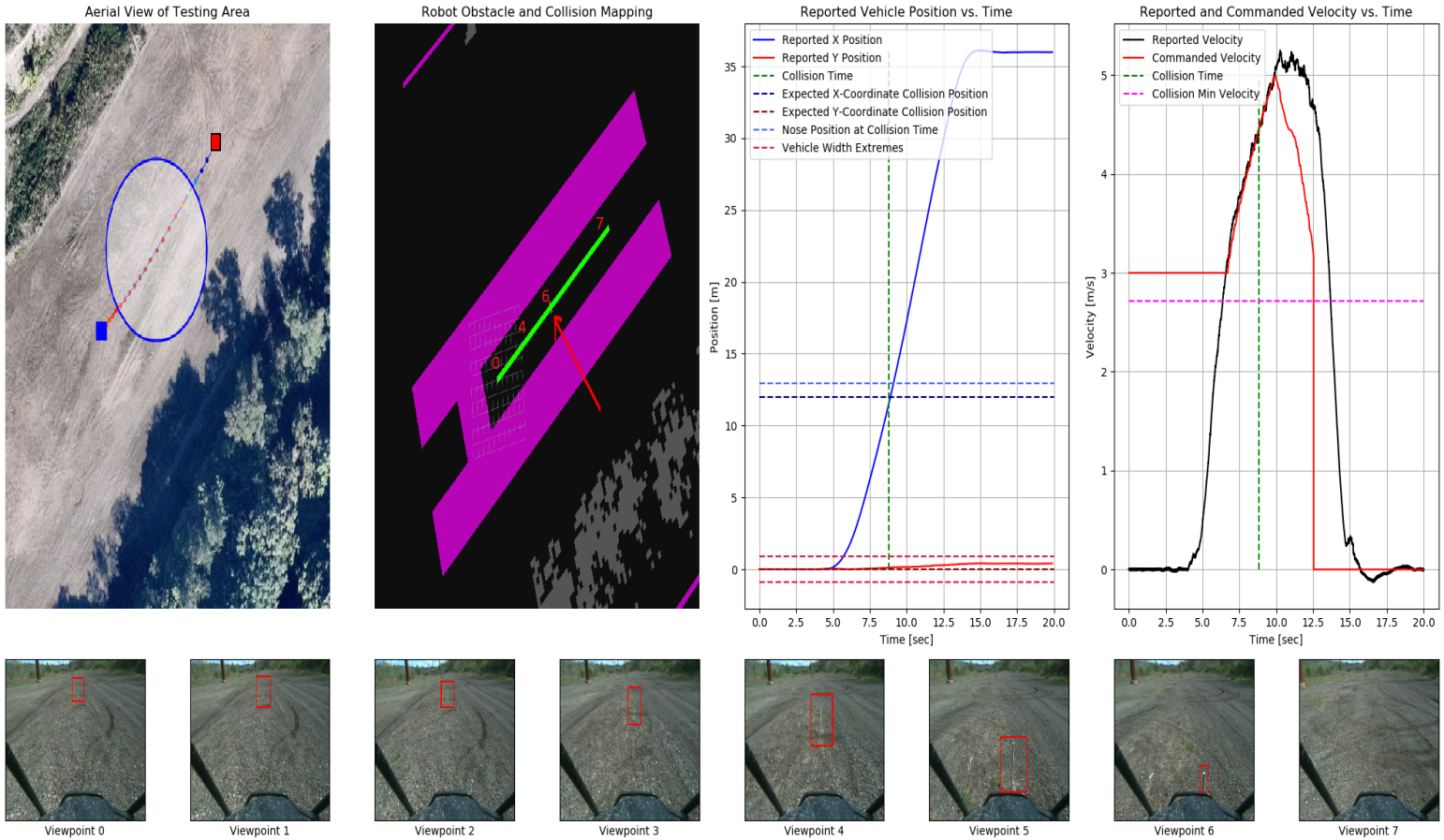


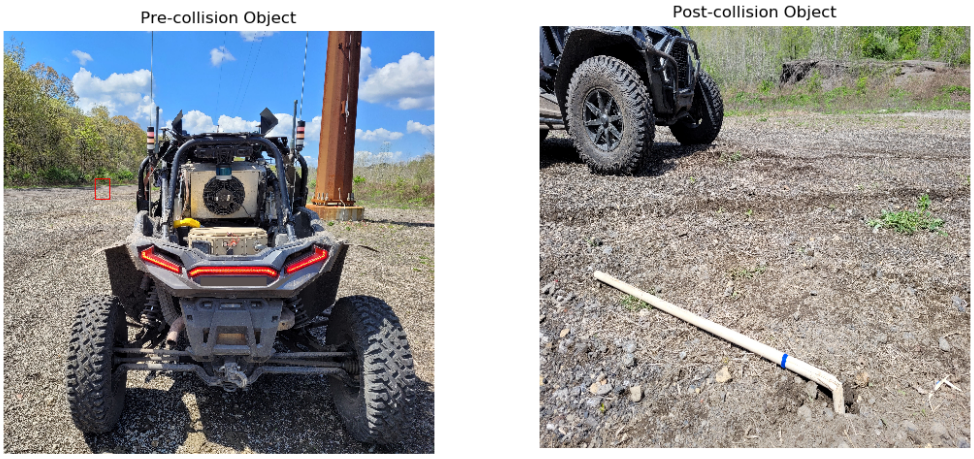
Figure 6.6: The first hardware experiment: a straight line trajectory through an embedded 31.75 [mm] post.

6.5.2 Experiment 2 - Post Override Test

The second experiment used the model from Mason et al. [89] to represent the post for override. This post is 25.4 [mm] in diameter, stands 0.914 [m] above the ground, and is embedded around 0.305 [m] below the ground. The parameterization of the soil was assumed consistent with the results found in Mason et al. [89]. This yielded an override minimum velocity of 2.4255 [m/s]. The post's position is marked with indicator "B" in Figure 6.8. Additionally, the electrical pole in Figure 6.8 that may be seen in viewpoint zero is indicated with an "A" in the collision map for localization. The vehicle began its trajectory at viewpoint zero. GPS had the vehicle localized at the center of the blue circle in Figure 6.8 at the time of collision.

The vehicle reached the target speed and x-coordinate position at the correct collision time. Again, note that the vehicle's nose is making contact with the pole at the time of the collision. Another thing to note is that the vehicle did strike the post on the passenger side of the front bull-bar. This is a near-head-on collision with the post, but the alignment was meant to be towards the center of the bull-bar, not the side.

After the collision, the post was completely failed by the platform. Additionally, the post had been slightly pulled out of the ground by around 50 [mm]. Figure 6.7 contains an image of the failed post. As in Experiment 1, the platform was not harmed.



(a) (b)

Figure 6.7: Two images of the 25.4 [mm] post used in the second experiment. Figure 6.7a demonstrates the condition of the object before being struck by the vehicle. Figure 6.7b demonstrates the condition of the failed object after being struck by the vehicle.

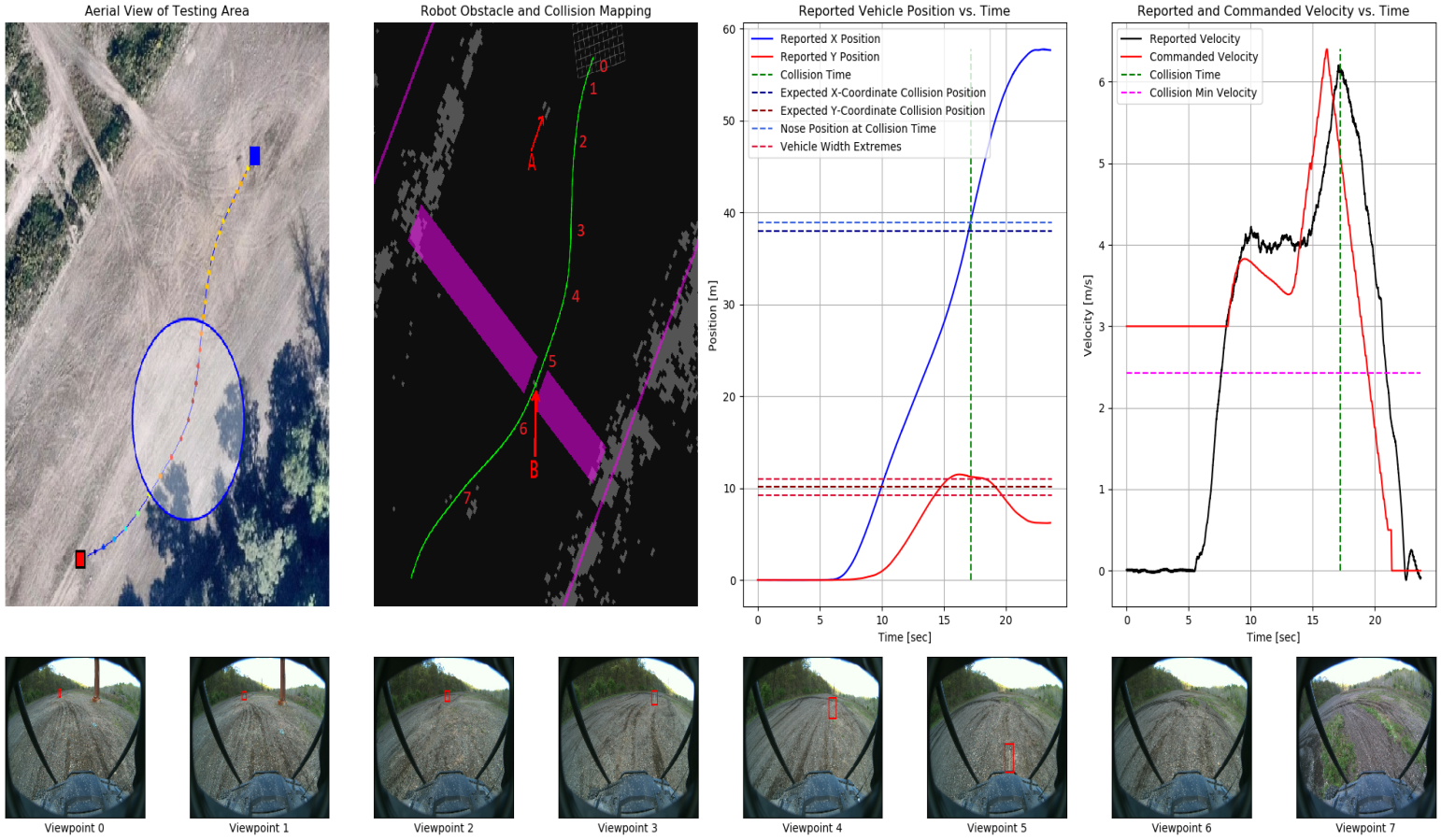


Figure 6.8: The second hardware experiment: a turning trajectory through an embedded 25.4 [mm] post.

6.5.3 Experiment 3 - Small Tree Override Test

The final experiment employed a simplified model from Blackmon and Randolph [16] to calculate the required override velocity for the small piece of vegetation (a tree). In Blackmon's model, the expression for the required work needed to fell a single standing tree relies on the diameter of the stem. This information, combined with vehicle inertial information, may then be used to calculate an override velocity in the manner presented in Mason et al. [89]. As the tree does not maintain a uniform radius, the average width of the tree at the point of impact (81.8 [mm]) was used as an approximation. This measurement yielded a minimum override velocity of 0.758 [m/s]. The tree's position is approximately at the center of the yellow box in Figure 6.10. The vehicle began its trajectory at viewpoint zero, and the GPS had the vehicle located within the blue circle at the time of collision.

(a) (b)

Figure 6.9: Two images of the 81.8 [mm] tree overriden in the third experiment. Figure 6.9b demonstrates the condition of the object before being struck by the vehicle. Figure 6.9b demonstrates the condition of the failed object after being struck by the vehicle.

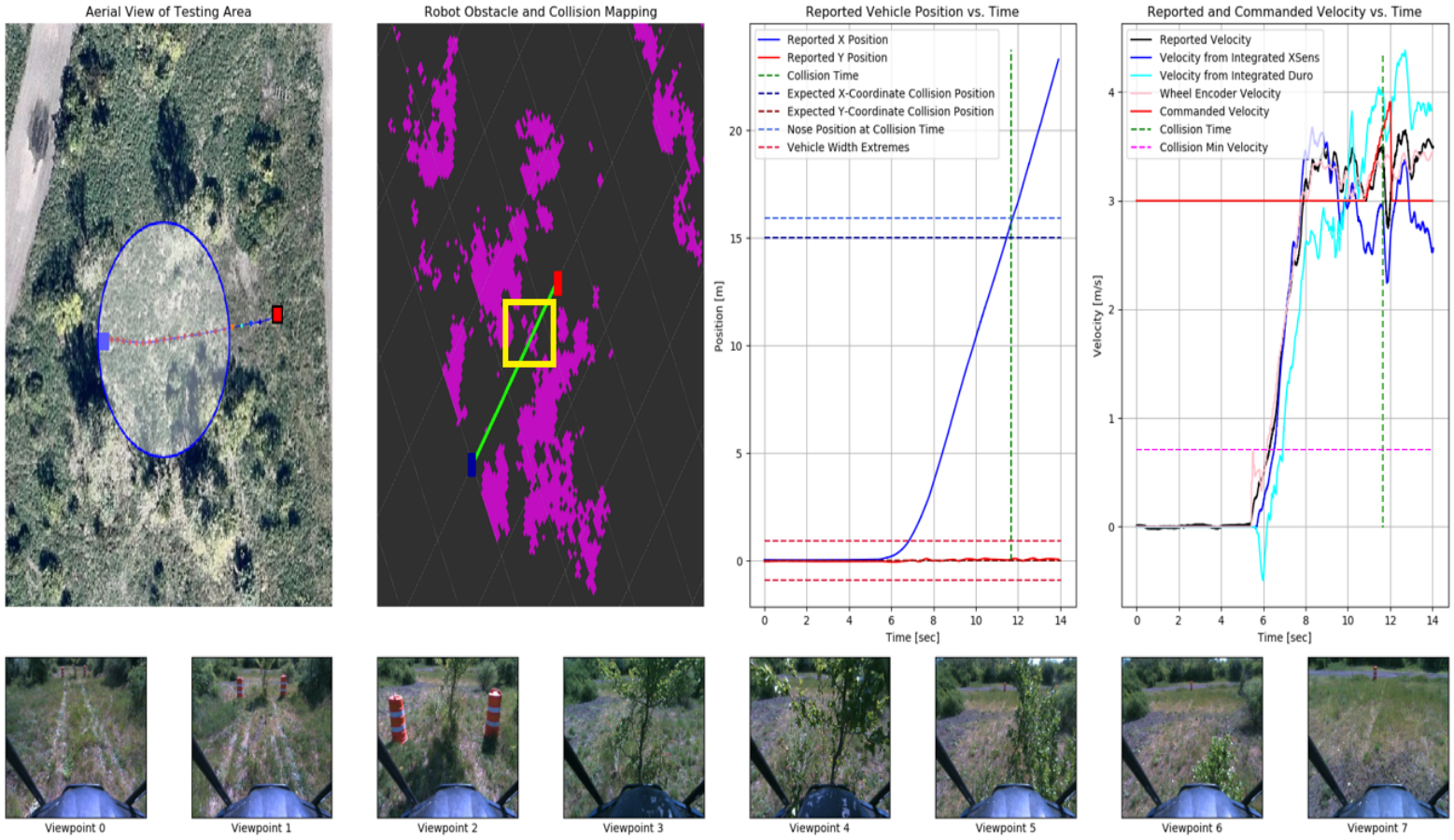


Figure 6.10: The third hardware experiment: a straight line trajectory through an embedded 81.8 [mm] tree.

The vehicle reached the target speed and x-coordinate position slightly before the collision time (0.48 [s]), which was likely due to the poor velocity tracking exhibited by the longitudinal controller at times 6-10 [s]. At around 12 [s], the vehicle's nose made contact with the stem of the tree, yielding an immediate slowdown that is reflected in the vehicle's reported velocity. The reported velocity loss from Super Odometry was 0.732 [m/s], which is lower than the expected loss of 0.758 [m/s]. Expectantly, integrating the IMU and GNSS output also directly reflects this slowdown in velocity. After colliding with the tree, the vehicle continued to roll until it came to a complete stop.

At the point of collision, the tree was overridden as shown in the point-of-view camera angles in Figure 6.10. However, after the vehicle traversed over the tree, the root structure of the tree returned it to an upright position. The final resting position and orientation of the tree after collision are shown in Figure 6.11.

Figure 6.11: A final location and orientation of the tree that participated in Experiment 3.

6.5.4 Discussion

The approach presented in this chapter utilizes existing low-computation-cost collision models to capture the effects of interactions with vegetative objects in the environment, enabling the design of a trajectory capable of overriding that vegetation. The approach leverages the robotic platform's onboard perception system to estimate the parameters of the vegetation models (e.g., tree diameter), which is then used to fit the collision model.

While there is inherent uncertainty in the measurement made by the onboard sensor suite, a conservative approximation (e.g., taking the largest-observed diameter in a time window) is utilized in this chapter to estimate the parameters of the vegetation model. Such approximations could lead to overly conservative behaviors when overriding vegetation of significant size, but allow for the system to model vegetation interactions solely through its onboard sensors. The generated trajectory captures the environmental interaction by modeling the expected loss of velocity experienced by the vehicle due to the interaction. This capability enables the vehicle to make real-time decisions about whether to override post-like environment objects and how to maneuver the wheeled robotic platform through a cluttered off-road environment.

The hardware trials were expected to show losses in kinetic energy due to the collision; however, surprisingly, no significant loss of kinetic energy was experienced during the collision with the posts. While the vehicle engine was in use during the impacts presented in this chapter, and the bull-bar had a compliant mount, the loss of energy associated with both experiments was expected to have resulted in a larger drop in the kinetic energy than was observed. The only experiment that reflects an observable drop in kinetic energy results in an equivalent loss of velocity of 0.732 [m/s] on collision, which is lower than the predicted value of 0.758 [m/s]. This final experiment suggests that a loss of kinetic energy can occur during interaction with the post, and that a dynamically feasible interaction trajectory should account for this loss to more accurately represent the real-world effects of the vegetation override interaction chosen by the robotic system.

6.6 Conclusions

The off-road operation of robotic systems remains a challenging area of research. It is not only clear from robotic-operated off-road vehicles, but also from human-operated off-road vehicles, that the unstructured nature of the terrain and any associated uncertainty yields a problem that is drastically different from on-road or on-trail driving. The presented work advances the state-of-the-art in off-road navigation by addressing vegetation interactions through the use of online hybrid dynamic optimization-based vehicle controllers that leverage classical vegetation override models. This understudied area of off-road driving has a wide range of applications and numerous areas for improvement to bring robotic platform operations more in line with those of expert human operators.

In this chapter, we present a trajectory optimization method that combines hybrid dynamics, a free-time formulation, and existing parameterized vegetation models from the off-road mobility and cross-country movement literature to override environmental

vegetation. Our method works by enforcing minimum velocity constraints, ensuring that collisions with the vegetation occur at a minimum override speed. These collisions occur at a designer-specified index of collision, but the designer need not select the time of collision, as that is handled directly by the solver.

While the collisions modeled in this chapter occur at a unique instance in time, future extensions should also look into addressing extended collisions with distributed objects, such as dense foliage. General improvements to the algorithm include changing the vehicle model to capture the frontal geometry and better representations of vegetative objects. Furthermore, exploring the implications of the designer-selected collision index hyperparameter and developing automated approaches to select that index optimally should be a priority.

In particular, we postulate that extending the study presented in this chapter to account for environmental aleatoric and epistemic uncertainty directly could provide a means for safer off-road behaviors. Of larger concern is that the provided vegetation models are themselves approximations of the vehicle-vegetation interaction, with factors that can be non-observable (e.g., root depth) or difficult to monitor (e.g., soil moisture) in real time. The authors believe that incorporating information-seeking behaviors to investigate model quality or including a measure of vehicle-vegetation interaction model confidence with respect to previously overridden vegetation should be a priority for the wider community.

Finally, we urge the off-road mobility community to continue developing and refining low-computational-cost vegetation interaction models and datasets for use by robotic systems. The algorithm presented in this paper leverages low-computational-cost representations of vegetation interactions to perform overrides; however, its extensibility to other types of vegetative and commonly encountered non-vegetative objects is dependent on these models. As an alternative to developing explicit vegetation-interaction models, the implicit models developed in learning-based approaches continue to show powerful advances in off-road vehicle control. However, many of these approaches are limited by the availability of high-fidelity off-road interaction data or representative simulated data.

Chapter 7

Interleaved Planning and Control for Vegetation Override

Our second supporting work on interleaving planning and control methods for vegetation override was presented at the 21st ISTVS International and 12th Asia-Pacific Regional Conference of the ISTVS in Yokohama, Japan.

Citation:

> **C. Noren**, B. Shirose, B. Vundurthy, S. Scherer, and M. Travers. "An Interaction-Aware Two-Level Robotic Planning and Control System for Vegetation Override," in *Proceedings of the 21st ISTVS International and 12th Asia-Pacific Regional Conference of the ISTVS*, Yokohama, Japan, October 2024

During off-road operations, mobile robotic platforms often encounter objects that influence the platform's route. As determining the outcome of an interaction with an object (e.g., overriding) is difficult, many robotic planners are designed to avoid interactions with all environmental objects. Yet, this object-adverse planning behavior is not reflected in the actions of expert human operators, who may interact with objects to find a viable path towards their goal. In this chapter, our objective is to enhance the performance of robotic traversals in off-road terrains by developing a planning paradigm that enables safe contact with environmental objects. Not only will this enable our robotic agents to traverse complex terrains, but we can then also leverage this planning paradigm to provide estimates of our task allocation costs in the task allocation layer. As shown in Figure 7.1, this capability provides a new context to the environmental information used in the task allocation layer.

Specifically, we design a two-level hierarchical planning and control system that couples a contact-informed regional motion planner with contact-constrained local nonlinear trajectory optimization techniques. The approach is demonstrated for classes of vegetative objects that are characterized by existing parameterized collision models from the terramechanics community. The top level of the hierarchy combines tree search with a set of override (velocity) constraints derived from these collision models during the search for a minimum-time trajectory towards the platform's goal. This minimum-time trajectory is then passed to the lower level of the architecture, which utilizes direct collocation and model predictive control techniques to ensure that the velocity constraints are enforced during the execution of the trajectory. The capabilities of the architecture are shown both in simulation and onboard a robotic platform, where vegetation is sensed, reasoned about, and then overridden

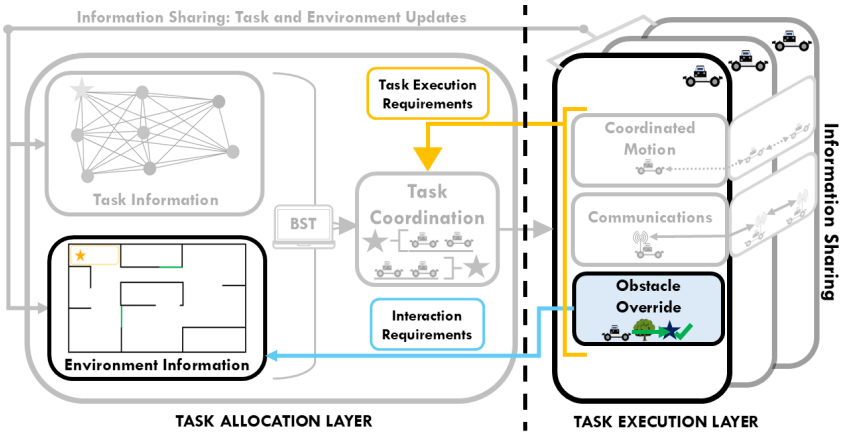


Figure 7.1: In this chapter, we focus on developing an interaction-aware planning system, specifically for vegetation override. This task execution layer capability can then be leveraged to provide more accurate task costs at the task allocation layer.

depending on the environment, platform, and object geometry.

7.1 Introduction

When operating in the field, off-road vehicles encounter a wide range of complex terrains. These terrains can vary greatly, demanding highly specialized and intelligent maneuvers for each situation. Autonomous systems are increasingly used to complement the decision-making of an expert driver and determine the "best" route (e.g., the fastest) through the terrain. One class of terrain that poses a unique challenge for off-road vehicle drivers is forest-like terrain, where vegetation is often encountered during operation. Rather than navigating around this vegetation, expert drivers often decide to override or "strike" specific pieces of vegetation to minimize the travel time to a destination.

However, determining when to strike objects in an off-road terrain is not a straightforward decision, even for expert human operators. To guide operators and logisticians, the off-road mobility and cross-country movement communities have produced models that consider both the geometric and inertial properties of a collision to determine time-optimal routes through complex environments [18]. Of particular note are the early experiments conducted at the U.S. Army Engineer Waterways Experiment Station (WES) (later, the U.S. Army Engineer Research and Development Center (ERDC)), which developed both data-driven and first-principles models of collision [16, 89]. These models were developed for both wheeled and tracked platforms and were characterized by extensive field tests [15, 16]. While the vehicles considered in these studies are often human-crewed, the broader adoption of off-road-capable robotic platforms has driven interest in utilizing these off-road mobility models in the design of robotic platforms [4, 17]. Vong et al. [17] considered these mobility models to design and analyze the capabilities of robotic systems, while Noren et al. [4] utilized these models in an online capacity to enable wheeled robotic platforms to override vegetative objects during off-road operations.

Unfortunately, the work in Noren et al. [4] is limited to interactions in which the collision is determined *a priori* necessary to reach the target goal. The limitation arises from the mathematical structure of the posed trajectory optimization problem, as both the point of collision and the goal state are provided as constraints in the underlying trajectory optimization problem. While the inclusion of such constraints is not uncommon in trajectory optimization works [83, 84, 85, 87, 88], utilizing this framework imposes a limited *myopic* view of the environment from the perspective of the robotic platform. Specifically, the trajectory optimization problem specifies *where* the platform must travel in the environment, regardless of whether a more efficient route exists.

To provide a holistic consideration of the environment, we propose to re-frame the problem space in a manner more consistent with that considered in robotic planning algorithms [76]. Many robotic planning problems focus on developing a plan that connects a start state and a goal state while avoiding objects in the operating environment. Classically, this is performed by first separating the operating environment into "free" and "occupied" space. In these classical approaches, the system designer assigns responsibility to the platform's perception system to determine what space is "occupied" and what space is "free." For problems where there is a clear separation between the "occupied" and "free" spaces, deterministic or sampling-based planning methods can often find a path between a start and end configuration [100]. However, in complex, cluttered environments (e.g., off-road environments), the free space may not contain a path connecting the start and end configurations [10].

The planning problem where no object-interaction-free path exists is known as the "navigation among movable obstacles" (NAMO) problem. In this particular problem, the goal is to enable the planning entity to restructure or rearrange the environment to meet its objectives [12]. Early approaches to the NAMO problem performed a state-space decomposition to identify manipulation points, and then conducted a heuristic search over the free-space configuration components to mitigate the complexity of multi-object planning [13]. Recently, Saxena has demonstrated that connecting the problem to the multi-agent pathfinding domain enables a decomposition of the problem into a configuration-search step and a physics simulation step [101, 102]. These planning approaches provide a framework for the planning problem considered in this thesis [10], but do not directly assess the kinematics of the planning agent, which were shown in Noren et al. [4] to be essential to account for the effect of the contact on the agent's motion.

Approaches that account for the kinematics commonly combine planning and boundary value control problems. An example combination includes combining the Linear Quadratic Regulator (LQR) optimal control strategy with Rapidly Exploring Random Trees (RRT) [100]. Recent work in this area has looked at combining LQR controllers with Control Barrier Functions (CBF) and RRT planners. Still, these works do not consider planning problems without free-space paths (i.e., consider only object-avoidance objectives) [103, 104].

In this chapter, we develop an interleaved interaction-aware robotic planning and control system for vegetation override, combining the work presented in Noren et al. [4] with classical sampling-based motion planning algorithms to generate interaction-aware plans in complex environments. This two-level planning system incorporates the collision models developed by the terramechanics community during the search process, modeling collisions explicitly during the connections between two states of the configuration space. Given

the result of this collision, a further determination may be made on whether or not the in-collision path should be used during operation.

7.2 Method (Algorithm) Overview

A brief definition of the problem is provided to provide context for the off-road navigation problem considered in this chapter. Following this problem definition, the specific implementation of a sampling-based planner, RRT*, is described, where the connecting mechanism between points in the configuration space is described at a high level. Finally, the last section presents an attenuated discussion of the controller and collision models used in Chapter 6 before concretizing this controller as the specific connector used in the proposed interleaved planning and control system.

7.2.1 Problem Definition

Consider a robotic platform represented by a state space, $\mathcal{X} \in \mathbb{R}^n$, a control space, $\mathcal{U} \in \mathbb{R}^m$, and a motion profile which is described by the $\mathcal{C}^2$ smooth continuous-time dynamics expression

$$\dot{x}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)). \tag{7.1}$$

In this chapter, the variable t represents time. Specific instants in time will be indicated via a subscript (e.g., t_0, t_g may represent the start and end (goal) times of the trajectory). Elements of the state and control spaces, $\mathbf{x}(t) \in \mathcal{X}$ and $\mathbf{u}(t) \in \mathcal{U}$ respectively, represent the state and control of a robotic platform at a specific time.

The aim of our method is to determine a set of states $\mathbf{x} : [t_0, t_g] \to \mathcal{X}$ and control inputs $\mathbf{u} : [t_0, t_g) \to \mathcal{U}$ to minimize the total cost associated with a trajectory. This cost, $J(t, \mathbf{x}(t), \mathbf{u}(t))$, may be evaluated along a specific trajectory

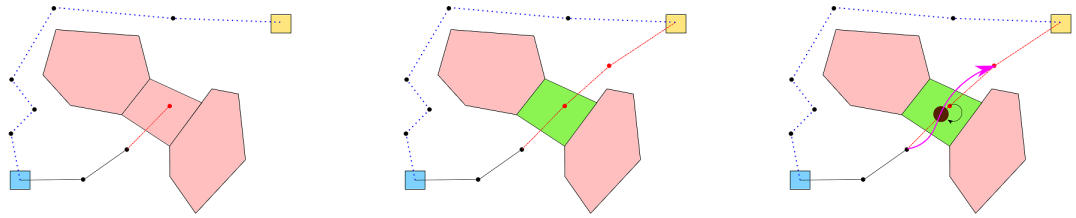
$$J\big(t_0, t_g, \mathbf{x}(t), \mathbf{u}(t)\big) = J_f\big(\mathbf{x}(t_0), \mathbf{x}(t_g)\big) + \int_{t_0}^{t_g} w\big(\mathbf{x}(\tau), \mathbf{u}(\tau)\big) d\tau. \tag{7.2}$$

As described in Kelly [88], the boundary cost term, $J_f(\cdot)$, and the integral cost term, $w(\cdot)$, may be selected by a designer to optimize specific behaviors of a system. We address a common task found in many off-road applications: minimum-time traversals. As such, the cost function described in (7.2) can be collapsed to $J\big(t_0, t_g, \mathbf{x}(t), \mathbf{u}(t)\big) = t_g$.

Given configuration space $\mathcal{X}$, consider a decomposition of this configuration space into an *object-free* space, $\mathcal{X}_{free}$ and *object-occupied* space, $\mathcal{X}_{obj}$. Assume that the space is completely decomposable into these subspaces: $\mathcal{X} = \mathcal{X}_{free} \cup \mathcal{X}_{obj}$. This decomposition is

commonly used in robotic planning (e.g., [100]) to describe the states that a robot may obtain during operation.

However, this standard approach yields a paradigm that neglects one of the unique properties of off-road operations discussed in Noren et al. [4]: *that off-road operations often require interactions with objects.* While "interaction" may be defined in many different ways, given the mechanical design of most off-road vehicles, "interacting" with the environment implies that the vehicle's chassis physically strikes the object. A further decomposition of $\mathcal{X}_{obj}$ is required to capture this aspect of off-road driving. That is, by decomposing $\mathcal{X}_{obj}$ into two additional subspaces: the space of "interactable" objects, $\mathcal{X}_{io}$ and "noninteractable" objects (obstacles), $\mathcal{X}_{nio}$, additional collision logic may be incorporated in the planning and control layer. In this chapter, we assume that given sufficient characterization, all elements of $\mathcal{X}_{obj}$ are determinable members of $\mathcal{X}_{io}$ or $\mathcal{X}_{nio}$, thus implying that $\mathcal{X}_{obj} = \mathcal{X}_{io} \cup \mathcal{X}_{nio}$. In reality, such a determination may be challenging given the perception systems available onboard a robotic platform. Thus, a further *unknown-interactable object* class $\mathcal{X}_{uio}$ may also be required. A conservative yet straightforward approach to addressing this problem would be to incorporate objects of this class into the $\mathcal{X}_{nio}$ class.



(a) RRT exploration, only $\mathcal{X}_{obj}$ (b) RRT exploration, both $\mathcal{X}_{io}$ and $\mathcal{X}_{nio}$ (c) Interaction-aware RRT exploration

Figure 7.2: Consider a motion planning problem starting at the blue square and moving through a two-dimensional configuration space to the end state at the gold square. In classical sampling-based planning approaches, which treat any object-occupied area of the configuration space as an obstacle ($\mathcal{X}_{obj} = \mathcal{X}_{nio}$), all obstacles must be avoided in order to reach the goal point. However, if this obstacle-avoidance constraint is relaxed for certain classes of obstacles, i.e., $\mathcal{X}_{obj} = \mathcal{X}_{io} \cup \mathcal{X}_{nio}$, shorter paths (e.g., through the green zone representing a subset of $\mathcal{X}_{io}$), may be found. Finally, it is often insufficient to solely sample through the space of $\mathcal{X}_{io}$ to generate a physically-realizable trajectory. Instead, by incorporating connectors between sampled states that enforce the dynamical constraints of the system (e.g., collision dynamics as in Noren et al. [4]), a physically realizable trajectory may be established between two points in the graph.

7.2.2 Asymptotically Optimal Global Planning

Given the natural complexity of outdoor environments, few environments can be represented through an explicit enumeration of objects found in the environment ($\mathcal{X}_{obj}$). An approach that avoids this complication is sampling-based planning methods [100], which instead rely on a collision-checking module to characterize whether a trajectory is in collision with an environmental object. In traditional sampling-based planning approaches, points are drawn pseudo-randomly from the configuration space of a robotic platform to build a graph connecting an initial (starting) configuration to a desired (goal) configuration. A simplified description of this process has the algorithm drawing different points from the configuration space, checking to ensure membership in the obstacle-free subspace ($\mathcal{X}_{free}$), and then connecting this newly drawn point to a subset of prior-sampled points depending on the output of the collision-checking module. A depiction of this simplified approach is shown in Figure 7.2a, where it can be seen that any point sampled from the space $\mathcal{X}_{obj}$ is rejected (in red).

The traversal distance and traversal time (assuming a constant velocity motion) may then be approximated by computing the cost-to-go from each point in the graph to the goal state.

7.2.3 Collision-Checking Module

Instead of the behaviors shown in Figure 7.2a, we seek to take advantage of environmental information to perform the object decomposition described in Section 7.2.1. This behavior is incorporated into the collision-checking module.

Definition of "Interactable" Objects

Unfortunately, even with the above problem decompositions, it is not clear *how* the obstacle-occupied space $\mathcal{X}_{obj}$ should be decomposed for an individual environment and terrain. Expectantly, interacting with certain objects (e.g., a concrete barrier) incorrectly may have disastrous consequences for a robotic platform. In this context, "incorrectly" is defined on a per-object basis (e.g., for the aforementioned concrete barrier, approaching at near-zero speeds may be acceptable) through interaction rules. Thus, for specific object classifications (e.g., single-standing tree, array of trees, etc.), an object may exist in either the $\mathcal{X}_{io}$ subspace or the $\mathcal{X}_{nio}$ subspace depending on the rules of interaction. At the same time, rules can be defined as a series of logical statements (e.g., *A robot may never interact with a concrete barrier*); such approaches do not scale with complex worlds. Instead, we utilize physics-inspired empirical models to define the interaction rules on a procedural basis. For this

work, these empirical models take the form of vegetation-override models.

1968 Blackmon Vegetation Override Model (from Chapter 6)

For completeness, we recall our discussion of the model presented in [16] in Chapter 6. The model presented in Blackmon and Randolph [16] was derived from a series of vegetation override tests conducted on different vegetation types in various environments. Blackmon and Randolph [16] provide unique regressions for the force required and energy expenditure to override these different types of vegetation, including singular coniferous and hardwood trees, arrays of multiple trees struck in unison, and "clumps" of bamboo. From continuous measurements of pushbar force, drivetrain metrics, distance traveled, time, measurements of the impacted trees, and characterizations of the aftermath of the collision, Blackmon and Randolph [16] construct a model primarily parameterized by the geometry of individual or multiple trees (for example, the radius of a tree) or the clump diameter.

In this study, the authors consider only experiments that require the override of a single tree or post. While Blackmon and Randolph [16] provide additional override models, this simplification to a single class of vegetation was drawn from limitations in the perception of the necessary characterizing features for arrays of trees. Thus, the methodologies presented herein are not limited in scope to single-standing trees, aside from the limits discussed in the original [16] manuscript itself. Given this simplification, equations B10-B12 from [16], which describe the force and work required to override a single standing tree, are of particular interest. These equations largely take the form

$$F_h = K_f d_s^3, \tag{7.3}$$

and

$$W = K_w d_s^3, \tag{7.4}$$

where F_h is the horizontal pushbar force, W is the work required to fail a single standing tree, d_s is the stem diameter, and K_f and K_w are constants that are dependent on vehicle geometry (e.g., pushbar height).

The measure of work produced from Blackmon's models may then be combined with additional vehicle information (e.g., the operating mass) in order to generate a suitable override velocity, v_{over}, for a sensed piece of vegetation. This override velocity is then incorporated as a velocity target for a trajectory optimization problem defined on the vehicle's motion, as further detailed in [4]. The equivalent relations to determine this v_{over} may be calculated in the manner discussed in Mason et al. [89].

The vegetation override models developed by the off-road mobility and cross-country

movement communities, including those by Blackmon and Randolph [16] and Mason et al. [89], abstract complex collision interactions into useful, low-computational-cost approximations. These approximations generally characterize the required force, work, or velocity needed for a vehicle to overcome a subset of vegetation, given some parameterization of the vehicle and its environment. Furthermore, the models form an actionable criterion as the basis of an interaction rule given the reachability of the override velocity from the robotic platform's current state. By evaluating whether or not the override velocity may be reached, a robotic platform can determine if a specific vegetative object exists in the class of objects defined by $\mathcal{X}_{io}$ (i.e., the necessary velocity is reachable) or $\mathcal{X}_{nio}$ (i.e., the necessary velocity is not reachable).

7.2.4 Interaction-Aware Connectors

Given the above decomposition, incorporating this collision information into the connector is necessary to ensure a physically realizable interaction. Ensuring such an interaction is realizable is possible by enforcing differential constraints between nodes in the search graph. Unfortunately, this particular requirement yields a planning problem of high computational complexity [76], especially given the hybrid nature of the modeled contact dynamics.

In our prior work ([4], Chapter 6), a two-point boundary value problem was specified that utilizes contact models to account for the hybrid dynamics experienced during collisions in robotic platform control. This approach can be extended into our prescribed search algorithm by running the two-point boundary value problem between all states, similar to the work presented in Xie et al. [105]. Unfortunately, such an approach would quickly become computationally intractable; thus, we move towards an approach where we selectively call the controller, as in Noren et al. [4], to evaluate only specific segments of the connected graph. The specifically evaluated segments are those where an object from class $\mathcal{X}_{io}$ is encountered, such as in Figure 7.2b. This "interleaved" evaluation updates both the trajectory taken between two points and the cost of the traversal.

7.3 Method Analysis

We first consider the situation where a vehicle must determine whether to override a piece of vegetation. Such a determination in most circumstances may be challenging to compute. Still, in certain simple scenarios, a pure kinematic analysis may be used to determine whether to override a piece of vegetation.

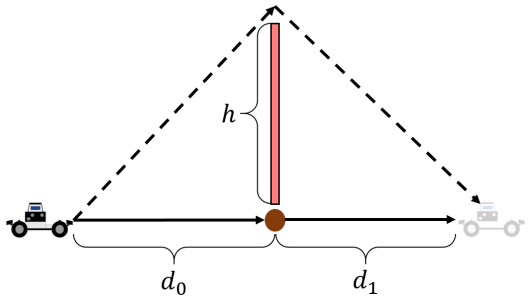


Figure 7.3: Top view of a simplified scenario in which a robotic platform must determine whether to strike a piece of vegetation (in brown) to travel a more direct route, or avoid interacting with any objects in the scene (brown or red) to reach a goal location.

Kinematic Analytical Formulation

Consider the scenario depicted in Figure 7.3, where a robotic platform must either choose to override a piece of vegetation (in brown) or avoid the vegetation and an adjacent obstacle (in red). Assuming that the vehicle initially travels at maximum velocity v_{max}, and has a maximum constant acceleration that it can sustain, a_{max}, it is the goal of the combined planning and control system to reach the position $d_0 + d_1$ [units] directly ahead of the platform in the shortest time.

By inspection, it is clear that if the vehicle is to avoid all objects in the scenario, the time taken to travel the entire distance is

$$t_{avoid} = \frac{1}{v_{max}}(\sqrt{d_0^2 + h^2} + \sqrt{d_1^2 + h^2}). \tag{7.5}$$

As this travel time represents the fastest time the platform can travel between the initial position and the desired ending location, the determination of whether or not the vehicle should override the post relies solely on whether or not the time required to reach the end location, which travels through the post, t_{strike}, is less than t_{avoid}. Note that, according to the discussion in Noren et al. [4], the velocity needed to override the post must still be reachable from the initial configuration to have a valid trajectory. For this example, we assume this to be true (e.g., $v_{max} > v_{col}$).

Given the geometry of the above problem and the collision models discussed in 7.2, we can bifurcate the trajectory into two distinct subsegments. The first segment represents the pre-collision motion, and the second segment represents the post-collision motion. As such,

the total travel time needed to override the post with respect to these two sub-segments is

$$t_{strike} = t_0 + t_1, \tag{7.6}$$

where t_0 represents the travel time on sub-segment 1 and t_1 represents the travel time on sub-segment 2.

Per the arguments above, the travel time on sub-segment 0 is determined to be

$$t_0 = \frac{d_0}{v_{max}}, \tag{7.7}$$

which means that to yield a faster time than t_{avoid}, the constraint that

$$t_1 < \frac{1}{v_{max}}(\sqrt{d_0^2 + h^2} + \sqrt{d_1^2 + h^2}) - \frac{d_0}{v_{max}}, \tag{7.8}$$

must hold.

The act of overriding the post results in the platform losing velocity to a value given by $v_{drop} \geq 0$. Depending on the value of v_{drop}, the platform can either continue to accelerate until the end of sub-segment 2 or accelerate to v_{max} and cruise at this maximum speed until the end of sub-segment 2. This disparity between behaviors can be captured by computing the distance d_1' needed to reach the maximum speed.

$$d_1' = \frac{v_{max}^2 - v_{drop}^2}{2a_{max}} \tag{7.9}$$

With the help of d_1', the travel time t_1 on sub-segment 2 and, subsequently, the constraint to decide between avoiding and striking the piece of vegetation via (7.8) can be determined as follows:

$$t_1 = \begin{cases} \frac{v_{max} - v_{drop}}{a_{max}} + \frac{d_1 - d_1'}{v_{max}}, & \text{if } d_1' \leq d_1, \\ \frac{2d_1}{v_{drop} + \sqrt{v_{drop}^2 + 2a_{max}d_1}}, & \text{otherwise.} \end{cases} \tag{7.10}$$

7.4 Numerical Simulation

The proposed method was tested in multiple simulated trials representing different configurations. In all simulated trials, a vehicle described by the dynamics presented in 7.4.1 and characterized by parameters in A.1 was initialized at a point, x_{ic}, and was required to reach a given configuration, x_g. In all tests, the vehicle starts at rest with a heading of zero degrees (aligned with the x-axis).

One key limitation observed in Noren et al. [4] was that an interactable object needed to

be selected for override before understanding the overall effect the object would have on the robot's traversal. This posed challenge arises from the associated "myopia" of local control methods, which utilize only near-field objects and a limited preview horizon to determine the platform's control. Such controllers include the controller proposed in [4]. This myopia appears in at least two clear scenarios: 1) where a vehicle must select between striking multiple objects; and 2) where a vehicle must account for multiple object interactions. We explore these cases in simulation in the next section.

7.4.1 Vehicle Modeling

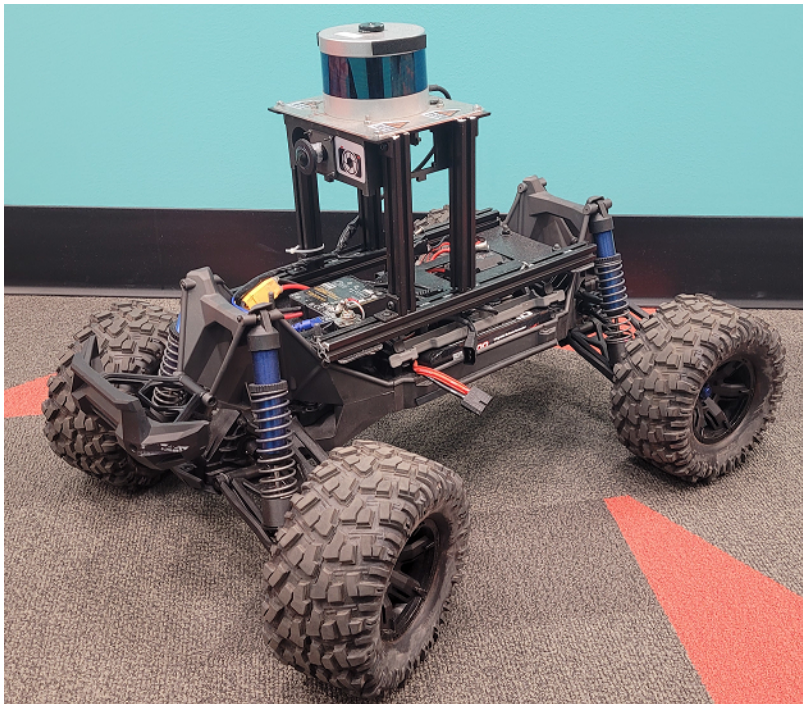


Figure 7.4: A modified small uncrewed ground vehicle with a similar configuration to the one utilized in this chapter.

The proposed architecture is extensible to various dynamical systems; however, to demonstrate its application, this paper considers a platform similar to the one shown in Figure 7.4. The platform is a modified vehicle from Traxxas and is discussed in further detail in Appendix A. The vehicle dynamics (7.1) were modeled as a nonlinear bicycle with the state: $\mathbf{x} = [p_x, p_y, \psi, v]$. The elements of this state vector are the vehicle's x-position, y-position, heading, and velocity, respectively. The available vehicle controls included velocity and steering angle: $\mathbf{u} = [v, \delta]$. We model the continuous-time vehicle dynamics as

$$\begin{aligned}\dot{p}_x &= v * \cos\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right),\\ \dot{p}_y &= v * \sin\left(\psi + \operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right),\\ \dot{\psi} &= \frac{v}{L}\cos\left(\operatorname{atan}\left(\frac{L_f}{L * \delta}\right)\right) * \tan(\delta),\end{aligned} \tag{7.11}$$

and the characterizing platform parameters are listed in Appendix A in Table A.1.

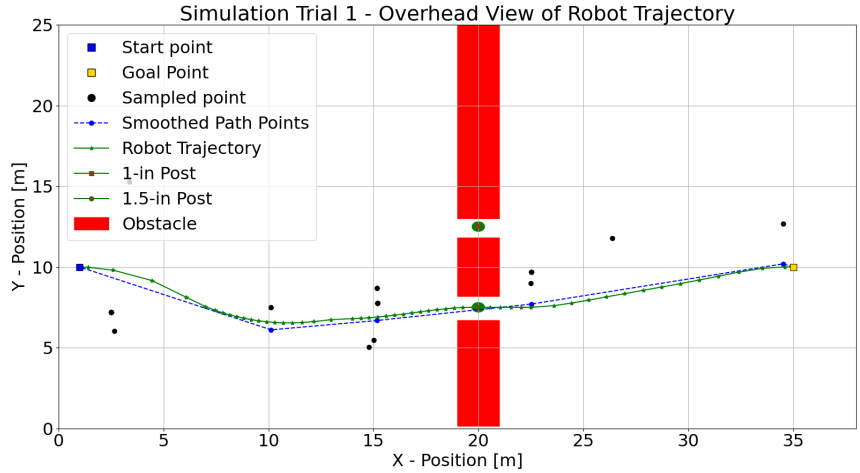
7.4.2 Scenario 1 - Obstacle Selection

The first considered scenario involved having the robotic agent determine between two paths to reach a goal. Assume the vehicle is bounded in its y-position between 0 and 25. The first path would require the platform to interact with a 38.1 [mm] (1.5 [in]) diameter vegetative object, and the second path would require the platform to interact with a 25.4 [mm] (1.0 [in]) diameter vegetative object. In both override scenarios, Blackmon and Randolph [16]'s model was utilized as discussed in Section 7.2.3.

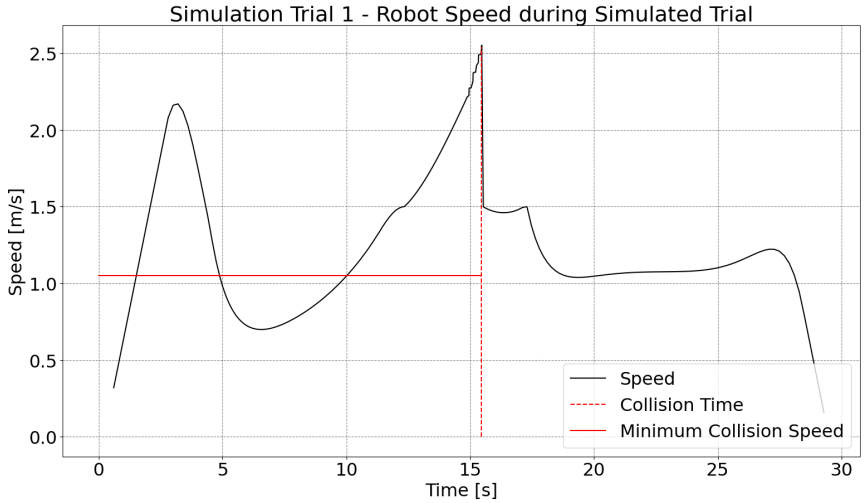
The example override for this scenario is shown in Figure 7.5. The starting configuration of the robotic platform is located at the center of the blue marker (position: (1, 10) [m]). Its initial heading is along the abscissa, and it starts with zero velocity. The desired ending configuration (shown as a gold square) is placed 34 [m] in front of the robot (position: (35, 10) [m]), with a goal heading aligned with the abscissa and a goal velocity of 1.5 [m/s]. Note that a line between the initial and desired spatial configuration generates a symmetric object field outside of the vegetative objects. Such a configuration yields a choice of trajectory in two distinct homotopy classes. Both objects appear 19 [m] ahead of the vehicle, offset exactly 2.5 [m] above (38.1 [mm]) and below (25.4 [mm]) the starting configuration.

While a biased sampling schema exists that could lead to selecting a route through the top object, using an unbiased sampler yielded paths that consistently passed through the lower object. Given the analysis provided in Section 7.3, it is clear that even with a perfect sampling schema, the geometric symmetry of the problem should bias the robotic platform to travel through the lower object. Furthermore, that same analysis indicates that a geometric configuration of vegetative objects exists in this schema, which would lead the platform to override the 38.1 [mm] object because a 25.4 [mm] object would require too much time to deviate towards (i.e., t_0 may be higher).

Finally, considering the velocity diagrams, the vehicle initially maneuvers to align with the vegetative object and then significantly increases its velocity to override the post. The depicted initial speed up (around 2.5 [s]) and slowdown (around 7 [s]) correlate to large changes in heading as the vehicle bounds through different curves at (x position) 2.5 [m] and



(a) Trial 1 - Simulated 25.4 mm post collision

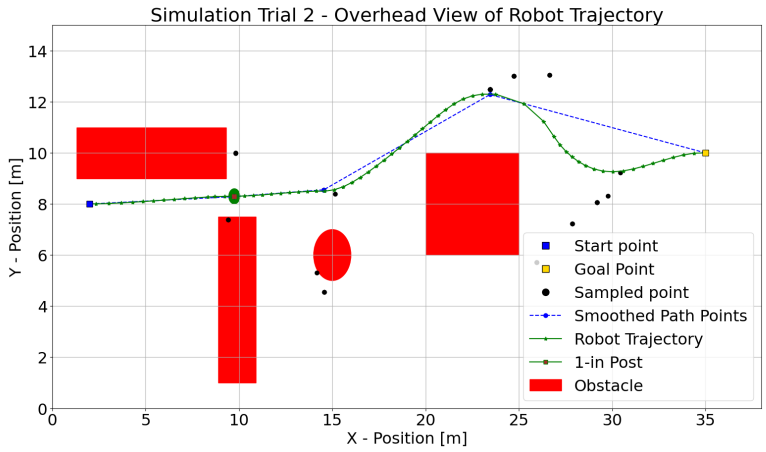


(b) Trial 1 - Velocity vs. Time with simulated post

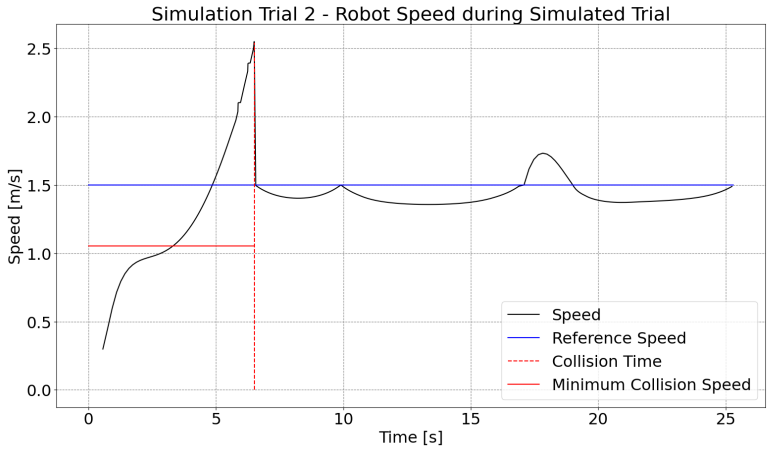
Figure 7.5: First simulated trial demonstrating both an object strike and avoidance.

(x position) 10.0 [m], respectively. The vehicle additionally accelerates into the vegetative object before maneuvering towards the goal. This acceleration was achieved by biasing the trajectory optimization to include a reference velocity (in this case, a travel speed of 1.5 [m] set as a boundary condition at the end of the segment) with the expected required override velocity computed with Blackmon's model.

7.4.3 Scenario 2 - Multiple Obstacle Avoidance



(a) Trial 2 - Simulated 25.4 mm post collision



(b) Trial 2 - Velocity vs. Time with simulated post

Figure 7.6: Second Simulated Trial Demonstrating both an Object Strike and Avoidance.

The final simulated experiment demonstrates another extension of the work presented in [4]: the ability to enable an obstacle override alongside other obstacle avoidance tasks. In [4], a single obstacle was overridden to reach a desired goal state, but environments are often much more complicated, with a single override affecting the entire resulting path. For this depicted scenario, the vehicle starts at the blue square (position (2,8)) with the same initial heading and velocity constraints as in the previous simulation and heads towards a similar goal configuration. A single 25.4 [mm] object is placed in front of the robotic

platform at position (10, 8), and Blackmon's model is again used to determine the override velocity.

The scenario presented in Figure 7.6 requires the vehicle to make a decision: whether to travel out of an initial confined area via an open corridor, which takes it towards the goal in the x-direction but away from the goal in the y-direction. Following this set of obstacles is another larger object (starting at the x position 20 [m]) that requires the platform to deviate from its initial heading. As shown in Figure 7.6, the platform biases its search away from sampled points that would take it away from the desired location or require large deviations from the goal point. Note that in this scenario, the vehicle's speed is penalized if it deviates from the provided reference speed (again, 1.5 [m/s]). The velocity graph depicts a behavior in which the vehicle speeds up to strike the object (around 7 [s]) and then maintains this reference velocity to the goal state.

7.5 Vegetation Override Hardware Results

This section details two hardware experiments, both utilizing the Blackmon and Randolph [16] model in Section 7.2 and a platform similar to the one in Figure 7.4. The vehicle was commanded to travel at a nominal 1.5 [m/s] in both experiments. A model predictive control algorithm, specifically the iterative Linear Quadratic Regulator (iLQR), was implemented to track a trajectory similar to those shown in the simulation. The results of these tests are shown in Figure 7.7 and Figure 7.8.

7.5.1 Trial 1 - Obstacle Selection Test

The first trial scenario demonstrated on robotic hardware was an obstacle override selection test. This test was conducted with an environment configuration similar to the simulation test shown in Section 7.4.2. The object the robotic platform interacted with was a 25.4 [mm] pine dowel rod. The pine dowel rod was embedded into the soil by hand.

The executed trajectory is shown in Figure 7.7. The vehicle started at a position of (1, 10) [m] and reached a target position at (10, 25) [m]. The robotic platform successfully avoided all obstacles (in red) while overriding the dowel rod placed at position (20, 9.25) [m]. The vehicle reached a collision velocity of around 1.492 [m/s] at the point of collision, which is higher than the required override velocity calculated in Blackmon's model. After the collision, the post was pulled from the ground and fell onto the terrain surface. The robotic platform was not damaged during the collision. Several viewpoints taken by the robotic platform along the trajectory are also shown in Figure 7.7.

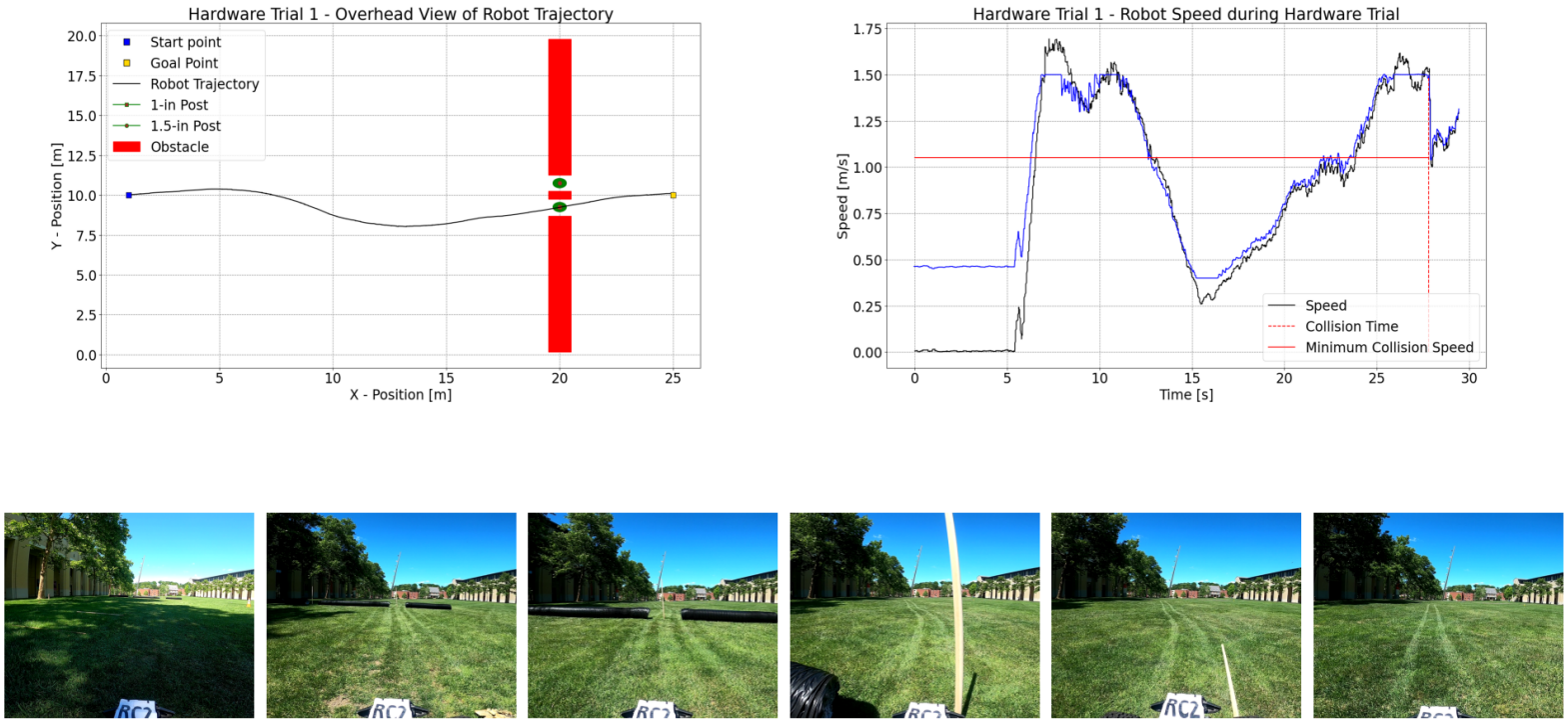


Figure 7.7: Hardware Trail 1 - Obstacle Selection Test.

7.5.2 Trial 2 - Multiple Obstacle Consideration

The second trial scenario was a multiple-obstacle consideration test. In this test, the platform would not only need to perform an override similar to the test shown in the simulations demonstrated in Section 7.4.3, but also to avoid additional zones that contained objects the platform could not interact with. These zones are depicted in red in Figure 7.8.

In this particular test, a 25.4 [mm] diameter post was embedded in the ground. The robotic platform started at position (2, 8) [m], the post was placed at position (9.0, 8.75) [m], and the desired location (goal) was determined to be (35, 12.8) [m]. As in the first hardware experiment, the vehicle reached a speed of 1.490 [m/s] at the point of collision. After the collision, the post was again pulled from the ground and came to rest on the surface. As in the first trial, the platform was not damaged.

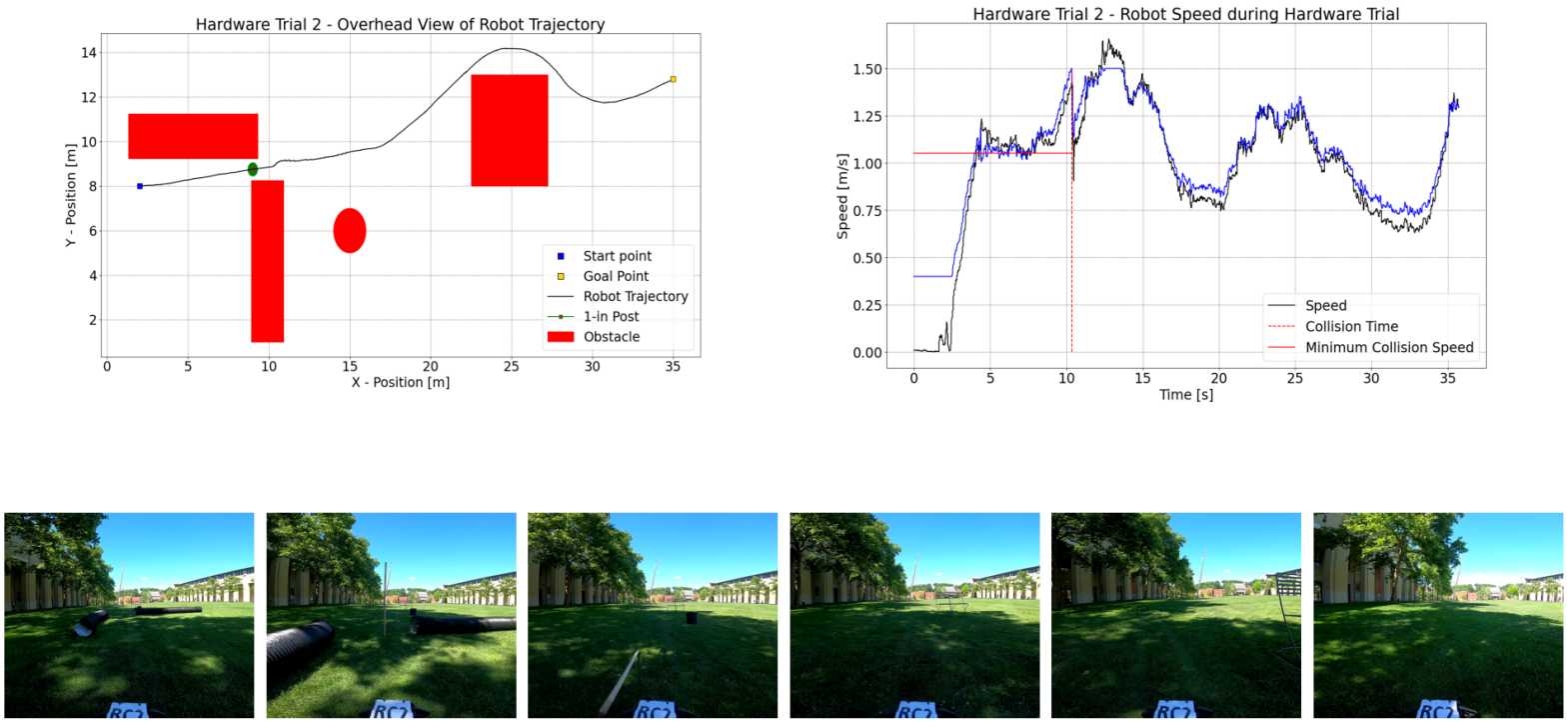


Figure 7.8: Hardware Trail 2 - Multiple Obstacle Consideration.

7.6 Conclusions

In this chapter, we presented a two-level motion planning architecture that combined information from existing parameterized vegetation models from the off-road mobility and cross-country mobility communities, controls from hybrid trajectory optimization, and sampling-based planning methods to support robotic decision making in off-road environments. The method extends the current state-of-the-art by incorporating collision-aware hybrid dynamics as an intelligent connector during the planner's graph construction.

The work that most closely matches the proposed work is presented in Xie et al. [105]. Xie combines Batch Informed Trees (BIT*) search with a two-point boundary value problem solver to perform optimal planning in time. However, Xie's work also focused on obstacle avoidance and thus does not account for the hybrid dynamics encountered during environmental interactions. However, we draw inspiration from Xie's work to propose an extension to our method: namely, investigating other sampling-based planning methods and comparing their computational performance on a robotic system.

While our work in this chapter also addressed many challenges identified in Noren et al. [4], concerns identified therein still require further investigation. Specifically, identifying additional collision models (e.g., for distributed vegetation such as dense foliage) and adapting the solver hyperparameters (e.g., for the collision index) should be considered to help minimize connector solve times.

Chapter 8

Conclusion

As robotics continues to mature as a discipline, we see an ever-evolving demand for increased robotic capabilities (for both single-agent and multi-agent systems) arising from complicated real-world challenges. Interestingly, some of the most public showings of robotic capabilities, the DARPA Challenges, reflect this progression. Directly evident is how early single-agent autonomous driving in the DARPA Grand Challenge and the DARPA Urban Challenge, which laid the foundation for wheeled robotic mobility, led to the DARPA RACER Challenge and much of the work in Part III. Alongside this, we also see how the DARPA Subterranean Challenge laid the foundation for multi-agent system development and created a set of baselines for much of the work in Part II. Undoubtedly, this progression of ideas in field robotics has shaped this thesis. Specifically, the framework presented herein advances the state of the art in the deployment of tightly coupled or coordinated robotic systems, providing a capability baseline for robotic convoying systems.

The key challenge that persists throughout this thesis is the inherent coupling and dependencies imposed by the unstructured environment, mission tasks, and the other robotic agents that comprise the multi-agent robotic system. This thesis advances coordinated robotic system operations by presenting a framework that abstracts components of the problem space into succinct operational requirements. We then embedded these requirements into the different functional and information layers of the framework. The approach addresses each of the aforementioned types of coupling, and the effectiveness of our technical contributions is demonstrated in both simulation and on robotic hardware.

We conclude this thesis by first summarizing the technical contributions of this work in Section 8.1. We organize this summary around the three major components that we claim are vital to improving the performance of multi-agent systems in unstructured terrains with tasks that have coupled execution requirements. These components align with our thesis

statement:

Thesis Statement:

The performance of multi-agent systems in unstructured environments with tasks that have coupled execution requirements can be improved by:

1. forming representative task abstractions,
2. responding to evolving mission information, and
3. reasoning about environmental interactability.

Following the summary in Section 8.1, we denote some limitations of our framework in Section 8.2. Specifically, we speak to challenges where forming a representative task abstraction may be difficult, indicating that our bi-level framework likely falls on a spectrum of framework designs that address both task allocation and task execution coupling. Following this, we also highlight the limitations of our system in addressing communication challenges and dynamism, as well as the influences of environmental interactability. Finally, in Section 8.3, we offer future research directions that build and refine the framework presented in this thesis. These directions not only highlight the limitations discussed in Section 8.2 but also provide insight into improving the individual components of the framework.

8.1 Thesis Summary

In Part I, we investigated operations that exhibit both task allocation and task execution coupling. In particular, we considered an automated robotic convoying application in unstructured terrains. We first introduced a low-level formation controller in Chapter 2, where we showed that by relaxing strict formation-keeping constraints, a distributed control schema can reflexively react to environmental and inter-convoy disturbances. This convoy controller not only allowed us to decrease inter-agent convoy spacing by over 50%, but more importantly, demonstrated an effective functional abstraction of the convoying task. In Chapter 3, we then leveraged that functional abstraction by introducing a variant of the vehicle routing problem with multiple synchronizations that included a set of "convoy constraints" (VRPMS-CC). To the author's knowledge, this is the first work to present convoy operations in the context of synchronized routing problems. The advantage of our approach is that a solution to the VRPMS-CC yields an optimal set of convoy assignments that our lower-level formation-keeping controller can implement. This bifurcation of responsibilities along the layers of the hierarchical framework arose from the abstraction

of lower-level task execution requirements as explicit constraints in the higher-level task allocation schema.

At the end of Part I, we noted how the VRPMS-CC could be affected by imperfect environmental information. Our solution leveraged the onboard sensing and motion planning systems of the robotic agents to formulate a dynamic vehicle routing problem. Just as in Part I, we leveraged a central optimizer to solve the dynamic vehicle routing problem and recover the optimal routes and assignments of each agent. This dynamic vehicle routing problem procedure demonstrates an information coupling between the agents and the central optimizer. As many of the unstructured environments we are interested in do not have available communication infrastructure that agents can use, in Part II, we developed a mobile ad hoc network construction technique that deploys nodes in response to low radio signal strength. We first outline this algorithm in Chapter 4, where we use a maximin metric to construct a spanning tree on the communications graph. This metric ensures that agents always maintain a minimum communications signal strength, supporting the transfer of information between assets in the multi-agent team. Furthermore, in Chapter 5, we introduce and integrate a communications recovery behavior to ensure unexpected system failures do not compromise the integrity of the communication system. We leveraged this communication system during the execution of the Dynamic VRPMS-CC in Part I, demonstrating the extensibility of the approach first outlined in Part I to dynamic routing tasks.

Finally, in Part III, we considered how environment-agent coupling affects both the task allocation and task execution layers of our framework. In particular, we relaxed traditional notions of obstacle avoidance to enable robotic agents to interact with a subset of environmental objects. We used this idea to introduce an "override" controller in Chapter 6, where a robotic agent leveraged our override controller to override vegetation in a natural environment. This override mechanism is motivated by classical terramechanics and off-road mobility models, such as the data-driven works in Blackmon and Randolph [16] and analytical models in Mason et al. [89], and demonstrates the need to represent the physical process underpinning the nature of the task. Specifically, the override controller imposes a pathwise constraint at the point of collision. To successfully override the object, the agent must satisfy this pathwise constraint. In Chapter 6, our constraint took the form of a minimum required override velocity "v_{over}" conditioned on the object's size. We then introduced an interleaved motion planning and control algorithm in Chapter 7 that selectively evaluated our interaction controller to "override" objects in the environment. This relaxation improved the system's reachability in complex, cluttered, and unstructured ("off-road") terrains, providing better estimates of task feasibility and optimality at the

task allocation level of the framework.

This thesis demonstrates how all three coupled aspects of multi-agent operations can be effectively designed and addressed within a task allocation and task execution framework. Through both simulation and on hardware, we showed that the presented framework is capable of realizing automated robotic convoying in unstructured environments. We further demonstrated the necessity of each component in addressing challenges faced by robotic systems, with each part of the thesis contributing to the introduction of a realizable framework for automated robotic convoying missions.

8.2 Limitations

During our study, we encountered several limitations in the practical and organizational aspects of our framework. These limitations often appeared either in modeling an element of the coupling addressed in the framework or as reflections on our realization of the robotic convoying system.

8.2.1 Limitations on Task Coordination

The framework presented in this thesis consists of two layers: a task allocation layer and a task execution layer. Principally, there exists an assumed separation between the two layers. As demonstrated in Part I, these two layers were motivated by the observation that the task allocation coupling and the task execution coupling did not need to be solved simultaneously. In particular, we leveraged the Convoy Coordinator's (Chapter 2) capability to act as an effective functional task abstraction for the robotic convoying problem. This abstraction then enabled us to allocate agents to convoys using the VRPMS-CC (Chapter 3). However, if forming this task abstraction independently of the task allocation problem proves challenging, a single-level framework may prove more effective in representing the task. Consider, for example, multi-agent pathfinding works. In combined task allocation and pathfinding ((MA-TC-PF) [35] and (Co-MAPF) [36]), the pathfinding component can be integral to determining feasible allocations. However, it is also clear that an allocation algorithm can only determine the optimal task allocations by considering the feasibility of the routes of individual agents, not just the coalition ("convoy") teams. Effectively, inter-agent coupling (e.g., path conflicts) may be a primary driver of the task coupling. If so, we cannot form the framework presented in this thesis. Such tight coupling between the task allocation and task execution layers requires the agents to address both types of coupling simultaneously. Currently, missions with such tight coupling lie outside the scope of the proposed framework. This set of extremely tightly-coupled missions, importantly, gives rise

to the idea that a balance may exist between the two types of discussed frameworks (i.e., the framework presented in this thesis and the aforementioned MAPF-style framework).

8.2.2 Limitations on Communications and Dynamism

To address challenges associated with the evolution of information during mission progression, we enforced a requirement that the agents must remain in communication with the base station (central optimizer) at all times. This requirement led to the creation of an automated communications network construction behavior that utilized the maximin communications spanning tree (Chapter 4) to deploy individual wireless repeaters. We then customized this behavior for robotic convoying applications in Chapter 5 by introducing a recovery mechanism in case an agent failed (lost communication) during system operation. Notably, the presented framework relies on the ability of agents to communicate information to one another. In the most basic sense, agents must possess the necessary capabilities to communicate information with one another; however, this limitation is much deeper than this superficial observation.

Foremost, in all testing conducted in Chapter 4 and Chapter 5, all agents were equipped with radio communication devices that enabled the transfer of information at a distance and gave a measure of relative radio signal strength. Outside of a potential heterogeneous communications capability across different agents in a multi-agent team —with the implication that some agents may not have the ability to transfer information by the designer's choice— the ability to communicate information at a stand-off distance can influence the ability for the convoy to maintain a coherent structure or lead to an unreasonable number of repeater placements. For example, should c_{thresh}, d_{LOS}, or d_{NLOS} be set too low, the ability to maintain communications will interfere with the required agents ability to execute the allocated tasks. At the current state of the work, communication constraints are present solely in the execution layer, limiting the system's response to allocation challenges. The ability to pose specific communications tasks would alleviate this limitation, but would require a balancing mechanism between the two layers of the framework.

Of additional note regarding this limitation is the impact of adversarial agents or environments on inter-agent communications. Unless designed robustly, communications systems are particularly susceptible to malicious information or actors. For example, while the presented framework demonstrated robustness concerning nodal failures (Chapter 4), an adversarial influence that:

1. incorrectly reports agent state information could indicate a perpetually "stuck" robotic convoy (Chapter 2),

2. provides inaccurate map information could consistently trigger a replan (Chapter 3),
3. inaccurately reports the measured signal strength could instigate an ineffective "peel-off" (Chapter 4), or
4. could create a "shell-game" of agent replacements due to a loss of communications (Chapter 5).

In effect, a consensus mechanism that can verify the veracity of data across different platforms is currently not present in the framework, but would be highly beneficial to improve its robustness.

8.2.3 Limitations on Environmental Interactability

Framework limitations regarding interactability largely stem from the task allocation component of the framework. It is clear from both our presented work and the literature on navigation among movable obstacles that reconfiguring the environment can alter the feasibility of tasks (consider simple children's "traffic jam" style puzzles). In this thesis, there is a strong notion of task independence regarding interactions. Namely, that interacting with the environment does not negatively change the feasibility of any other task in the environment. In particular, for problems with highly-coupled task allocation and task execution phases (e.g., MAPF-style works), this specific assumption likely does not hold.

Furthermore, from the works presented in this document, the framework reduces environmental interactions to a type of input data used in the task allocation layer. Specifically, the problem instance provides this information to the allocation layer in the form of the inter-location shortest path distances (i.e., replace an all-pairs obstacle-free shortest path algorithm with the procedure outlined in Chapter 7). Relegating the influence of interactability to merely a measure of task cost comes with certain limitations. In particular, we may wish to pose specific environmental interactions as individual tasks instead of as traversal costs (e.g., clearing an obstacle or removing an impediment). This representation of interaction tasks could be an interesting avenue to pursue, as we do not currently consider such tasks in our framework.

8.3 Looking Forward and Future Directions

The nature of scientific research is to evolve and progress over time. While much of this progression is organic, we would like to provide a few comments that we hope will guide future research.

8.3.1 On task allocation/task execution coupling

In this thesis, we considered a convoying problem that allows for a delineation between the task execution and task allocation layers. It is clear from certain applications (e.g., intra-logistical planning) that finding a suitable task abstraction may not be easy to identify, blurring the delineation between the layers. Exacerbating this problem, certain tasks exhibit high degrees of inter-agent coupling due to the required precision of execution needed in the task execution layer (e.g., strict formation-keeping). These problems are generally considered within the domain of multi-agent pathfinding (MAPF) [35, 45], as the path-planning component can drive system operations as much as the task allocation component. Understanding the stratification of problems on the spectrum between a hierarchical framework and the solvers presented in many MAPF works could provide helpful insight into the fundamental interplay between task allocation and task execution. A thorough investigation into the intersectionality between the two frameworks should reveal which framework naturally addresses which specific tasks. To align the framework presented in this work with problems similar to those considered in the MAPF domain, incorporating the convoy movement problem ([106]) could prove fruitful.

Furthermore, the convoying problem exhibited in this work demonstrated a low degree of dynamism [42], enabling on-the-fly replanning. It is clear from the complexity of the vehicle routing problem that our approach will struggle to scale to large, multi-agent systems with high degrees of dynamism. Future investigations should investigate the structure of these problems. While we demonstrated an exact methodology and a heuristic warm-start for solving the VRPMS-CC, adapting the solution methodology depending on mission constraints (e.g., planning time vs. needed accuracy) could be an interesting avenue of direction. Furthermore, decentralized auction-based methods ([37]) or learning-based methods ([39]) could provide an alternative to, or an augmentation of, the techniques presented in this framework. Finally, in Chapter 3, we only considered a single clustering rule (decomposition-based teaming). Investigating alternative clustering rules to model different mission profiles would extend our approach to a larger array of mission profiles.

8.3.2 On information coupling

While our earlier discussion on the degree of dynamism highlights the need for better information coupling, the design of the ad hoc network construction technique itself may also be of further interest. In particular, Tatum [55] first introduced a communications-aware map prediction algorithm for subterranean environments, which they utilized to optimize the placement of individual communications repeaters. Recent works, including

a maturation of prediction algorithms (see: Ho et al. [65]), have revitalized interest in concepts originally explored by [55]. By developing a network construction technique that is both predictive of the environment and reactive to the experienced signal strength, we anticipate that the agents will experience a reduction in deployed communication assets. Regarding the nature of the recovery behavior, it is clear that the current design generates a significant bulk movement of communication nodes to maintain the communication chain. Such bulk motions would be unsuitable for specific applications (e.g., defense), challenging the applicability of our recovery behavior in environments with adversarial agents.

8.3.3 On environmental coupling

Our last suggestion is to consider our interaction-aware interleaved search in the broader context of navigation among movable obstacles literature. In particular, we consider interactions that can be solely evaluated and conditioned on a single ego-agent's state and action. This simplifying assumption enables the state of the environment to remain untracked during system operation. Furthermore, by incorporating multi-agent interaction policies, we could further increase the reachable space of the multi-agent team. Both conventional and recent works [13, 101, 102, 107] demonstrate that the environment object configuration can also preclude successful object manipulations, especially for dexterous manipulators. Finally, while the interactions in this work are assumed to reflect a certain determinism, uncertainty in environmental sensing should play a larger role in evaluating the effectiveness of an agent in interacting with its environment. While parametric uncertainty is perhaps the most obvious demonstration of this aspect of the problem, an inherent risk is also associated with environmental interactions. We do not consider such uncertainties in this thesis, but their inclusion in interaction-aware motion planning and control algorithms is instrumental in developing mature interaction-aware robotic systems.

Appendix A

Overview of Robotic Agents

Figure A.1: The team of robotic agents that forms the basis of many of the hardware experiments demonstrated in this work. The quadruped agents are compatible with the presented framework, but are often not demonstrated in each section.

Citation:

> P. Sriganesh, J. Maier, A. Johnson, B. Shirose, R. Chandrasekar, **C. Noren**, J. Spisak, R. Darnley, B. Vundurthy, and M. Travers, "Modular, Resilient, and Scalable System Design Approaches - Lessons learned in the years after DARPA Subterranean Challenge," in *2024 IEEE International Conference on Robotics and Automation (ICRA) Workshop on Field Robotics*, Yokohama, Japan, May 2024

We demonstrate many of the technical contributions in this thesis on a novel robotic convoying system first developed at Carnegie Mellon University. We created this system based on the experiences of Team Explorer in the DARPA Subterranean Challenge [20]. We have documented many of the insights gained during our design of the system in our 2024 IEEE International Conference on Robotics and Automation (ICRA) Workshop paper [49]. We specifically designed the system to focus on enhancing inter-agent teaming through a robust and adaptable framework. The framework consists of extensible capabilities (e.g., the coordinated movement behavior in Chapter 2) that are realizable on the different agents of a heterogeneous agent team. Potential applications include mapping unknown, complex underground environments at high speed while maintaining accurate localizations and real-time, persistent communications.

A.1 System Operating Concept

In order to decrease operator workload and improve system redundancy, we designed the system to perform multi-agent platooning (convoying) in a linear formation (see Chapter 2). This platooning behavior requires multiple agents to form into a linear formation and travel together toward a common mission waypoint. The formation order can be determined autonomously or by a human operator. The agents can then autonomously or in a guided manner, where an operator only controls the lead robot, travel to the mission waypoint.

We embrace the concept of sliding-mode autonomy, enabling the operator to adjust the level of autonomy required for a task while maintaining high-level control. Our sliding-mode autonomy paradigm has different operational modes (similar to Team Explorer's implementation [108]) with increasing degrees of autonomy:

- **Full Manual Mode:** the operator commands directly control the robot
- **Smart Joystick Mode:** user-commanded direction is used in conjunction with a planner to avoid obstacles and navigate through narrow spaces
- **Waypoint Mode:** the robot navigates to a target location provided by the operator
- **Exploration Mode:** the robot autonomously explores an area and operates with the highest level of autonomy

A.2 Robotic Agents

The developed robotic system consists of a heterogeneous team of robotic platforms (agents) and an operator interface. The agent team (shown in Fig. A.1) may consist of legged

Property	Unit	Magnitude
L_T	[m]	0.77
L	[m]	0.48
L_f	[m]	0.23
L_{nose}	[m]	0.14
h_{bumper}	[m]	0.18
w	[m]	0.54
m	[kg]	14.7

Table A.1: Traxxas Vehicle Platform Parameters

or wheeled platforms. Legged platforms that are compatible with our framework include quadrupeds from Boston Dynamics. The wheeled platforms are sourced from Traxxas and can obtain a top speed of 10 [m/s] in complex environments. While the vehicle is capable of reaching speeds of 10 [m/s], the nominal operating speed is limited to only 3 [m/s] for safe operations. Relevant vehicle parameters include the platform's length (L_T), the wheelbase length (L), the front axle to center of mass distance (L_f), the front axle to nose distance (L_{nose}), the height of the middle of the attached bumper (h_{bumper}), the platform width (w), and the platform mass (m). These parameters are given in Table A.1.

A.3 Agent Payload

Each agent is equipped with a custom-built payload that contains onboard compute. In Chapter 2, the onboard compute is a Jetson AGX Xavier. The onboard computer was later upgraded to Jetson AGX Orin 64GB developer version in Chapter 3, Chapter 4, Chapter 5, and Chapter 7. Furthermore, each platform is also equipped with a set of exteroceptive sensors (both a Light Detection and Ranging (Velodyne 16 Lite LiDAR) sensor and cameras (multiple 195-degree fisheye cameras from Leopard Imaging Inc.)), and an inertial measurement unit (EPSON IMU line; possible options are: G365, G366, G370). Additional payload camera options include the Zed series cameras (ZEDX, ZEDX-mini, ZED-one, ZED2, etc.) from Stereolabs, as well as all Intel RealSense series cameras (D405i, D435i, D455i). This payload enables the agents to simultaneously map their environment and compute their odometry relative to their starting location (e.g., the position of the operator interface) using a feature-based Simultaneous Localization and Mapping (SLAM) methodology known as "Super Odometry" [99].

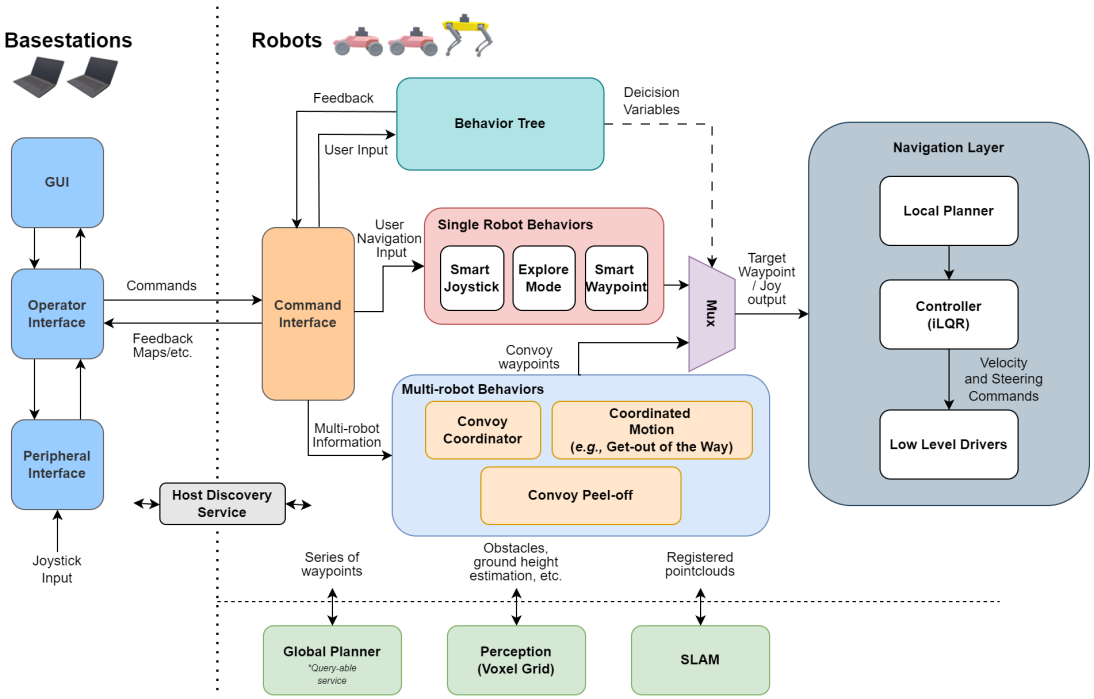


Figure A.2: An overview of the system architecture

A.4 Agent Architecture

Each agent is also equipped with an onboard radio that allows it to communicate information to the other agents and the operator interface. The system uses Robot Operating System (ROS) and Data Distribution Service (DDS) middleware for communication between the base station and robotic agents. Each agent maintains its own local motion-primitive-based obstacle avoidance planner and its own local receding-horizon tracking controller. To operate in environments without a supporting communication infrastructure, each agent is equipped with a communication radio that enables it to contribute to the communication network construction techniques described in Chapter 4 and Chapter 5. The construction behavior ensures all agents are connected to each other and to the operator interface. The operator interface is also equipped with a communications radio.

Behavior trees are central to our architecture design, managing complex system-mode transitions, enhancing modularity, and ensuring a clear structure [109]. Within our proposed architecture, the behavior tree serves as the central authority, determining the robot's active mode. The behavior tree receives requests from the command interface, assesses their viability, switches the robot to the requested mode, and enables the corresponding

channel on the mux.

Each robot runs its own behavior tree, validating requests against predefined conditions. These conditions are based on hardware availability, environmental constraints, subsystem health, or the agent's status. For instance, if the robot is not receiving joystick commands or if SLAM is not initialized, a "smart joystick mode" request will be discarded, as the "smart joystick mode" requires SLAM to function correctly. This structure of the behavior tree makes it easier and faster to define and implement new modes, along with their activation conditions.

Once the behavior tree determines the operational mode, it communicates this decision to the mux. The mux filters the outputs from different subsystems, given the current robot mode specified by the behavior tree. This ensures that the navigation layer does not receive conflicting targets, promoting a centralized control flow throughout the system. It also facilitates seamless integration of different subsystems.

The current behavior tree state (operational mode, authorized basestation, convoy order, etc.) is sent back to the operator interface as feedback. This helps create an adaptive interface that presents valid options to the user based on the operator's selection and input from robot behavior trees.

Bibliography

[1] S. Y. Kim, "Ksc battalion conducts convoy movement training [image 10 of 19]," 2025, [Online; accessed June 27, 2025 from Defense Visual Information Distribution Service]. [Online]. Available: https://www.dvidshub.net/image/8923542/ksc-battalion-conducts-convoy-movement-training (document), 1.2

[2] M. Abban, "Uscgc bear (wmec 901) participates in operation nanook [image 5 of 14]," 2022, [Online; accessed June 27, 2025 from Defense Visual Information Distribution Service]. [Online]. Available: https://www.dvidshub.net/image/7369331/uscgc-bear-wmec-901-participates-operation-nanook (document), 1.2

[3] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154. (document), 2.5, 2.5

[4] C. Noren, B. Vundurthy, S. Scherer, and M. Travers, "Trajectory optimization for vegetation override," in *Proceedings of the 16th European-African Regional Conference of the International Society of Terrain-Vehicle Systems (ISTVS)*, 2024. (document), 7.1, 7.2.1, 7.2, 7.2.3, 7.2.4, 7.3, 7.4, 7.4.3, 7.6

[5] D. A. Green, "The future of autonomous ground logistics: Convoys in the department of defense," School of Advanced Military Studies (SAMS), School of Advanced Military Studies (SAMS), 250 Gibbon Ave. Fort Leavenworth, KS 66027-2134, Electronic, February 2011, [Online, accessed 15-September-2023]. 1.1, 3.2

[6] D. Starry, *Mounted Combat in Vietnam*. United States Government Printing Office, 1978. 1.1

[7] D. Dominique, *The Original Tactical Convoy Handbook*. Wounded Warrior Publications, 2015. 1.1, 2.3, 3.4

[8] S. Nahavandi *et al.*, "Autonomous convoying: A survey on current research and development," *IEEE Access*, vol. 10, pp. 13 663–13 683, 2022. 1.1, 1.4, 2.1, 3.2

[9] H. Psaraftis, "Dynamic vehicle routing problems," *Vehicle Routing: Methods and Studies*, pp. 223–248, 1988. 1.1

[10] C. Urmson, "Navigation regimes for off-road autonomy," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, May 2005. 1.1, 7.1

[11] G. Nestlinger, J. Rumetshofer, and S. Solmaz, "Leader-based trajectory following in unstructured environments; from concept to real-world implementation," *Electronics*, vol. 11, no. 12, 2022. 1.4, 2.2, 2.4.1

[12] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, 1991. 1.4, 7.1

[13] M. Stilman and J. Kuffner, "Navigation among movable obstacles: real-time reasoning in complex environments," in *Proceedings of the 2004 IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, vol. 1, 2004, pp. 322–341. 1.4, 7.1, 8.3.3

[14] M. Rybansky, "Trafficability analysis through vegetation," in *Proceedings of the International Conference on Military Technologies (ICMT)*, Brno, Czech Republic, 2017, pp. 207–210. 1.4, 6.1, 6.2

[15] ——, "Determination the ability of military vehicles to override vegetation," *Journal of Terramechanics*, vol. 91, 2020. 1.4, 6.2, 7.1

[16] C. A. Blackmon and D. D. Randolph, "An analytical model for predicting cross-country vehicle performance, vol. ii: Longitudinal obstacles. appendix b: Vehicle performance in lateral and longitudinal obstacles, vegetation," U.S. Army Engineering Waterways Experiment Station, Tech. Rep., 1968. 1.4, 6.2, 6.3.2, 6.3.2, 6.5, 6.5.3, 7.1, 7.2.3, 7.2.3, 7.4.2, 7.5, 8.1

[17] T. T. Vong, G. A. Hass, and C. L. Henry, "Nato reference mobility model (nrmm) modeling of the demo iii experimental unmanned ground vehicle (xuv)," Army Research Laboratory, Aberdeen Proving Ground, Tech. Rep., 1999. 1.4, 6.2, 7.1

[18] M. Bradbury *et al.*, "Next generation nato reference mobility model (nrmm) development," North Atlantic Treaty Organization Science and Technology Organization, Tech. Rep., 2018. 1.4, 6.2, 7.1

[19] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 12–28, 2017. 2.1, 2.2

[20] S. Scherer *et al.*, "Resilient and modular subterranean exploration with a team of roving and flying robots," *Field Robotics*, vol. 2, pp. 678–734, 2022. 2.1, 4.3, 4.4, 4.5.2, 4.6.2, 4.7, 5.2, A

[21] J. Yazbeck, A. Scheuer, and F. Charpillet, "Decentralized near-to-near approach for vehicle platooning based on memorization and heuristic search," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 631–638. 2.2, 2.4.1

[22] A. Albrecht, N. F. Heide, C. Frese, and A. Zube, "Generic convoying functionality for autonomous vehicles in unstructured outdoor environments," in *Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1949–1955. 2.2, 2.5

[23] J. Shin, D. Kwak, and J. Kim, "Autonomous platooning of multiple ground vehicles in rough terrain," *Journal of Field Robotics*, vol. 38, 09 2020. 2.2

[24] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological systems," in *Proceedings of the 2004 International Conference on Informatics in Control, Automation and Robotics*, 2004, pp. 222–229. 2.4.1

[25] J. Zhang, C. Hu, R. Chadha, and S. Singh, "Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation," *Journal of Field Robotics*, vol. 37, 04 2020. 8, 2.4.3

[26] X. Zhao, W. Yao, N. Li, and Y. Wang, "Design of leader's path following system for multi-vehicle autonomous convoy," in *Proceedings of the 2017 IEEE International Conference on Unmanned Systems (ICUS)*, 2017, pp. 132–138. 2.5

[27] A. Bayuwindra, J. Ploeg, E. Lefeber, and H. Nijmeijer, "Combined longitudinal and lateral control of car-like vehicle platooning with extended look-ahead," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 790–803, 2020. 2.5

[28] M.-M. Mohamed-Ahmed, A. Naamane, and N. K. M'Sirdi, "Path tracking for the convoy of autonomous vehicles based on a non-linear predictive control," in *Proceedings of the 12th International Conference on Integrated Modeling and Analysis in Applied Control and Automation*, Sep. 2019. 2.5

[29] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple travelling salesman problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 100369, no. 40, 2021. 3.2

[30] M. Drexl, "Synchronization in vehicle routing - a survey of vrps with multiple synchornization constraints," *Transportation Science*, vol. 46, no. 3, pp. 297–316, 2012. 3.3

[31] I.-M. Chao, "A tabu search method for the truck and trailer routing problem," *Computers and Operations Research*, 2002. 3.3.1

[32] R. Soares, A. Marques, P. Amorim, and S. N. Parragh, "Synchronisation in vehicle routing: Classification schema, modelling framework and literature review," *European Journal of Operational Research*, 2023. 3.3.1

[33] S. Ropke and J.-F. Cordeau, "Branch and cut and price for the pickup and delivery problem wiht time windows," *Transportation Science*, vol. 43, 2009. 3.3.1

[34] H. H. Doulabi, G. Pesant, and L.-M. Rousseau, "Vehicle routing problems with synchronized visits and stochastic travel and service times: Applications in healthcare," *Transportation Science*, vol. 54, pp. 1053–1072, 2020. 3.3.1

[35] Z. Ren, C. Zhang, S. Rathinam, and H. Choset, "Search algorithms for multi-agent teamwise cooperative path finding," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 3.3.2, 8.2.1, 8.3.1

[36] N. Greshler, O. Gordon, O. Salzman, and N. Shimkin, "Cooperative multi-agent path finding: Beyond path planning and collision avoidance," in *Proceedings of the 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 20–28. 3.3.2, 8.2.1

[37] H.-L. Choi, A. K. Whitten, and J. P. How, "Decentralized task allocation for hetero-

geneous teams with cooperation constraints," in *Proceedings of the 2010 American Control Conference (ACC)*, 2010. 3.3.3, 8.3.1

[38] R. Patel *et al.*, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 3.3.3

[39] H. Fu, M. You, H. Zhou, and B. He, "Closely cooperative multi-agent reinforcement learning based on intention sharing and credit assignment," *IEEE Robotics and Automation Letters*, 2024. 3.3.4, 8.3.1

[40] A. Bogyrbayeva *et al.*, "A deep reinforcement learning approach for solving the traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, 2023. 3.3.4, 3.8

[41] L. He, J. Pan, S. Narang, W. Wang, and D. Manocha, "Dynamic group behaviors for interactive crowd simulation," 2016. 3.3.4

[42] P. Toth and D. Vigo, *Vehicle Routing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014. 3.4, 3.5, 3.7.2, 8.3.1

[43] W. Winston, *Operations Research: Applications and Algorithms*. Belmont, CA: Thomson Learning, 2004. 3.5

[44] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, pp. 254–265, 1987. 3.6, 3.6.1

[45] R. Stern *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the 2019 Symposium on Combinatorial Search (SoCS)*, 2019. 3.6, 3.6.1, 4.2, 4.6, 5.4, 5.4.1, 8.3.1

[46] DARPA, "Defense advanced research project agency, subterranean (subt) challenge (archived): Subterranean challenge final event," 2019. [Online]. Available: https://www.darpa.mil/research/challenges/subterranean 3.6.2, 4.2, 5.1, 5.2

[47] G. Yang, "Theory and methodology: Transformation of multidepot multisalesmen problem to the standard travelling salesman problem," *European Journal of Operational Research*, 1995. 3.6.3

[48] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society of Applied Mathematics*, vol. 10, no. 1, March 1962. 3.6.3

[49] P. Sriganesh *et al.*, "Modular, resilient, and scalable system design approaches - lessons learned in the years after DARPA subterranean challenge," in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA) Workshop on Field Robotics*, 2024. 3.7.1, 5.5.1, A

[50] A. R. da Silva, L. Chaimowicz, T. C. Silva, and M. A. Hsieh, "Communication-constrained multi-robot exploration with intermittent rendezvous," in *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. 3.8, 5.2, 5.6

[51] J. Gielis, A. Shankar, and A. Prorok, "A critical review of communications in multi-

robot systems," *Current Robotic Reports*, vol. 3, no. 4, pp. 213–225, 2022. 4.3, 5.2, 5.3.2

[52] M. Tranzatto *et al.*, "CERBERUS: autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge," *CoRR, arXiv*, 2022. 4.3, 5.2

[53] M. Zoula and J. Faigl, "Wireless communication infrastructure building for mobile robot search and inspection missions," in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 5970–5976. 4.3, 4.4, 4.5

[54] C.-L. Lu *et al.*, "A heterogeneous unmanned ground vehicle and blimp robot team for search and rescue using data-driven autonomy and communication-aware navigation," *Field Robotics*, vol. 2, pp. 557–594, 2022. 4.3, 5.2

[55] M. Tatum, "Cmu-ri-tr-20-19: Communications coverage in unknown underground environments," Master's thesis, Carnegie Mellon University, 2020. 4.3, 4.8, 5.2, 8.3.2

[56] T. S. Vaquero, M. S. da Silva, K. Otsu, M. Kaufman, J. A. Edlund, and A.-a. Aghamohammadi, "Traversability-aware signal coverage planning for communication node deployment in planetary cave exploration," in *Proceedings of the 2020 International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2020. 4.3, 5.2

[57] G. Kazazakis and A. Argyros, "Fast positioning of limited-visibility guards for the inspection of 2d workspaces," in *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2002, pp. 2843–2848. 4.3, 5.2

[58] M. Ernestus *et al.*, "Algorithms for art gallery illumination," *Journal of Global Optimization*, vol. 68, 05 2017. 4.3, 5.2

[59] B. Ballinger *et al.*, "Coverage with k-transmitters in the presence of obstacles," *Journal of Combinatorial Optimization*, vol. 25, 01 2013. 4.3, 5.2

[60] G. Antonelli, F. Arrichiello, S. Chiaverini, and R. Setola, "A self-configuring manet for coverage area adaptation through kinematic control of a platoon of mobile robots," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 1332–1337. 4.3

[61] R. Pandey, A. K. Singh, and K. M. Krishna, "Multi-robot exploration with communication requirement to a moving base station," in *Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering (CASE)*, 2012, pp. 823–828. 4.3

[62] M. Wzorek, C. Berger, and P. Doherty, "Router and gateway node placement in wireless mesh networks for emergency rescue scenarios," *Autonomous Intelligent Systems*, vol. 1, 12 2021. 4.3

[63] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 6th ed. Springer, 2018. 4.4

[64] B. Yamaguchi, "A frontier-based approach for autonomous exploration," in *Proceedings*

1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), 1997, pp. 146–151. 4.7

[65] C. Ho *et al.*, "Mapex: Indoor structure exploration with probabilistic information gain from global map predictions," 2024. 4.8, 8.3.2

[66] B. Morrell *et al.*, "An addendum to nebula: Toward extending team costar's solution to larger scale environments," *IEEE Transactions on Field Robotics*, vol. 1, pp. 476–526, 2024. 5.2

[67] H. M. Ammari, "A computational geometry-based approach for planar k-coverage in wireless sensor networks," *Association for Computing Machinery (ACM) Transactions on Sensor Networks*, vol. 19, no. 2, 2023. 5.2

[68] C. Rizzo, D. Tardioli, D. Sicignano, L. Riazuelo, J. L. Villarroel, and L. Montano, "Signal-based deployment planning for robot teams in tunnel-like fading environments," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1381–1397, 2013. 5.2

[69] D. Tardioli *et al.*, "Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments," *Journal of Field Robotics*, vol. 36, no. 6, pp. 1074–1101, 2019. 5.2

[70] M. A. Islam, M. Maiti, Q. M. Alfred, P. K. Ghosh, and J. Sanyal, "Attenuation modelling and machine learning based snr estimation for 5g indoor link," in *Proceedings of the 2020 IEEE VLSI Device, Circuit and System Conference (VLSI-DCS)*, 2020. 5.2

[71] X. Wang, M. C. Gursoy, T. Erpek, and Y. E. Sagduyu, "Jamming-resilient path planning for multiple uavs via deep reinforcement learning," in *Proceedings of the 2021 IEEE International Conference on Communications (ICC) Workshop*, 2021. 5.2

[72] B. Woosley, P. Dasgupta, J. G. R. III, and J. Twigg, "Multi-robot information driven path planning under communication constraints," *Autonomous Robots*, vol. 44, pp. 721–737, 2020. 5.2

[73] B. Paden, M. Cap, S. Z. Young, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, 2016. 6.2

[74] S. Thurn *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, 2007. 6.2

[75] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Springer Tracts in Advanced Robotics*, vol. 56, 2009. 6.2

[76] S. M. Lavalle, *Planning Algorithms*. University of Illinois, Cambridge University Press, 2006, (Online, accessed 05/15/2023). 6.2, 7.1, 7.2.4

[77] C. Jianyu, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, 2017. 6.2, 6.5

[78] T. Howell, B. Jackson, and Z. Mancheseter, "Altro: A fast solver for constrained trajectory optimization," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 7674–7679. 6.2

[79] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 6, no. 34, pp. 1603–1622, 2018. 6.2

[80] C. Liu and M. Tomizuka, "Control in a safe set: Addressing safety in human-robot interactions," *Proceedings of the ASME 2014 Dynamic Systems and Control Conference*, vol. 3, pp. 22–24, 2014. 6.2

[81] C. Noren, W. Zhao, and C. Liu, "Safe adaptation with multiplicative uncertainties using robust safe set algorithm," *Proceedings of the 2021 Modeling Estimation and Control Conference (MECC)*, vol. 54, pp. 360–365, 2021. 6.2

[82] A. Kelly *et al.*, "Towards reliable off road autonomous vehicles operating in challenging environments," *The International Journal of Robotics Research*, vol. 25, pp. 449–483, 2006. 6.2

[83] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using sumt," in *9th IEEE International Workshop on Advanced Motion Control*, 2006, pp. 665–671. 6.2, 7.1

[84] ——, "A combined continuation and penalty method for the determination of optimal hybrid mechanical trajectories," in *IUTAM Symposium on Dynamics and Control of Nonlinear Systems with Uncertainty*. Springer Netherlands, 2007, pp. 187–196. 6.2, 7.1

[85] K. Yunt, "An augmented lagrangian-based shooting method for the optimal trajectory generation of switching lagrangian systems," *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, vol. 18, pp. 615–645, 2011. 6.2, 7.1

[86] T. Howell, K. Tracy, S. Le Cleach, and Z. Manchester, "Calipso: A differentiable solver for trajectory optimization with conic and complementarity constraints," *Robotics Research ISRR 2022. Springer Proceedings in Advanced Robotics*, vol. 27, pp. 1–25, 2023. 6.2

[87] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation." *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987. 6.2, 6.3.4, 7.1

[88] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, pp. 849–904, 2017. 6.2, 6.3.1, 6.3.4, 7.1, 7.2.1

[89] G. L. Mason, B. Q. Gates, and V. D. Moore, "Determining forces required to override obstacles for ground vehicles," *Journal of Terramechanics*, vol. 49, no. 3-4, pp. 191–196, 2012. 6.2, 6.3.2, 6.3.2, 6.3.2, 6.3.2, 6.3.4, 6.3.5, 6.4, 6.5, 6.5.1, 6.5.2, 6.5.3, 7.1, 7.2.3, 8.1

[90] M. N. Moore *et al.*, "Override forces through clumps of small vegetation," *Journal of*

Terramechanics, vol. 116, p. 100988, 2024. 6.2

[91] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," 2020. 6.2

[92] M. Sivaprakasam *et al.*, "Tartandrive 2.0: More modalities and better infrastructure to further self-supervised learning research in off-road driving tasks," 2024. 6.2

[93] S. Sharma *et al.*, "Cat: Cavs traversability dataset for off-road autonomous driving," *IEEE Access*, vol. 10, pp. 24 759–24 768, 2022. 6.2

[94] M. G. Castro *et al.*, "How does it feel? self-supervised costmap learning for off-road vehicle traversability," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 931–938. 6.2

[95] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast Traversability Estimation for Wild Visual Navigation," in *Proceedings of 2023 Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023. 6.2

[96] E. Chen, C. Ho, M. Maulimov, C. Wang, and S. Scherer, "Learning-on-the-drive: Self-supervised adaptation of visual offroad traversability models," 2024. 6.2

[97] J. Frey, M. Patel, D. Atha, J. Nubert, D. Fan, A. Agha, C. Padgett, P. Spieler, M. Hutter, and S. Khattak, "Roadrunner – learning traversability estimation for autonomous off-road driving," 2024. 6.2

[98] M. Prágr, J. Bayer, and J. Faigl, "Autonomous exploration with online learning of traversable yet visually rigid obstacles," *Autonomous Robots*, 2023. 6.2

[99] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments," in *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8729–8736. 6.5, A.3

[100] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. 7.1, 7.2.1, 7.2.2

[101] D. Saxena, M. S. Saleem, and M. Likhachev, "Manipulation planning among movable obstacles using physics-based adaptive motion primitives," in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 6570 – 6576. 7.1, 8.3.3

[102] D. Saxena and M. Likhachev, "Planning for manipulation among movable objects: Deciding which objects go where, in what order, and how," in *Proceedings of the 2023 International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 33, no. 1, 2023, pp. 668–676. 7.1, 8.3.3

[103] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," *Proceedings to the 3rd International Conference on Automation, Control, and Robots (ICACR) 2019*, pp. 22–29, 2019. 7.1

[104] G. Yang, M. Cai, A. Ahmad, A. Prorok, R. Tron, and C. Belta, "Lqr-cbf-rrt*: Safe

and optimal motion planning," *arxiv*, 2023. 7.1

[105] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4187–4194. 7.2.4, 7.6

[106] H. Mokhtar, M. Krishnamoorthy, N. R. Dayama, and P. R. Kumar, "New approaches for solving the convoy movement problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 133, p. 101802, 2020. 8.3.1

[107] D. M. Saxena and M. Likhachev, "Planning for complex non-prehensile manipulation among movable objects by interleaving multi-agent pathfinding and physics-based simulation," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8141–8147. 8.3.3

[108] S. Scherer *et al.*, "Resilient and modular subterranean exploration with a team of roving and flying robots," *Field Robotics Journal*, pp. 678–734, May 2022. A.1

[109] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018. A.4