

Data-driven multi-robot planning for online user-directed coordination

Ellen A. Cappel

CMU-RI-TR-20-48

September 2020

Robotics Institute, School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Nathan Michael, CMU, Chair

Howie Choset, CMU

Maxim Likhachev, CMU

Mac Schwager, Stanford University

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

For Mom, Dad, Emily, and Mark

Abstract

Multi-robot applications frequently seek to employ human operators to direct robot actions online because fully automated planners struggle to encode human expertise or handle the extenuating circumstances that occur during real world operations. However, it is extremely challenging for a human to direct multi-robot teams, especially online, i.e., in real-time. From entertainment to defense, applications can require detailed inter-robot coordination direction which can be difficult for an operator to specify. Finding motion plans that then meet user intent is complicated by the need to operate in cluttered environments while absolving the human operator from having to consider physical constraints of the system such as dynamic feasibility or safety.

Existing multi-robot policy specification and planning methods have difficulty providing all the required capabilities to enable high-speed, online, human-directed multi-agent motion generation in cluttered environments. While reaction based or local control methods have been widely used for multi-robot applications due to their fast computation times, these methods can fail to produce coordinated responses or safety guarantees. In contrast, optimization methods and search approaches are frequently used to coordinate actions across robots and generate high quality motion plans. Unfortunately, the large number of inherent decision variables, especially as frequently incurred when considering numerous obstacles, often means these methods require computation times in excess of online operation limits.

In this thesis, we enable a search based methodology to find safe and feasible multi-robot trajectories at fast time scales even in highly cluttered environments by leveraging a group-based representation of multi-robot actions and offline acquired data. Our approach generalizes across environments and enables an operator to coordinate multi-robot motions online for applications requiring high-frequency plan generation such as required by joystick control. We show results for two human-commanded multi-robot applications: theatrical performance and urban reconnaissance. Both applications require a multi-robot planner to resolve differences between operator input and feasibility requirements given the current state of the system, and generate dynamically feasible and safe trajectories for all agents within time and computation constraints.

This thesis establishes an online multi-robot planning and control system for the specification of multi-robot behaviors, and we demonstrate the feasibility of the approach using a multi-quadrotor system, controlling up to 30 physical robots online in the context of the theatric application. We then address assumptions made under the theatric context to meet the high-frequency planning and cluttered environment context of the urban reconnaissance application, and present a generalized formulation of multi-robot behaviors and leverage examples of multi-agent actions from real world data sets to inform an online search policy. We demonstrate that the generalized representation coupled with the proposed data-based search heuristic enables high-speed multi-robot coordination in cluttered environments, providing needed capabilities to enable multi-robot solutions for complex real world applications.

Acknowledgments

This work would not have been possible without the guidance of many people in and outside of CMU and I am sincerely grateful for their assistance and encouragement.

Foremost, I would like to thank my advisor Nathan Michael. His patient mentoring has been an invaluable privilege, and always appreciated. Throughout my PhD Nate not only helped with tasks from testing motor controllers to writing firmware, but taught me principles of research and work habits that will continue to shape my career path. I also give sincere thanks to my PhD advisory committee members Mac Schwager and Maxim Likhachev for their valuable perspectives and advice, and to Howie Choset, who gave me my start in robotics research in the masters program and continued his support as a member of my committee.

I have been privileged to work with many gifted collaborators, especially Ali Momeni for opportunities to work with the Beckonbot project and the performance collaboration with YouTube and AURORA. I also thank the many student, faculty, and industry collaborators on these projects for involving me in these uniquely creative challenges. These experiences helped shape my thesis research.

I owe a debt to all my colleagues in the Resilient Intelligent Systems Lab at CMU for their invaluable collaborations, peer review, and discussion. Arjav Desai was hugely helpful in contributing to our multi-robot behavior research; his assistance with designing and running experiments, writing papers, and providing critique and feedback was invaluable. My thanks to Matt Collins for making our hardware experiments with the Crazyflie systems possible by developing the charging boards and early controllers, and to my coauthor Derek Mitchell for his help with coding, demos, and theory review. I offer my heartfelt appreciation to Vishnu Desaraju, who first taught me nonlinear controls ten years ago and continued to teach me all the way through my time at CMU. I'd like to thank Micah Corah, Kumar Shaurya Shankar, John Yao, Wennie Tabib, and the many other members of RISLab past and present for their insightful discussions on Monte-Carlo search, affine transforms, state estimation, and the future of robotics. To my RI friends Victor Hwang, Liz Cha, Humphrey Hu, Karthik Lakshmanan, Arun Venkatraman, Nitish Thatte, Tim Lee, Allie Del Giorno, and Ada Taylor, who critiqued practice talks, reviewed essays, debugged code, and helped build robot golf courses—you made life as a PhD student not only collaborative, but illuminating and fun.

Finally, my deepest gratitude to my friends outside CMU for their love and support. To the family who joined us during my graduate degree, John, Katharine, and Danny, but most especially to Mom, Dad, Emily, and Mark: your years of unwavering support made this accomplishment possible. Thank you.

Contents

1	Introduction	1
2	Background and Related Work	7
2.1	Interpreting human intent for multi-robot systems	8
2.2	Multi-robot planning and trajectory generation	9
2.2.1	Reactive methods	9
2.2.2	Optimization methods	10
2.2.3	Search methods	11
3	A system for online behavior specification and execution	17
3.1	System Design	19
3.2	Input Parameterization	21
3.3	Multi-robot Behaviors and Trajectory Planning	25
3.3.1	Verification and Mitigation	27
3.3.2	Behavior transitions	28
3.3.3	Offline verification	32
3.4	Evaluation and Results	34
3.4.1	Evaluation: Robustness and coverage	34
3.4.2	Evaluation: Hardware Experiments	35
4	Formation planning with experience	39
4.1	Introduction	40
4.2	Formation Planning, an Example	42
4.3	Proposed Methodology	46
4.3.1	Robots, Formations, and Behaviors	46
4.3.2	Environment and Collision Checking	47
4.3.3	Definition of Shape Transforms	48
4.3.4	Cost Function	50
4.3.5	Optimization Problem	51
4.3.6	Search graph formulation	52
4.3.7	MCTS formulation	52
4.4	Experiments and Analysis	54
4.4.1	Method performance characteristics	57
4.4.2	Transfer across environments	57
4.5	Discussion and Conclusion	58
5	Multi-robot planning via action search	61
5.1	Approach Overview	62

5.2	Notation and problem formulation	65
5.2.1	Robot, formation, and obstacle notation	66
5.2.2	MDP over states and actions	67
5.3	Demonstration data as $\{\mathbf{s}, \mathbf{a}\}$ tuples	68
5.3.1	Position and translation components of states and actions	69
5.3.2	Rotation component, Ω_t , of action \mathbf{a}_t	69
5.3.3	Discussion of R_0 and \mathbf{S}_0	70
5.3.4	Rotation and shape components of state	71
5.3.5	Shape transform A_t	72
5.4	Online action search	73
5.4.1	MDP as tree search	73
5.4.2	Selection policy	74
5.5	Evaluation Metrics	75
5.6	Evaluation	78
5.7	Concluding remarks	81
6	Data-driven search for joystick-directed reconnaissance	83
6.1	Introductory discussion	84
6.2	Methodology Attributes	90
6.2.1	Method Completeness and Complexity	92
6.2.2	Optimality	93
6.2.3	Solution Safety	95
6.3	Evaluation	97
6.3.1	Data attributes	97
6.3.2	Solution attributes	99
6.3.3	Experimental setup	100
6.3.4	Validity of biased action selection	101
6.3.5	Generalizing with action resolution	104
6.3.6	Scaling with group size	108
6.3.7	Generalizing with environment complexity	109
6.3.8	Method comparison	113
6.4	Concluding Remarks	118
7	Conclusion	121
7.1	Contribution Summary	122
7.2	Opportunities for Future Work	123

List of Figures

1.0.1 Multi-robot examples in the entertainment and defense fields	2
1.0.2 Two user-directed, multi-robot system pipelines	5
3.1.1 System overview	19
3.2.1 Representative behavior descriptors	21
3.2.2 Behavior composition, validation, verification, and refinement	26
3.3.1 The behavior transition process	29
3.3.2 Coverage over representative behavior descriptor combinations	32
3.3.3 Interface examples	33
3.4.1 Dynamic feasibility, inter-robot clearance distance, and timing properties	36
3.4.2 Hardware results	36
3.4.3 Flight images	37
3.4.4 Theatric performance photos	38
4.1.1 Motivating example.	43
4.2.1 Example costs.	44
4.3.1 Cost calculation.	50
4.4.1 Representative trial.	55
4.4.2 Training curves.	56
4.5.1 Environment transfer.	58
5.1.1 Complex environment coordination example	63
5.1.2 Approach overview diagram	65
5.6.1 Coordinated action examples during search	79
5.6.2 Environments of varying complexity used in evaluation	81
5.6.3 Dynamic feasibility trajectory characteristics	81
6.2.1 Toy search example illustrating completeness	94
6.3.1 Dataset attributes	98
6.3.2 Validity of biased action selection	103
6.3.3 Action resolution and search set size	106
6.3.4 Time scaling with collision checking method and number of robots	109
6.3.5 Randomly generated environment examples and related parameters and features	110
6.3.6 Search performance over environment variations	111
6.3.7 Comparison of approaches, 2.5D scenario	114
6.3.8 Comparison of approaches, 3D scenario	116
6.3.9 Comparison of trajectories, 3D scenario	117

List of Tables

5.6.1 Solution metrics with environment variation	80
6.3.1 Metrics quantifying discrete search and interpolated trajectories	100
6.3.2 Validity of biased action selection	102
6.3.3 Comparison methods over 2.5D example scenario	114
6.3.4 Comparison methods over 3D example scenario	116

Chapter 1

Introduction

Planning motions online for multi-robot teams in response to human operator input is a challenging but sought after capability. From transporting objects [5] to performing escort missions [7] and sensor coverage [21, 25], numerous diverse applications seek to employ multi-agent teams. Compelling examples include applications as disparate as entertainment and defense, where companies and academic groups across nations have invested significant resources into developing methods to perform multi-robot coordination. In entertainment, hundreds to thousands of quadrotors were employed in coordinated light shows and viewed by millions of people for high impact events including the 2017 Super Bowl in Houston, TX USA [40], the 2017 New Year's celebration in Guangzhou, China [30], and the 2018 Winter Olympics in PyeongChang, Korea [39]. In the field of defense, large scale international competitions are held to foster development of multi-agent urban reconnaissance and search and rescue capabilities. These competitions include the 2010 MAGIC (Multi Autonomous Ground Robotic International Challenge) event, sponsored by United States and Australian government agencies [8, 38] and offering 1.6 million dollars in collective



(a) Quadrotor light show at the PyeongChang 2018 Winter Olympics [39]. Photo credit: Intel Corporation.



(b) The team of ground robots used for reconnaissance in the MAGIC 2010 event by the University of Michigan competition winners [65]. Photo credit: Australian Government Defense Science and Technology Organisation [8].



(c) Visual user interface from [33] corresponding to a multi-robot simulation environment used in the 2016 competition of RoboCup Rescue Simulation League. Image is taken from the 2016 champion team paper [33].

Figure 1.0.1: Multi-robot examples in entertainment (Fig. 1.0.1a), reconnaissance (Fig. 1.0.1b), and search and rescue (Fig. 1.0.1c).

prize money, and the long running RoboCup Rescue, held annually since 2000 with multi-robot simulation competitions added in 2006 [42, 68, 74]. However, the challenges in encoding human expert knowledge into an autonomous artificial intelligence and the complications of handling unexpected operating circumstances in real world environments leads many applications to leverage human operator input to direct team actions. In entertainment, quadrotor theatrical performances have sought to use human performers to direct aerial choreography [15, 54]. In the aforementioned examples of the RoboCup Rescue and MAGIC 2010 competitions, methodologies focus on incorporating human operators for disaster relief [13] and reconnaissance [65], respectively.

While human input can enable operation in these difficult scenarios, operator directed, multi-robot motion planning still faces a number of significant challenges. With respect to enabling online human direction, an operator’s intent for a group of agents must be translated to individual agent actions while minimizing cognitive load on the user, even when detailed coordination descriptions or a frequent rate of interaction is required. Similarly, an operator should not be required

to consider vehicle dynamic limitations or potential collisions. Operator intent for the group must be mapped to dynamically feasible and collision free plans for every member of the robot team, and the difficulty of motion planning is increased by the need to coordinate motions between group members. The complexity of path planning and likelihood of collisions is increased in highly cluttered environments, where modeling free space can be nontrivial. Operator commands are issued while a system is operating, requiring high speed re-planning with respect to the robots' current states. And while we view the operator as expert, humans may still specify erroneous input, giving instructions that prove to be infeasible. The online nature of this application places further restrictions on the timing and computational limits of any potential methodology.

The current state of art for multi-robot motion planning is varied and well developed, but lacking with respect to online coordinated policy specification and rapid planning methods for teams in highly cluttered environments. Across human–multi-agent interfacing approaches, methods trade off instruction complexity with input frequency. Applications that require detailed coordination between agents face the challenge of specifying all the required degrees of freedom without a correspondingly high cognitive burden, to allow a human to easily command the system online. When high interaction rates are required, it is inevitable that commands scale down in complexity, and so planning systems must determine how to map a low degree of freedom input to the high-degree of freedom, multi-robot system. Motion planning methods for multi-agent systems fall into several broad approach classes. Optimization methods produce high quality plans but are generally time intensive due to the number of decision variables considered. Search based methods likewise face challenges with the large search spaces inherent to coordinating actions between agents. Highly

decentralized or local control and planning approaches, in contrast, scale well with large numbers of agents and their low computation requirements yield fast online responses. However, the very nature of these approaches—that the majority of decision making is handled at an individual level—means that approaches in this class face challenges with coordinating plans between agents or keeping a cohesive “group” identity, as well as challenges guaranteeing plan safety (dynamic and collision constraints). The wide range of approaches, however, allows us to build on existing methods and enable operation in the respective multi-agent domains of theatrics and reconnaissance.

This thesis considers two application domains requiring human-directed, multi-robot planning: that of a theatrical application, and that of a reconnaissance scenario. The theatrical application requires detailed inter-agent coordination, planning for multiple multi-robot groups, and takes place in an open environment with commands from the user occurring on the order of tens of seconds. In the reconnaissance application, a user inputs low-degree of freedom input at high-frequency, via joystick command, necessitating that the planning system determine coordination between agents. This application further considers the motion of a single group of agents (without group splitting or merging) in a highly cluttered environment.

The two applications have numerous similarities: both require the system to interpret user commands into dynamically feasible trajectories, sparing the user from thinking about environmental or inter-robot collision constraints, reconciling the physical state of the system with the desired plans at the time the command is given. This allows for similar system approaches, shown in the parallel system diagrams of Fig. 1.0.2. However, the requirements for high-frequency interac-

tion, need to coordinate member inter-actions without direct user specification, and operation in cluttered environments makes the reconnaissance application more challenging than the theatrical application.

Therefore, this thesis first presents a multi-robot system for theatrical specification and then shows how the assumptions made for a specific class of multi-robot behaviors which give highly detailed instruction for coordination between agents, commands on the orders of seconds, and operation in an open environment are broadened to a multi-robot behavior representation and plan-

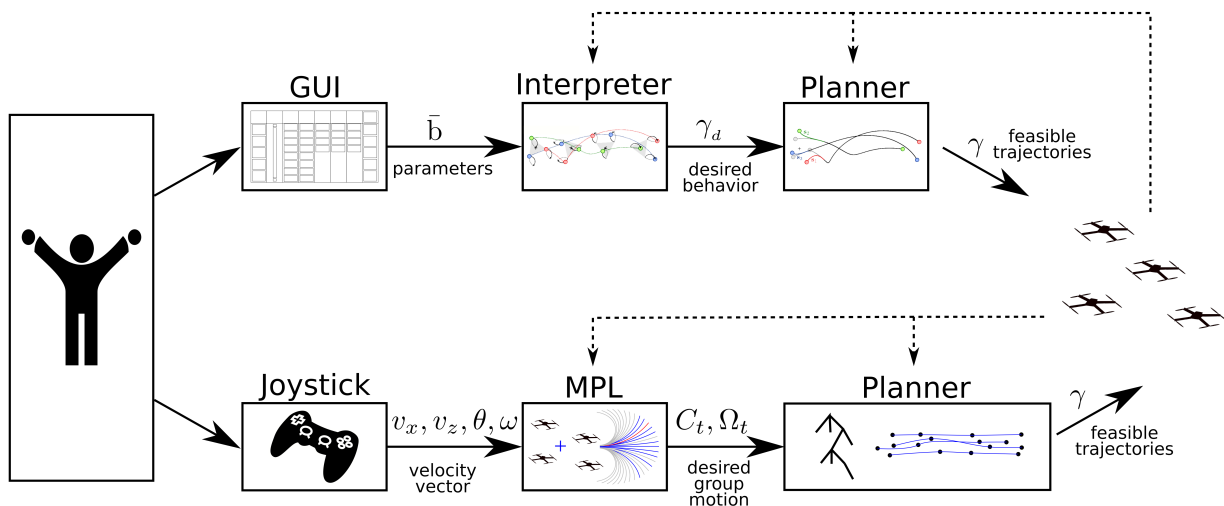


Figure 1.0.2: Two system pipelines used in this thesis. Both allow human user online input for the control of a multi-robot team.

Top line The top pipeline shows a GUI (graphical user interface) where a user can specify parameters, sent to the interpreter to define a desired group behavior. The planner resolves differences between the actual system state and the desired behavior to output dynamically feasible and collision free trajectories for all robots in the group. The parameterized input allows a user to specify detailed coordinated motions online, and is used in a theatrical application where behaviors are specified by a user with a frequency on the order of tens of seconds. Dynamic feasibility resolution via polynomial trajectory interpolation and time scaling works well for the open environments of the theatrical application.

Bottom line The bottom pipeline illustrates how a user may drive a group of robots using a joystick interface. Here, the commands from the operator only specify velocity and heading, and do not describe motions between agents. The user's input is mapped to a motion primitive describing the flight path for the center of the group, and the planner resolves all questions of coordination and collision avoidance between agents and the environment. Joystick input allows for high-frequency user-team interaction at the cost of detailed motion specification. The planning approach leverages offline data to inform search and mitigate the challenge of searching a large action space to rapidly find feasible actions online that allow a team to navigate highly cluttered environments.

ning approach that generalizes across applications, responds to high-frequency commands without inter-agent coordination instruction, and operates in highly cluttered environments to enable human control of a multi-robot group for urban reconnaissance applications.

This thesis address the challenges of online multi-robot planning in highly cluttered environments through the development and validation of a general model and data-driven search approach.

Our contributions, and an overview of document structure, is as follows:

- **Chapter 3** This chapter describes a methodology for inputting multi-agent motion objectives within the context of a theatrical application, and a motion planning system for generating dynamically safe trajectories resolving differences between operator input and feasibility requirements. This chapter highlights both the challenges and online planning requirements for multi-robot systems, and the limitations of current approaches.
- **Chapter 4** We look at extending the approach shown in Chapter 3 by considering an alternate representation for multi-robot behaviors and planning for groups of robots using search. This method allows us to consider environments with obstacles and provides a framework for incorporating experience, which leads to improved search times and solutions.
- **Chapter 5** We show that discrete search over multi-robot actions provides a method for composing behaviors online, and that actions can be provided by data sources, treated here as demonstrations provided by a user. Using datasets provided from the theatrical application of Chapter 3, we show that we can re-create the theatrical behaviors generated through the specialized system with this more general representation and that we can now operate rapidly in highly cluttered environments.
- **Chapter 6** The final chapter focuses on further analysis of the methodology of Chapter 5. Rather than specifically focusing on replicating the specialized capabilities, we use multiple data sources to inform the search heuristic policy, and analyze the methodology to characterize performance over variations in numbers of robots, action resolution and search size set, and performance over environment variations. We conclude by comparing the proposed method to two alternate approaches from the literature for a user-directed application in a urban reconnaissance scenario.

Chapter 2

Background and Related Work

A broad summary of human-directed multi-robot system design is to create a system that (1) accepts human user instruction, (2) translates instruction to planning goals and constraints, and (3) finds trajectories that satisfy feasibility requirements. This thesis addresses all three of these requirements for general multi-robot system design as well as the additional challenges required by the two specific multi-robot applications of a theatrical performance and reconnaissance. This section therefore discusses approaches in the literature that relate to these topics. We begin with discussing the challenges and existing approaches for specifying human intent for multi-robot systems (Sect. 2.1). We then discuss approaches for multi-robot planning and trajectory generation (Sect. 2.2). This is a large area of research with many approaches, and we structure our discussion around local or reactive approaches `sSec:ReactiveMethods`, optimization based approaches `sSec:OptMethods`, and search methods `sSec:SearchMethods`.

Key to our approach is that a user does not command robots individually but addresses robots in teams or groups. In the following discussion of related works, we therefore focus where possible

on group or formation based methods rather than on the related multi-robot problems of routing or scheduling.

2.1 Interpreting human intent for multi-robot systems

For robots to move in response to a user’s intent, a robot system must be able to extract and process motion or task-level information from operator input. The determination of motion requirements from human input is still an active research effort in single-robot domains as well as the emerging field of human–swarm or human–multi-robot interaction. However, the body of literature studying the related questions of what strategy is best used by a human operator to communicate intent to groups of robots, and through what interface methodologies, has grown rapidly in recent years with advances in multi-robot technologies [45]. Human operator input for multi-robot systems has been formulated as:

- selection or tele-operation of leaders in leader-follower formulations through GUI [9], haptic [72, 73], gesture [76], and joystick [90] control;
- using a human operator to perform the function of a “switch” to trigger behavior modes in a hybrid-control formulation through GUI [11, 50] or speech and facial recognition [63];
- allowing the operator to control swarm behaviors such as flocking by performing simulated environment modification—mimicking biologically inspired control methods—through managing artificial attraction or repulsion fields in simulation [44], via drawing interfaces [35], or gesture and “demonstration” through motion-mimicking [3].

These different approaches are all answers to the same question: what combination of interface and input methodology best maps the reduced dimensionality of a user’s input (a word, gesture, or button-click) to the high dimensional, application- and platform-dependent state-space of multi-

robot trajectory generation?

2.2 Multi-robot planning and trajectory generation

2.2.1 Reactive methods

Swarm approaches, or local control methods for individual agents that yield global behavior, are frequently used in computation limited applications. We include in this discussion highly decentralized approaches as frequently employed by reactive control methods, for example, the wide range of approaches stemming from velocity obstacle control methods introduced in [82, 83].

The low computation requirements of these approaches lend themselves well to high frequency interaction applications such as ours, and numerous methods of allowing a human user to direct swarm motion have been pursued [45]. However, local control methods make coordinating the group as a whole highly challenging, a problem generally exacerbated in high clutter environments. Even examples that do seek to encode group behavior at a local level [36] can lead to deadlock between agents and may not provide collision guarantees. Group cohesion and motion guarantees with respect to velocity based control methods have been addressed by integrating motion planning with higher level planning approaches for multi-agent systems in open environments [2], with multiple aerial robots [4], and small numbers of obstacles [5]. However, these approaches employ optimization based methods to resolve constraints, the time and computation limits of which may be prohibitive for our application as previously discussed. We therefore do not consider local control strategies in this work, although these methods could potentially augment the motion planning

portion of the approach.

2.2.2 Optimization methods

Optimization methods that plan trajectories for groups of quadrotors [20, 58, 67] offer numerous advantages. These methods most often generate polynomial time trajectories with respect to collision and dynamic constraints, yielding assured safe plans over the trajectory horizon. These methods further produce optimal solutions which minimize (or maximize) a desired cost function, and costs of interest may include energy [18] or formation parameters [5]. However, optimization approaches are highly time intensive and generally infeasible for applications requiring rapid online re-planning. Optimization complexity increases with the number of decision variables, so approaches which consider individual robots in a coupled fashion become intractable with larger numbers of robots or obstacles [20, 58]. More recently, approaches have sought to limit the number of decision variables by optimizing parameters representing the group of agents subject to linear constraints representing only the local environment [5]; while this has been shown to work in online planning scenarios, the environment representation can be limiting in highly cluttered scenarios and optimizing formation parameters can still be time intensive, making it problematic for high-frequency interaction. While a full optimization problem for all robots is intractable in our operating context, we are able to incorporate optimal motion planning methods for polynomial trajectories [57, 66] within our constraints, and we leverage these methods to generate safe continuous plans from the discrete output of our search approach.

2.2.3 Search methods

Search approaches are frequently used when the number of decision factors in a problem is large. Graph representation coupled with the use of heuristics or other informed selection strategies allow search approaches to tractably refine large decision spaces [77]. With respect to the graph representation (i.e., the chosen discretization or representation), some search methods are also optimal (the path minimizes a cost function) and/or complete (will return a solution or conclude no solution exists in finite time) [85]. Finally, planning dynamic motions based on search solutions has an increased probability (or in some approaches, guarantee) of success [28]. However, coordinating actions between agents during search can be computationally challenging. Planning in the joint configuration space of agents is exponential in the number of agents, making search with even a limited set of individual actions for each robot potentially intractable with the required number of agents or over longer search horizons; search approaches which plan for agents individually must still resolve inter-agent conflicts [81, 85]. Additionally, edge checks can be expensive depending on the constraints considered, for example, when requiring collision or motion planning computation [14]. Finally, it can be difficult to generate good heuristics to inform search [31, 89]. We employ a search based approach to make tractable the large number of options inherent to coordinating a team of robots in complex environments, and discuss our approach for mitigating search complexity and time considerations in the following sections.

Search in multi-robot applications has frequently been used with respect to routing problems, where no formation or group requirements are considered, and goal and destinations across robots are often widely spaced through the environment. They do not generally require coordination in

the same ways that team-based constraints do. For example, only needing to consider planning with respect to other robots when geometrically conflicting [81, 85]. While routing approaches can potentially be used for group planning problems, as both problems seek to move a set of robots to a set of goals, the consideration for maintaining some notion of a collective generates both different constraints and abstractions. In the following discussion of search approaches as applied to the problem we consider in this work, we attempt to focus on search with respect to formation planning where possible, but mention several general multi-robot planning approaches as examples of methods for combating various problems common to general multi-robot applications.

As discussed, one of the largest challenges in multi-robot search is combating the large search space formed by considering options across robots. Sample based approaches seek to combat the large space of multi-robot actions (or configurations) by growing a search tree online, biasing sampling towards perceived high value areas. Monte Carlo methods have shown success in high branching search spaces where little domain information is available, and so have been evaluated for multi-agent planning problems [31, 89] where results indicate that the incorporation of heuristic can greatly improve results. In sample based approaches such as Multi Agent RRT* (MA-RRT*) [14, 41], edge formation may be non-trivial, as samples are taken in the state configuration space and can only be connected to the graph if the sample is judged reachable from some current configuration in the graph. And as with other search methods, incorporation of an informed sample strategy was shown to dramatically effect search performance.

Approaches such as [28] address the problem of finding motion plans based on solutions returned purely from searching a geometric space by searching a sequence of two graphs that con-

sider collision and higher order time derivatives for an agent, respectively. This approach specifically considers the application of rapid online plan generation and successfully finds plans for teams of ten to twenty robots in cluttered workspaces within a time frame of one to two seconds. However, the approach plans for each robot independently following a prioritized ordering strategy and does not consider the joint space of motions for the entire team.

Many multi-agent planning approaches seek to reduce the dimensionality of the decision space by grouping planning elements. For example, approaches such as [6] and [52] consider grouping agent actions as macro-actions in order to coordinate between agents, although the problem type, applications, and time scales in these works are not appropriate for our purposes. The most common abstraction however is the consideration of multiple agents as a group or team, so that decisions may be made for some single team representation. Search specific planning methods that consider a team or group formulation include methods such as [62], which augments the fundamental Dijkstra path finding algorithm with a split or merge decision variable to divide agent groups to sub-groups to route around obstacles. However this decision process is computationally intensive, scales with number of robots, and is too slow for online use. Approaches such as [77] consider explicitly switching leadership designation among a prespecified number of robots in a larger group. This approach requires heuristics based on formation shape and pre-specification of the number of leaders for use in a use MHA* search formulation to route formations of robots around obstacles. This approach reported favorable average search times on the order of one to two seconds for twenty robots, which is promising although slightly time intensive for a high-frequency command requirement, although the need for formation-based heuristics and search scaling with

the number of chosen leaders is less appealing.

To allow a user to rapidly interact with a robot team with a high degree of control yet correspondingly low cognitive burden, we employ two separate input methodologies to meet application-specific requirements. Chapter 3 describes a parameter-based mapping approach for easy human specification of complex coordination policies, while Chapter 5 describes the use of joy-stick selected motion primitives [87, 88] extended to group direction. These approaches trade off command information content for decreased specification time, additionally increasing the burden on the system planner with the demand for fast coordination with little input in the reconnaissance application.

Existing multi-robot policy specification and planning methods have difficulty providing all the required capabilities to meet this increased burden, especially in complex environments. The computational simplicity that makes reaction or swarm approaches fast is a liability for applications that require coordinated responses and safety guarantees. Planning approaches which consider robot actions in relation to each other can produce the safe and coordinated motions we require, but are in general too slow for online application requirements.

Due to the numbers of robots considered in our application, along with the desire for safety assurances and ability to safely coordinate motions between agents, we pursue a centralized planning approach that makes decisions over groups of agents for computational tractability coupled with polynomial trajectory interpolation for rapid generation of dynamically feasible plans. Centralized interpretation of parameter based commands is suited for the open environment of the theatric application, but we generalize this approach to a search based method over discrete group-

level actions for the reconnaissance application. We leverage prior data to enable online search within application time limits, allowing tractable, informed operation in the large search space of multi-robot actions. The following chapters present our system for online human-directed multi-robot control, establishing the value of the proposed methodology with respect to the patterned coordination of multi-robot theatrics and rapid group movement through the cluttered scenarios of multi-robot reconnaissance.

Chapter 3

A system for online behavior specification and execution

In this chapter we define the concept of multi-robot behaviors and develop a system to create dynamically feasible behavior trajectories online in response to user input. Objectives are defined by an expert user and we work in an obstacle free environment with multiple groups of robots.

We allow a human user or high level planner to direct overall system objectives. Objectives are specified during runtime without any prior knowledge of command ordering or timing and are translated by a centralized planner into *behaviors*, dynamically feasible trajectories for groups of robots. We present a parameterized input system for defining objectives online, where parameters are carefully chosen to both be easily understood by humans—easing user specification under the time constraints of online operation—and directly translatable into system-understood requirements. Our approach yields a large yet tractable input space for online behavior specification. Our

approach is fully able to plan for splitting and merging of robot groups as well as the concurrent control of multiple groups, and we present evaluation results for validation and verification of the approach.

The methodology presented is a general approach which might be used across multi-robot applications. However, we choose to specifically look at directing multi-robot behaviors in the context of a theatrical application. Multi-robot theatrics is a multi-million^{1,2} dollar industry in its own right, and can additionally be viewed as a “stepping stone” application from which to explore future work in multi-robot behavior planning. Theatrical applications allow us to prove operation in the relatively straight forward domain of operation in known, non-cluttered and static environments (we extend to operation in cluttered and dynamic environments in Part II). Further, theatrical applications are directly interested in objectives governing robot motions. This can be thought of as a good base case, because many other multi-robot applications might use motion objectives to pursue other metrics of interest. For example, an exploration application might compose motion behaviors to meet an information-based objective. We leave the composition of motion behaviors to meet objectives in other metrics, or specifying objectives using other methods or parameters, to future work.

We therefore present the methodology here in Part I as motivated by use for a theatrical application, and we show hardware results for a user-directed, multi-robot performance. All work described in this section has been completed, and supports the material proposed in Part II. We begin by defining multi-robot *behaviors* in Sect. 3.2 and describe interpreting behaviors to dynam-

¹<https://skytango.com/interest-growing-in-drones-for-entertainment-shows/>

²<https://skytango.com/market-value-of-drone-applications-in-media-entertainment-industry-valued-at-over-8-bn-dollars-says-pwc/>

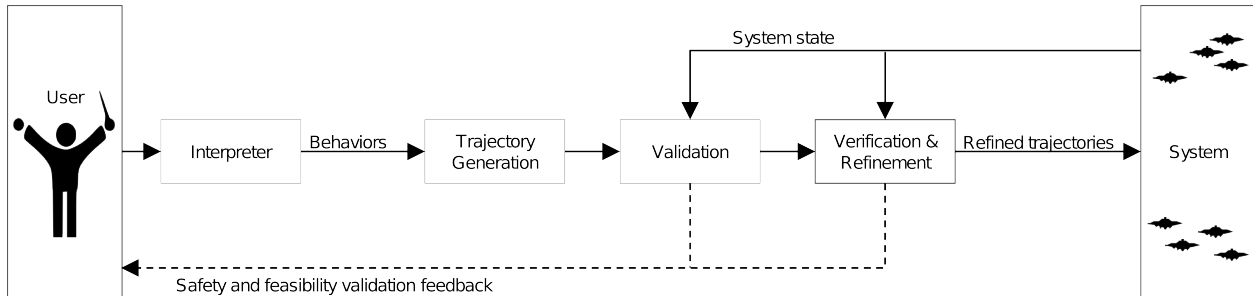


Figure 3.1.1: An overview of the proposed system. User-issued input in the form of *descriptors* is collected by an “Interpreter.” Descriptors, organized into *behaviors*, are output to the trajectory generation subsystem. The proposed multi-robot trajectories exemplifying the user-requested behavior are then checked against the current system state to determine validity. After passing this check, the proposed trajectories are verified for dynamic feasibility. In the event that the proposed behaviors do not meet feasibility or safety constraints, an online search refines trajectories to satisfy safety and feasibility limits.

ically feasible trajectories in Sect. 3.3. We evaluate the methodology in Sect. 3.4.

3.1 System Design

We require a system formulation that can respond to requests in real-time (low latency) with corresponding behaviors that are time-optimal while preserving feasibility and safety.




















To meet these requirements, we therefore propose a centralized planning architecture in order to coordinate intra-group robot trajectories to ensure that group-level motion plans reflect the performer’s theatric intent. A centralized planning methodology additionally ensures that all proposed motion plans across all multi-robot teams are collision free in order to maintain performer safety and audience trust in the performance. Motion plan design builds on prior work in multi-robot trajectory generation [79, 80] in order to determine optimal shape assignments and ensemble motion specifications for user-specified groups of robots. To transition robots in a time optimal manner between desired plans, optimal trajectories are generated by solving an unconstrained quadratic program [66], and trajectory timing refined through online search [37, 66]; the differentially flat

quadrotor model [19] is used to ensure dynamic feasibility given actuator constraints. This proposed formulation seeks to balance low computation times with near-optimal trajectory generation for teams of robots [27].

To enable human–multi-robot interactive theatric performance, we propose a full system that provides a methodology for inputting theatric intent online and translating performer input into dynamically feasible and safe motion plans for teams of robots. We propose a formation-based approach to enable specification of robot team motion in a manner that seeks to reduce the user interaction burden by avoiding individual robot motion specifications. The performer specifies motion descriptors such as formation shape, flight mannerisms, or destination, and the system composes these descriptors into dynamically feasible and safe behaviors. These behaviors then undergo validation and verification checks and if necessary, are modified to meet collision and actuator constraints. The resulting trajectories are distributed to the robot team.

A block diagram of the proposed system, showing user input, behavior generation, validation, and verification and mitigation components is outlined in Fig. 3.1.1. In this section, we step through each block pictured in Fig. 3.1.1 and describe our approach to specifying user intent (Sect. 3.2); generating multi-robot *behaviors* based on user input (Sect. 3.3); validating and verifying (Sect. 3.3.1) behaviors; and mitigating infeasible behaviors by modifying intra-group transitions (Sect. 3.3.2).

Figure 3.2.1: Representative behavior descriptors, b_i , and descriptor sets, B_i , with associated representative values. Descriptor sets are grouped in the table to show the contribution of a descriptor towards an element of the multi-robot trajectory formulation. For example, the “heading” descriptor directly specifies the group trajectory component $\mathbf{S}(t)_{\psi}$ as indicated by the column label.

$\mathbf{C}(t), \mathbf{R}(t)$				$\mathbf{S}(t)_{xyz}$	$\mathbf{S}(t)_{\psi}$	$\mathbf{S}(t)$
robots	time	action	goal	shape	heading	manner
[1 ... N]	t seconds	 hover  takeoff  land  go-to-target  forward-rev  side-side  up-down  circle-target  turn-in-place	 $v*dt$ [Ax, Ay, Az] [Bx, By, Bz] :	 as-is  polygon  line  filled	 as-is x y [Ax, Ay] [Bx, By] :	 fixed  variable  drunk  nervous

3.2 Input Parameterization

While we do not define a formal grammar, specifying theatric intent is similar to answering the questions of “who,” “what,” “where,” “when,” and “how.” The user specifies *descriptors*, b_i , that describe which robots a performer intends to direct, what action the robots should take, the target destination, and any characteristic flight mannerisms that the robots should exhibit. Descriptors are composed into an m -length vector called a *behavior*, $\bar{\mathbf{b}}$. Each behavior descriptor, b_i , may take a discrete number of values, and Fig. 3.2.1 highlights several behavior descriptors and potential value assignments. Denoting B_i as the set of values associated with the behavior descriptor b_i , the total number of potential behaviors achievable by the system is:

$$perm(\bar{\mathbf{b}}) = \prod_{i=1}^m \left(\sum_{k=1}^{K_i} \frac{|B_i|!}{k!(|B_i| - k)!} \right). \quad (3.2.1)$$

Equation 3.2.1 describes the fact that the total number of potential behaviors achievable by the system is the product of the total number of descriptor combinations across descriptor sets. The total number of descriptor combinations possible for a given descriptor set B_i is given by the

summation in Eq. 3.2.1, formulated as the sum of binomial coefficients for set B_i . If only a single descriptor may be chosen from a descriptor set, $K_i = 1$. If a combination of descriptors may be chosen (for example, if choosing k robots from the total number of possible robots), K_i may equal up to $|B_i|$.

As a collection of descriptors drawn from each respective set, a motion behavior contains all the requisite information for defining multi-robot trajectories. This section details each descriptor category and explains how descriptors contribute information to the trajectory formulation, such as trajectory duration, endpoint constraints, and motion characteristics.

Behavior duration: The starting time of a behavior, t_s , is the time at which the system receives the command from the user.³ The “time” descriptor specifies the duration of a behavior, giving ending time t_f , the specified timing duration from start time t_s .

Formation specification: We describe a formation of robots by specifying each robot’s state in a local reference frame [27], which we call the *shape* frame. The positions and headings of each robot in a local reference frame as a function of time t are $\mathbf{s}(t) = [x(t), y(t), z(t), \psi(t)]^T$, $\mathbf{s} \in \mathbb{R}^3 \times SO(2)$, with a vector $\mathbf{S}(t)$ containing all of the positions in the local reference frame of the n robots in the formation: $\mathbf{S}(t) = [\mathbf{s}_1(t), \dots, \mathbf{s}_n(t)]$. The *shape* descriptor specifies desired starting positions, $\mathbf{s}^{xyz}(t_s)$, in the shape frame [27] and the *heading* descriptor specifies $\mathbf{s}^\psi(t)$ for each vehicle. A vehicle’s heading is defined relative to its current frame or oriented toward a target in the inertial frame (a theatrical maneuver called “spotting”).

³In practice, t_s is set to a value slightly ahead of the instruction receipt time to account for planning computation time, allowing robots to transition between trajectories without discontinuities.

Vehicle motions are defined by both the *manner* and *action* descriptors.

Manner: The *manner* descriptor is similar to an adjective in language, giving more information about the flight characteristics that each robot should display during the behavior. Two characteristics of interest to the story are “drunk” and “nervous” mannerisms, which a robot performs by moving along a wobbly course of motion, with slower, larger motions for “drunk” and faster, smaller motions for “nervous.” We represent these motions as bounded polynomial trajectories generated through randomly chosen keypoints obeying timing and distance constraints. Trajectory $\mathbf{s}_n^{xyz}(t)$ for robot n is a spline fit through k keypoints in x , y , and z [66] so that dt_{ij} , the time between each pair of consecutive waypoints i and j , is bounded ($dt_{min} \leq dt_{ij} \leq dt_{max}$) and the sum of all dt 's equals the full time span: $\sum_{i=0, j=i+1}^{i=k-1} dt_{ij} = t_f - t_s$. The position of each keypoint for the n^{th} robot lies within a ball of radius δ centered around the robot's starting position, $\mathbf{s}_n^{xyz}(t_s) \in \mathcal{B}_\delta(\mathbf{s}_n^{xyz}(t_s))$. The bounding values dt_{min} , dt_{max} , and δ are defined on a per-mannerism basis. The “fixed” mannerism denotes regular flight such that the vehicles hold their positions in the local frame throughout the behavior.

All mannerisms, $\mathbf{s}(t)$, must remain within specified limits ensured through appropriate choice of bounding values dt_{min} , dt_{max} , and δ :

$$\mathbf{s}_n^{xyz}(t) \in \mathcal{B}_\delta(\mathbf{s}_n^{xyz}(t_s)), \quad (3.2.2)$$

$$|\dot{\mathbf{s}}_n^{xyz}(t)| \leq v_{lim}, \quad (3.2.3)$$

$$|\ddot{\mathbf{s}}_n^{xyz}(t)| \leq a_{lim}. \quad (3.2.4)$$

Further, the inter-robot clearance distance, d , must be respected at all times, so that for all combinations of robots in $\bar{\mathbf{b}}$:

$$|\mathbf{s}_i^{xyz}(t) - \mathbf{s}_j^{xyz}(t)| \geq d, \forall i, j \in \bar{\mathbf{b}}. \quad (3.2.5)$$

In general, we choose to only allow a user to specify a single mannerism descriptor. However, for appropriate choice of bounding values dt_{min} , dt_{max} , and δ , the combinations of mannerisms $\mathbf{s}(t) = \mathbf{s}_1(t) \oplus \dots \oplus \mathbf{s}_j(t)$ will obey the constraints stated in (3.2.2) – (3.2.5), where \oplus describes a polynomial fit through all keypoints generated for each mannerism \mathbf{s}_j . The vehicles can therefore be “nervous drunks” if required, and the length of the mannerism descriptor set is permitted to be greater than one.

Action: The *action* descriptor specifies the motion of the entire formation. Combined with the *goal* descriptor, we can design a trajectory that moves the local formation reference frame through the inertial frame. The state of each robot is defined in the inertial frame at time t by the vector $\mathbf{x}(t)$, containing position coordinates and heading of the vehicle: $\mathbf{x}(t) = [x(t), y(t), z(t), \psi(t)]^T$, $\mathbf{x} \in \mathbb{R}^3 \times SO(2)$. The state of an n vehicle system is given by $\bar{\mathbf{x}}(t) = [\mathbf{x}_1(t), \dots, \mathbf{x}_n(t)]$. We design smooth trajectories for each state-space dimension via time parameterized polynomials up to an appropriate order to ensure smoothness in the trajectories and their derivatives and satisfy dynamic properties of the vehicle control model.

The position of the origin of the local frame with respect to the inertial frame at time t is $\mathbf{C}(t) = [x(t), y(t), z(t)]^T$, $\mathbf{C}(t) \in \mathbb{R}^3$. We denote $\mathbf{R}(t) \in SO(3)$ as the time varying rotation computed from

the Euler rotations around the inertial x , y , and z axes, $\mathbf{R}(t) = \mathbf{R}_z(t)\mathbf{R}_y(t)\mathbf{R}_x(t)$. To describe a smoothly varying, differentiable rotation, Euler angles are defined as polynomial trajectories [55].

Actions such as “circle-target” or “turn-in-place” specify formation rotations, while periodic actions (“forward-rev,” “side-side,” and “up-down”) define trajectories along the specified axis through waypoints centered about the target location. All actions are composable with all goals and timing specifications to yield valid polynomial trajectories for $\mathbf{C}(t)$ and $\mathbf{R}(t)$.

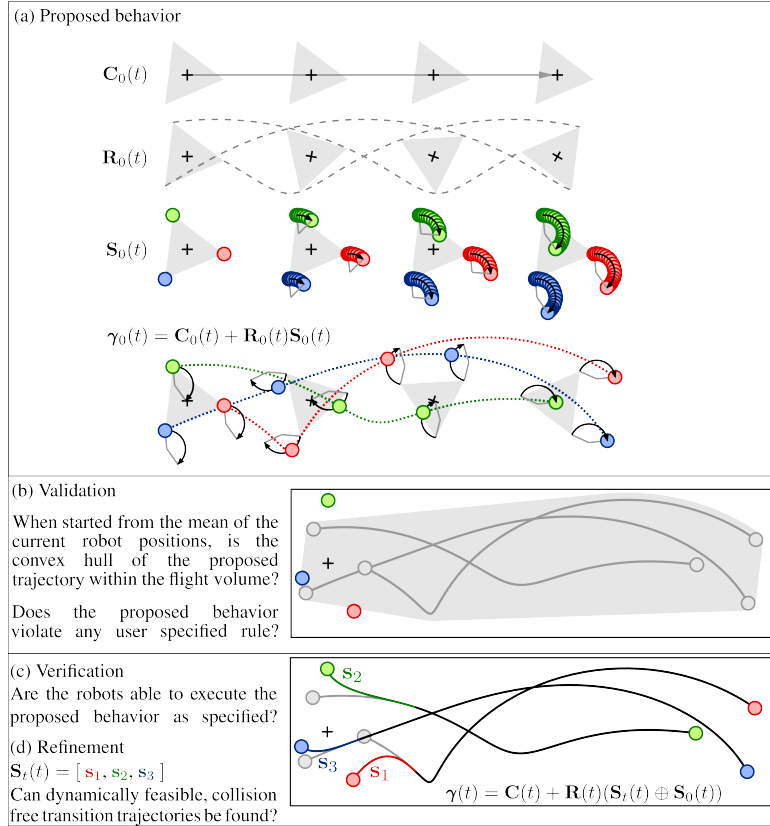
Trajectory initial location: The starting state of a trajectory governing the motion of a formation of robots is established based on the current states of the robots at the time the instruction is specified. Upon instruction receipt, the local coordinate frame in which the formation shape is defined is established with an identity rotation and located at the mean of the specified robots’ current positions and with higher order terms equal to the mean of the robots’ higher order states, leading to the definition of states, $\mathbf{C}(t_s)$ and $\mathbf{R}(t_s)$.

Trajectory ending location: The ending states, $\mathbf{C}(t_f)$ and $\mathbf{R}(t_f)$, are specified by the “goal” and “action” parameters. Goals are defined as (x, y, z) locations in the inertial frame and actions specify motion primitives in relation to those locations.

3.3 Multi-robot Behaviors and Trajectory Planning

Behaviors specify all the information required to generate polynomial trajectories for each robot in a formation. As previously described in Sect. 3.2, the descriptors forming a behavior, taken in conjunction with the states of the specified robots at the time the behavior instruction is issued,

Figure 3.2.2: An illustrated overview of behavior composition, validation, verification, and refinement. Subfigure (a) shows the composition of robot motions in a local shape reference frame with an inertial motion and rotation, illustrating Eq. 3.3.1. Subfigure (b) shows validation of the proposed behavior, depicted with respect to the current robot states. Subfigures (c) and (d) show verification given the current robot states and refined behaviors with dynamically feasible transitions, respectively.



inform first, the desired start, goal, and any intermediate desired states in the local formation frame (as described by the shape, heading, and manner descriptors); second, the desired start, goal, and any intermediate desired states in the global reference frame (as described by the action and goal descriptors); and third, the desired duration of the behavior. In each respective reference frame, local and global, trajectories may be generated between the desired specified states as polynomial functions of time, enforcing smoothness and continuity constraints to an appropriate order based on the robot dynamic model, using common optimal trajectory generation methods [58, 66]. Trajectories for robots in a behavior, $\boldsymbol{\gamma}(t)$, are formed by composing the local shape-frame trajectories, $\mathbf{S}(t)$, with the trajectories describing the motion of the shape frame through the inertial frame, $\mathbf{C}(t)$

and $\mathbf{R}(t)$, as:

$$\boldsymbol{\gamma}_n(t) = \begin{bmatrix} \mathbf{C}(t) + \mathbf{R}(t)\mathbf{s}_n^{xyz}(t) \\ \mathbf{s}_n^\psi(t) \end{bmatrix}, \quad (3.3.1)$$

where $\mathbf{s}_n(t)$ is one of the n local robot trajectories as specified in $\mathbf{S}(t)$, and the superscripts xyz and ψ denote those respective elements of the local state vector. A pictorial representation of an example behavior is shown in Fig. 3.2.2a.

There are primarily two reasons why a behavior may be invalid in an online setting. First, the current vehicle states may lead to a specified behavior colliding with flight volume boundaries. Second, a user may impose state-transition rules that limit descriptor combinations.

We validate a behavior by first confirming that the descriptors, given the system state, do not result in rule-set violations. An allowable specification is defined as $\{\mathbf{C}_0(t), \mathbf{R}_0(t), \mathbf{S}_0(t)\}$ given the current system state and the descriptor specifications and represents the proposed desired behavior. For example, as shown in Fig. 3.2.2(b), the convex hull of the behavior is confirmed to remain within the flight volume and is marked as valid.

3.3.1 Verification and Mitigation

Given a valid desired behavior, we verify that the behavior is realizable by checking the following conditions (in order).

1. The current states of the robots specified by the behavior are sufficiently close to the starting states defined by the desired behavior, i.e., $\bar{\mathbf{x}}(t_s) \simeq \bar{\mathbf{x}}_0(t_s)$.
2. The proposed trajectory accelerations are within the specified limit, $|\ddot{\boldsymbol{\gamma}}(t)| \leq a_{lim}$.

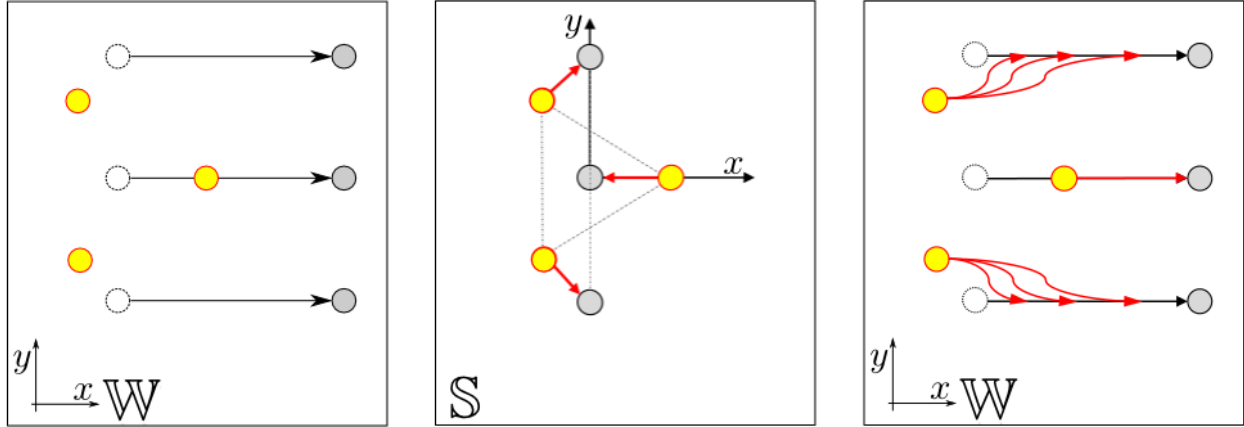
3. The n robots in the behavior maintain an inter-robot spacing greater than or equal to the minimum clearance distance, $|\boldsymbol{\gamma}_i(t) - \boldsymbol{\gamma}_j(t)| \geq d$, for $i, j \in [1, \dots, n]$.

If any condition fails, we immediately proceed to design refined trajectories to mitigate the failure, leading to a dynamically feasible, inter-robot collision-free behavior that remains within the arena volume. We employ the methodology described in [27] in order to mitigate these conditions, and summarize this approach with respect to our application in the following section.

3.3.2 Behavior transitions

A proposed theatric behavior constructed from user input, as described in Sect. 3.2- 3.3, may not meet the three conditions specified at the beginning of this section. We therefore detail a trajectory refinement technique to transition robots from their current states at the time an instruction is issued to the proposed behavior formed as presented in Sect. 3.3. An illustration of this process is shown in Fig. 3.2.2, where the robot’s current states (colored circles) are shown next to the proposed behavior (shown in gray dashed lines), and the solid lines indicate transition trajectories that enable the robots to transition from their current states to the proposed behavior trajectories.

We formulate the transition between behaviors as a goal assignment problem. The methodology builds on techniques related to time optimal trajectory generation and feasibility verification with respect to kinodynamic constraints as well as prior work in the areas of multi-robot formation control. We generate optimal trajectories by solving an unconstrained quadratic program to yield an initial trajectory assuming a conservative time-scale [66]. The trajectory time-scale is further refined through application of the bisection method to find candidate end times [37, 66] that ensure dynamic feasibility given actuator constraints and the differentially flat quadrotor model [19].



(a) A behavior transition problem, shown in the inertial frame, \mathbb{W} . Yellow circles are the robots' current states; black arrows from the dashed white circles to the gray circles represent the proposed behavior trajectories, moving robots from left to right in a line formation.

(b) Optimal assignment of robots from their current positions (yellow) to the desired line formation, goal positions (gray) in the local shape reference frame, \mathbb{S} . The red arrows show the transition trajectories moving robots from their starting positions to their assigned goal locations.

(c) Transition trajectories (red) in the inertial frame, \mathbb{W} . The red arrows depict the same transition trajectory designed in the shape frame performed over different candidate transition times.

Figure 3.3.1: Overview of the behavior transition process. Figure (a) shows the proposed behavior transition relative to the robots current states; (b) shows the assignment process in the local *shape* frame; (c) shows how different proposed time scalings for the transition trajectories affect the overall inertial transition trajectories.

The proposed formulation seeks to balance low computation times with near-optimal trajectory generation for teams of robots. We also ensure inter-robot collision avoidance by leveraging robot prioritization and trajectory time-scaling to avoid collisions given conflicting trajectories [80].

Optimal assignment: To transition robots from their current states to a proposed behavior, individual robots are first assigned to specific formation positions. We assume that robots are homogeneous and interchangeable with no specific preference to goal locations within a formation and require that the region defined by the convex hull of source and goal locations is obstacle free. Additionally, robot start and goal positions must be located d , a predefined minimum safe distance, apart.

Assignment is performed in the local shape reference frame (as depicted in Fig. 3.3.1) and

seeks to minimize the associated traversal time costs. The optimal assignment is computed based on methods detailed in our prior work [80] and seeks to minimize the p -norm of the costs incurred by the team in order to reach the goal configuration,

$$\phi^* \underset{\phi}{\operatorname{argmin}} = \left(\sum_{i \in I_N} \|P(s_i, g_{\phi_i})\|^p \right)^{\frac{1}{p}}, \quad (3.3.2)$$

where I_N is the index set of the robots in the group and s_i and g_{ϕ_i} correspond to the initial and optimally assigned goal configurations of the i^{th} robot, respectively. In this work, we choose to minimize the total distance traveled by the robots and thus let $p = 2$. The optimal assignment is computed via the Hungarian algorithm with $O(N^3)$ computational complexity [47].

Given the start and goal assignments, we can then use a conservative time duration dt_t to compute transition trajectories $\mathbf{S}_t(t)$ in the local shape frame for the time period from t_s to $t_t = t_s + dt_t$.

Feasible trajectory generation: The proposed transition trajectories $\mathbf{S}_t(t)$ are combined with proposed behavior trajectory components $\mathbf{C}_0(t)$ and $\mathbf{R}_0(t)$ over the transition time period t_s to t_t per (3.3.1) to yield individual robot trajectories to be executed by the team of robots in order to transition between shapes. However, prior to transmitting the desired trajectory to each robot, we ensure that each trajectory does not require motions that exceed platform actuator constraints. To this end, we compute the maximum mass normalized thrusts required by each trajectory $\boldsymbol{\gamma}_n$ based on the model [19] and scale the shape transition trajectory duration, dt_t , accordingly so as to ensure feasibility for all systems.

Alternatively, we note that for visual appeal it is preferable that the robots rapidly transition between shapes. Therefore, if the resulting transition trajectories are overly conservative, we pursue a minimum transition time to enable rapid and feasible shape transitions:

$$\text{minimize: } t_t \quad (3.3.3)$$

$$\text{subject to: } -T_{max} \leq \ddot{\boldsymbol{\gamma}}(t) \leq T_{max} , \quad (3.3.4)$$

with $t \in [t_s, t_t]$, T_{max} as the maximum allowable mass normalized thrust, and

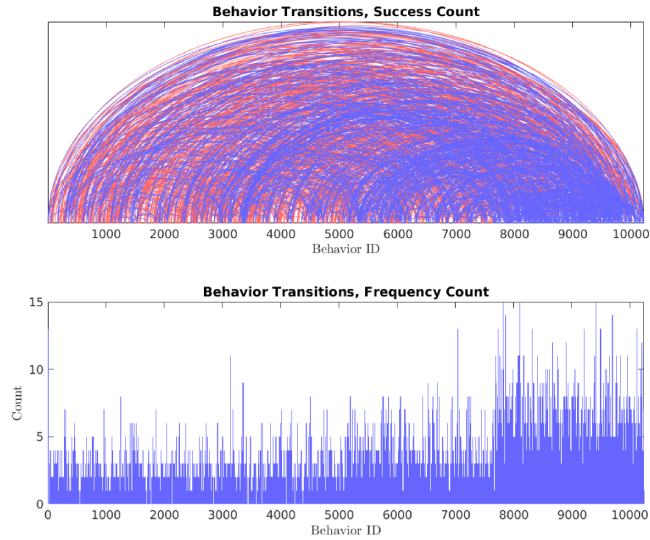
$$\ddot{\boldsymbol{\gamma}}(t) = \ddot{\mathbf{C}}(t) + \ddot{\mathbf{R}}(t)\mathbf{s}(t) + 2\dot{\mathbf{R}}(t)\dot{\mathbf{s}}(t) + \mathbf{R}(t)\ddot{\mathbf{s}}(t) . \quad (3.3.5)$$

We solve this minimization problem online via a bisection line search [24], computing the corresponding acceleration time-scale for each candidate time, t_{ts} , and update the trajectory duration upon termination ($t_t \leftarrow t_{ts}$).

Online search for minimal-time transitions: Given a minimum time transition for all robots, we perform a safety check to ensure that all trajectories preserve a minimum separation distance between robots. If a minimum separation distance is not preserved, prioritization and time-scaling techniques [80] are applied according to an assigned ordering derived from the robot start and goal positions with respect to the shape specification.

Given the prioritization order, each robot trajectory is checked for collisions against trajectories of higher priority robots. In the event that a collision occurs between robots, the trajectory of the

Figure 3.3.2: Plots showing coverage over representative behavior descriptor combinations. Behaviors are validated across varying numbers of robots, with instructions issued at randomly chosen time intervals. Success indicates that the descriptor combination produces a valid behavior and the system is able to interpret, refine, and transform the behavior into a dynamically feasible, collision-free trajectory. Top plot: Arcs describe transition success rates between behaviors, where blue and red correspond to success and failure, respectively. Bottom plot: Behavior validation count. These figures were generated from 48,000 online issued behaviors in simulation, as discussed in Sect. 3.4.1.



lower prioritized robot is assigned a small, positive time offset to avoid collision with the robot of higher priority. This process is repeated iteratively until no collisions exist between the given robot and all higher priority robots. This prioritization results in collision free trajectories for all robots in the formation relative to the minimum transition time of the highest priority robot.

3.3.3 Offline verification

System verification is the process of analyzing a system for desired properties, to give evidence that the system meets the desired requirements. One approach to system verification is through formal methods, mathematically based techniques used to reason about systems and their performance [23, 32, 46]. State of art formal methods for multi-robot applications include approaches such as those based on Linear Temporal Logic [26, 70] or satisfiability modulo theory [71]. These approaches, however, have drawbacks which limit their use for our application. Approaches [26,

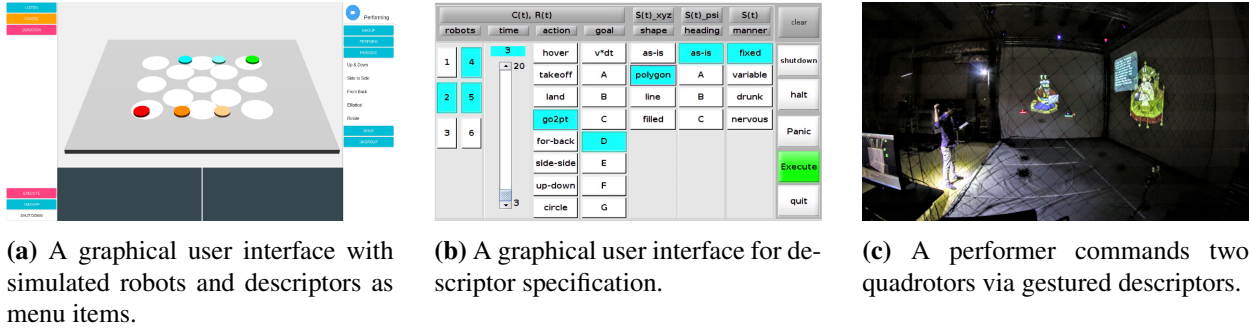


Figure 3.3.3: Interface examples used to convey user-specified descriptors to the multi-robot system. The GUI (a) allows a user to click on simulated robots and select behavior descriptors from organized menus. The GUI (b) shows categorized descriptors as depicted in Fig. 3.2.1. Alternatively, a performer can command robots using gestured descriptors, as shown in (c). In (c), a motion capture system tracks reflective markers on the performer, and this motion data is passed to a gesture recognition system to identify gestured descriptors that are sequentially combined to specify a behavior.

[70, 71] cite computation times roughly on the order of minutes for problems using two or more robots, which is not conducive to online use or responsive interaction with a human user. While [71] presents relaxations which can be used to compute sub-optimal trajectories at faster time scales (problems for ten to twenty robots can be solved on the order of 1-2.5 minutes), these timescales and the use of discretized motion primitives and sub-optimal trajectories still present drawbacks within the context of our application, where we seek to follow time-optimal trajectories for visual appeal.

In the event that formal verification is not viable, statistical verification through model checking and offline simulation is commonly performed to give quantitative insight into system performance [23, 32, 49]. We therefore choose to leverage offline simulation using a high fidelity dynamic simulation environment, simulating all vehicle dynamics including motor response times, across a large number of trials to verify all descriptor combinations assuming a discretization of the state-space of the system that approximately covers all possible starting and ending states within the flight volume. We depict the results of these offline trials in Fig. 3.3.2 and detail both the number

of times a descriptor combination is tested and the number of successful behavior transitions. A behavior transition is considered successful if:

1. The descriptor combination forms a valid $\{C, R, S\}$ tuple, meaning the code implementation is error-free;
2. The descriptor combination does not violate a user specified rule; and
3. A dynamically feasible, inter-robot collision-free trajectory is generated from the specified behavior input.

The resulting transition table is employed online to assist in performing fast online validation. Behaviors with intermediate success rates frequently fail due to instruction timing. Therefore, we may choose to use this validation table as a conservative heuristic, and rather than check every online instruction, reject behavior transitions with success rates below a cutoff value.

3.4 Evaluation and Results

3.4.1 Evaluation: Robustness and coverage

We evaluate the proposed approach through both simulation and hardware experiments. This section shows the robustness and coverage of the proposed planner through extensive dynamic simulation, issuing approximately 48,000 randomly generated behaviors. We show that all plans obey safety and feasibility constraints, and we illustrate timing effects as the system scales between 1 and 10 robots.

We perform evaluation in a high fidelity dynamic simulation environment, simulating all vehicle dynamics including motor response times, to show the robustness of our approach and the associated coverage over the space of behaviors. All behavior instructions were issued at random

times during the course of currently executing behaviors. This required the system to generate dynamically feasible and safe transition trajectories given the (randomly chosen) current system state, or recognize that the transition given the current system state was infeasible. Figure 3.3.2, as described in Sect. 3.3.3, shows the coverage results over 48,000 randomly issued behaviors. This plot reports the number of times a behavior is generated as well as the success rate of the behavior transition.

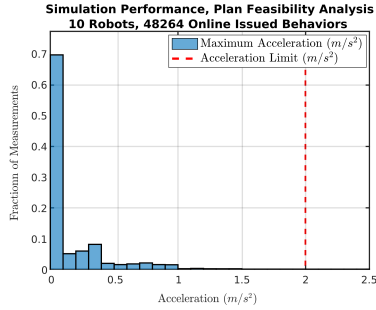
Figure 3.4.1 shows that the planner always generates dynamically feasible and safe plans for valid behavior transitions. All motion plan accelerations remain below the specified limit (Fig. 3.4.1a), and all plans maintain safe inter-robot clearances (Fig. 3.4.1b). We further report the scaling properties of the methodology with increasing numbers of robots in Fig. 3.4.1c, showing how portions of the approach scale, as percentages of total computation time, with increasing numbers of robots. While the computation time required by a centralized approach will increase as behaviors are planned for additional numbers of robots, the presented methodology is well able to handle the numbers of robots required by the theatric presentation even when run as an unoptimized MATLAB implementation.

3.4.2 Evaluation: Hardware Experiments

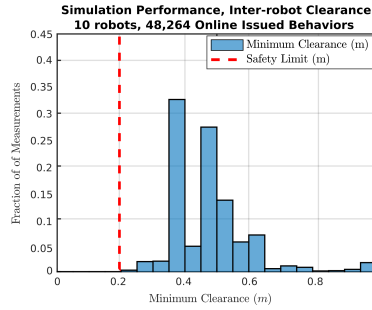
We verify simulation results through hardware experiments, using the CrazyFlie⁴ platform and software framework [64].

Full evaluation over all simulated behaviors is not feasible in hardware. We therefore verify simulation results by running 100 random behavior transitions in hardware on 10 quadrotors. Each

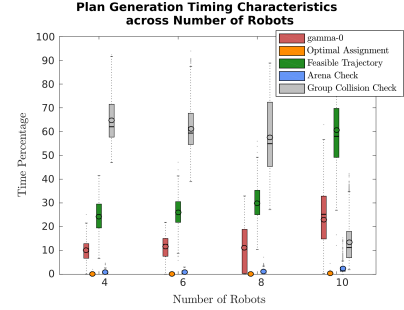
⁴<https://www.bitcraze.io/crazyflie-2/>



(a) Dynamic feasibility: Histogram of acceleration measurements, taken every 0.1 seconds, of the acceleration required by the generated trajectories for all robots over 48,000 online issued instructions. All accelerations are below the specified acceleration limit.

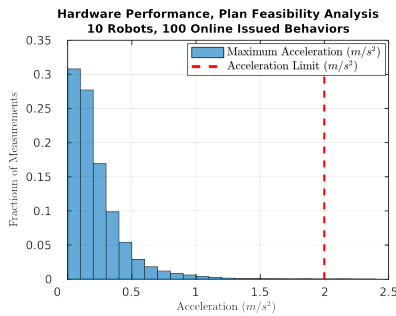


(b) Safety: Histogram of robot-robot distance measurements, taken every 0.1 seconds, across all robots over 48,000 online issued instructions. All inter-robot clearance distances are above the specified safety threshold.

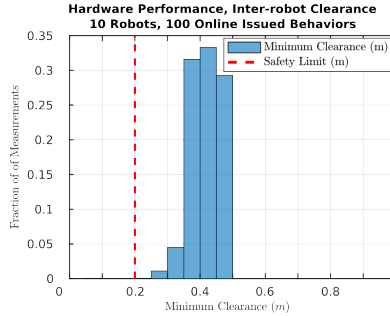


(c) Timing characterization of various stages of the planning strategy and scaling properties given increasing robot numbers.

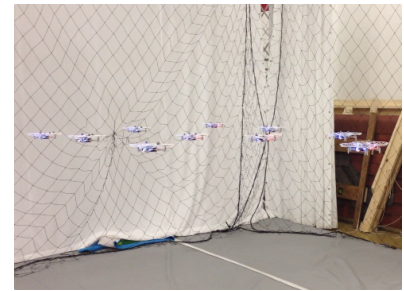
Figure 3.4.1: Dynamic feasibility, inter-robot clearance distance, and timing properties collected over 48,000 online-issued behavior instructions in simulation. Each online issued instruction for a behavior required the computation and execution of a dynamically feasible transition.



(a) Dynamic feasibility: Histogram of acceleration measurements, taken every 0.1 seconds, of the measured acceleration exhibited by 10 quadrotors over 100 online issued instructions. All accelerations are below the specified acceleration limit.



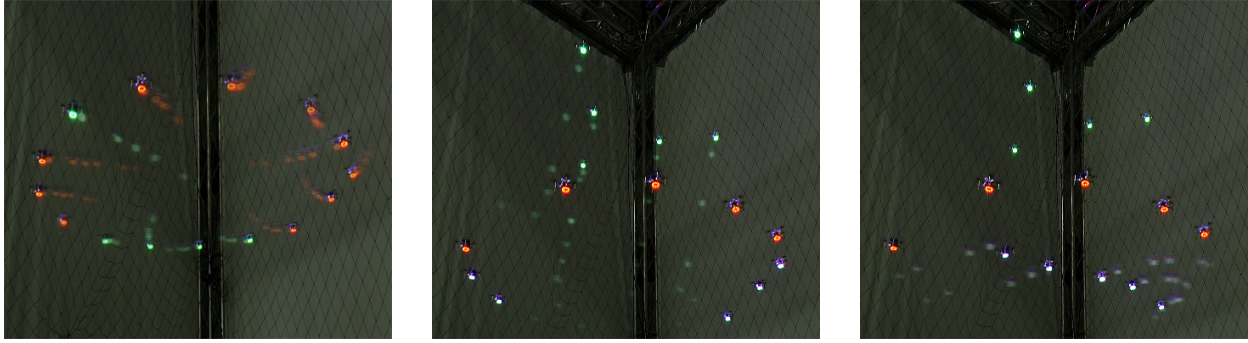
(b) Safety: Histogram of measured robot-robot distances, taken every 0.1 seconds, across 10 quadrotors responding to 100 online issued instructions. All inter-robot clearance distances are above the specified safety threshold.



(c) Image of 10 quadrotors flying in formation during a sequence of 100 online-issued behaviors.

Figure 3.4.2: Dynamic feasibility (a) and inter-robot clearance distances (b) measured over the course of 100 online-issued behavior instructions (each requiring the computation and execution of a dynamically feasible transition) to 10 quadrotors flying in a motion capture arena (c).

individual behavior was performed for a random amount of time typically lasting from between 30 seconds to one minute in length, before a currently executing behavior was interrupted with a new behavior instruction. The Crazyflie platforms were used to validate the methodology because due



(a) Two groups of quadrotors (green and red) merge to form a circular formation of fifteen quadrotors.

(b) A group of 5 quadrotors (green) leave a circle formation and move to a new location as a triangle formation.

(c) A further group of 5 quadrotors (blue) split from their previous group and merge to form a line formation.

Figure 3.4.3: Online behavior execution for multi-group shape formation for a fifteen quadrotor ensemble.

to their small size, a team of ten vehicles was able to execute multi-group behaviors in a limited motion capture volume. We note that the flight time of the Crazyflie robot platforms, however, was between only four and six minutes given the energy drain of the LEDs on the batteries and the aggressiveness of chosen behaviors. To validate 100 behavior transitions over a 10-robot team, we therefore flew the team in over ten trials, performing over 100 battery changes across all vehicles, for a total in-air flight time of approximately one hour.

Finally, we perform a sequence of behaviors designed to highlight system features including group splitting, merging, and complex formation changes using 15 quadrotors.

A photo of all ten quadrotors in formation, performing a representative behavior from the trials, is shown in Fig. 3.4.2c. Figures 3.4.2a and 3.4.2b present the accelerations and minimum clearance distances exhibited by all quadrotors over the full flight time composed of all trials. These hardware results verify the simulation experiments. We observe that the reported vehicle accelerations remained under our planning acceleration limit, and all clearance distances between vehicles remained within the stated limits. Hardware experimental data are consistent with simulation results



Figure 3.4.4: Photos from a theatric performance. Clockwise, from top left: User practicing gesture-based input; two groups forming lines; one quadrotor performing a solo; red team transitions across from the blue team; bottom row: two groups of three quadrotors in triangle formations circle each other by performing a rotation maneuver as a formation of six.

and confirm our dynamic simulation trials.

While the 100 behavior transition results reported in Figs. 3.4.2a and 3.4.2b were randomly chosen to cover the behavior input space, a final demonstration of planner capability via system performance was performed using fifteen quadrotor platforms. For this performance, a set of complex behaviors was chosen which included splitting and merging of groups and performing periodic maneuvers with varied flight formations. These instructions were issued online to the fifteen robot fleet to demonstrate the scalability of the approach and highlight the system’s performance abilities. This performance is shown in an accompanying video, available online.⁵

Additionally, timelapsed images of representative multi-group merging and splitting choreographies performed by the 15-robot team is shown in Fig. 3.4.3. We additionally show images from a user-directed theatric performance (Fig. 3.4.4), where a performer employed the described system to direct varying numbers of robots in a three act narrative.

⁵<http://www.andrew.cmu.edu/user/ecappo/AURO17.mp4>

Chapter 4

Formation planning with experience

In this work, we present a Monte Carlo Tree Search (MCTS) formulation to find sequences of collision-free, multi-robot formations online. While using discrete search for formation planning might at first appear intractable for online use or suboptimal due to the discrete restriction of solution options, we show that search can actually offer advantages over continuous optimization formulations and that the presented formulation of MCTS can yield fast, low cost solutions for online execution, offsetting the disadvantage of combinatoric search over discrete space. This paper shows the performance of the proposed methodology compared to existing optimization approaches in an illustrative environment and, using a full tree search over all possible formation options as a baseline, we show that we can choose search parameters to achieve fast online performance within a defined epsilon of optimal; leverage offline training to improve online performance; and transfer a trained graph across environments.

4.1 Introduction

Multi-robot planning algorithms often plan tasks for groups of robots acting in concert, for either the computational convenience of subdividing large teams of agents and/or because coordinating motions among a collection of robots naturally fits the task structure of the application. Group planning is found across a wide range of multi-robot applications. Examples include carrying objects [5], performing escort missions [7] or sensor coverage [21, 25], and depicting characters or figures in entertainment applications [1, 15, 16]. Coordinating groups of robots is often formulated through spatial constraints: robots are directed to form specific geometric shapes or obey inter-robot distance constraints, and these spatially specified groupings are commonly called *formations*. The problem of determining robot formations and moving formations through an environment is *formation planning* [53, 61].

Planning for formations of robots to avoid collisions between both obstacles and robots can be challenging because computational complexity scales with the number of agents and planning time is limited for applications which require online updates. Planning to move formations of robots through an environment becomes more difficult with increased numbers of obstacles or geometrically complex free space because robots may not be able to reach or maintain the desired shape.

In this work, we are specifically interested in planning for formations of robots in complex and cluttered environments. We approach the problem with the assumption that a high level planner or human operator has specified a desired formation and goal destination for a group of robots. We therefore wish to determine how to modify a formation to meet environment constraints while

remaining as close as possible to the desired behavior. This may be stated as an optimization problem: what formation at any given point along a planned path best minimizes a cost function between the candidate formation and the desired formation, subject to environmental constraints?

This optimization formulation is how state of art methods approach the formation planning problem. The approach of [5] formulates a sequential convex optimization over parameters governing formation design. The approach minimizes a cost metric in terms of formation translation, rotation, and scale and uses the surfaces of obstacle-free regions defined as convex polytopes to constrain the formation to the defined free space. While continuous optimization methods have generally been regarded as computationally complex, the authors show that the presented formulation finds solutions over a local time horizon fast enough for online use.

However, restricting formations to lie within polytopes can limit solutions by excluding obstacle-free space outside the polytopes regions, and finding obstacle-free polytopes, even locally, can be computationally expensive. These drawbacks become more noticeable in geometrically complex environments: larger polytopes may neglect to capture regions of free space left between objects, or conversely, a greater number of smaller polytopes may be required to more accurately represent complicated free space geometry.

In this work, we consider the practicality of using discrete search as a viable method for online formation planning. While limiting solutions to a pre-specified set of discrete options and the combinatoric nature of the search space are obvious drawbacks to the approach, we show that these limitations may be overcome by leveraging experience to learn the value of search sequences, and that the proposed methodology offers several advantages over existing approaches: solutions are

not limited to polygonal bounding volumes, the method is able to provide higher resolution with respect to the environment for the same amount of computation time compared to optimization approaches, and our formulation is able to specifically consider transition cost between formations in the cost of the solution.

The contribution of this paper is the formulation and input of domain knowledge to a learning based search formulation—specifically Monte Carlo Tree Search (MCTS)—to yield a tractable methodology for online multi-robot formation planning. Our results demonstrate that a sample-limited MCTS implementation is able to leverage experience to improve performance, and that performance holds across environments. These two findings demonstrate the applicability of the proposed approach for online use: a large search tree may be trained offline and used online across varying environments with specified sample limits to find solutions within calculated performance bounds.

4.2 Formation Planning, an Example

We first present an example scenario to introduce relevant concepts and elaborate on facets of formation planning that motivate a search-based methodology. In our example, a square formation of robots travels through a non-smooth corridor, shown in two dimensions in Fig. 4.1.1a. The environment is represented as a voxelized occupancy grid. Non-smooth corridors are chosen as a common environment of interest, such as when considering cars, dumpsters, or signs bordering buildings in urban streets. Applications which require operation in these environments, such as exploration, reconnaissance, and survey or inspection tasks, often involve observation of objects

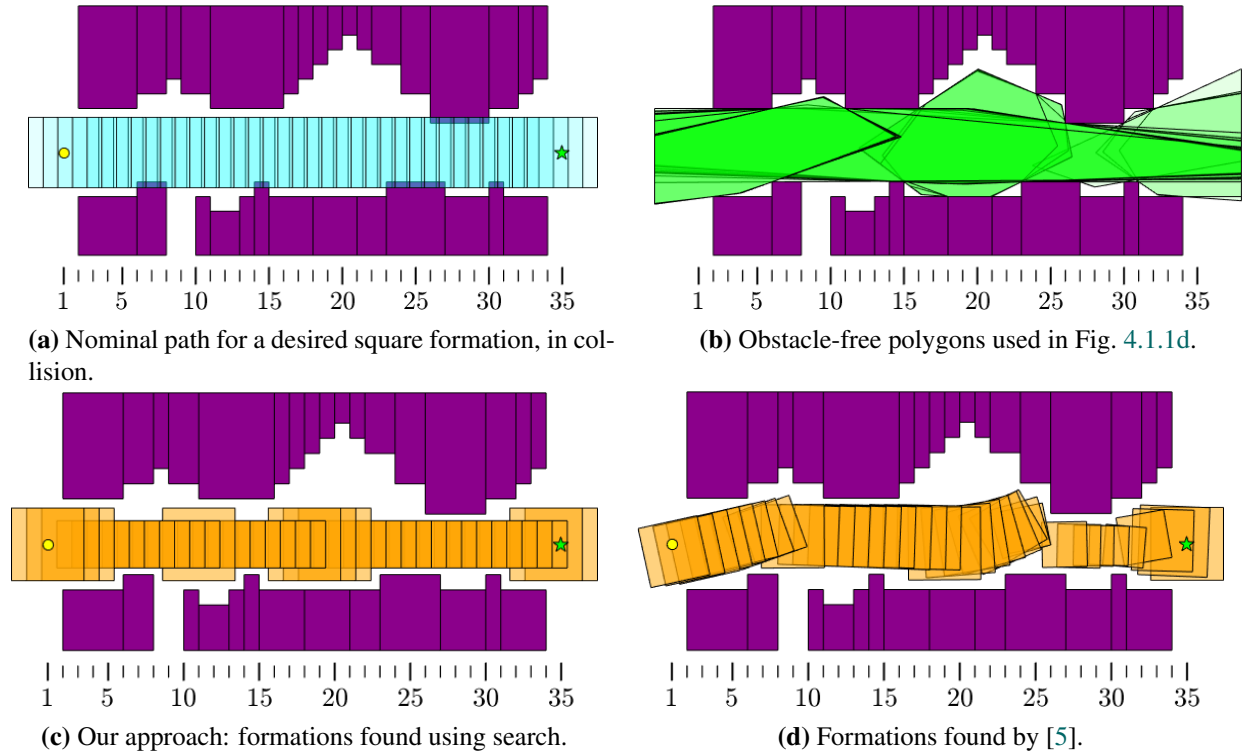


Figure 4.1.1: These four figures illustrate two example solutions for the same formation planning problem. Fig. 4.1.1a shows the desired path for a square formation colliding with the environment. Fig. 4.1.1d shows the output of [5] that depends upon convex decompositions of the free space (Fig. 4.1.1b); Fig. 4.1.1c shows a solution found using our approach, searching a discrete set of pre-specified formation choices.

or surfaces, necessitating planning paths in close proximity to environment features.

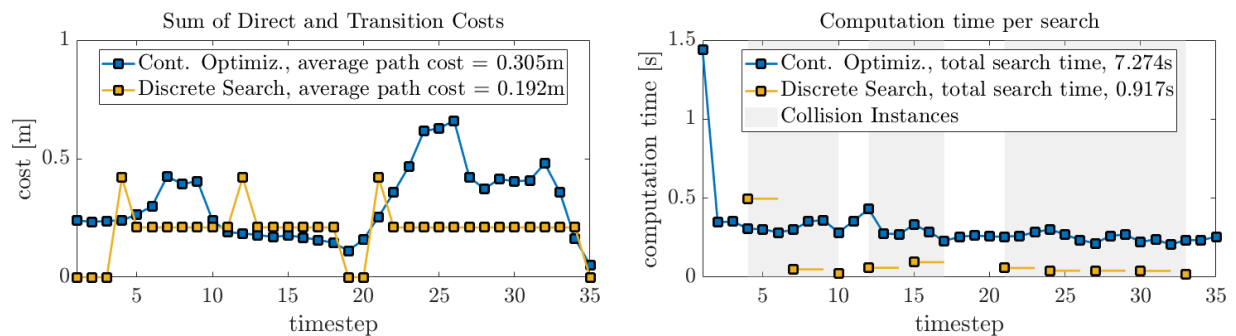
Figure 4.1.1a shows a square formation (blue) moving left to right through a corridor, colliding at several places with the environment. We consider two approaches: (1) The sequence of formations returned from an optimization-based approach [5] (orange, Fig. 4.1.1d) relies on free space defined as convex polygons (green, Fig. 4.1.1b); (2) Our proposed approach of search, Fig. 4.1.1c, shows a sequence of formations (orange) composed from a set of discrete shapes.

Two drawbacks of [5] are apparent in this example, stemming from minimizing a greedy cost based on local, polygonal-bounded free space. First, the solution is limited to space defined by the convex polygons, whereas search is able to better utilize free space, shown at timesteps 11, 18, and

20 of Fig. 4.1.1c. Second, the transition cost between formations across time is not considered, so there can be large differences between consecutive time steps, especially when solutions fall within different polygons (see timesteps 8 and 26 in Fig. 4.1.1d).

While deferring a mathematical description of cost to Sect. 4.3.4, we examine this idea of similarity–or dissimilarity–between two formations by considering the average displacement of a robot in a candidate formation from its specified position in a reference formation. Figures 4.2.1a–Fig. 4.2.1b show displacement measures for the solutions examined in this example.

Continuous optimization can, as expected, provide non-colliding formations that minimally displace robots from their desired positions at a single timestep and are of lower cost than discrete options (Fig. 4.2.1a, timesteps 12-17). However, because the approach moves a formation into polygonal-bounded freespace, it can also move robots away from their original desired positions. When combined with a measure of cost associating with transitioning between formations at successive timesteps, the greedy minimization performed by the optimization methodology can



(a) Combined cost: sum of average robot displacements at a timestep and between timesteps.

(b) Methodology timing characteristics. (Search timing is shown spanning the covered timesteps.)

Figure 4.2.1: We calculate the cost of a formation by considering the average distance a robot is displaced relative to a reference formation. Fig. 4.2.1a shows the calculated cost for a formation at timestep t as the sum of two quantities: the direct cost (between the solution formation at t and the desired formation at t), and the transition cost (between the solution formation at t and the previous solution formation at $t - 1$). Fig. 4.2.1b shows timing characteristics of the two approaches, implemented in MATLAB.

be seen to provide far higher cost solutions than a search strategy (even over discrete options) that considers the cost of transitioning between formations. This is most noticeable at timesteps 9, 19-27, and 32-34 of Fig. 4.2.1a.

When searching a finite, discrete set of options, B , the number of possible solutions for a sequence of timesteps, D , is B^D . Evaluating a large number of solutions may be costly depending on the chosen evaluation method, but by restricting the solution space to a finite set over a chosen horizon, it is feasible to reason over multiple timesteps during a single application of the method. For the same computation time, discrete search can therefore provide a solution with D times the resolution of the optimization approach, which requires application at every timestep. Conversely, search can provide the same resolution solution roughly D times faster. Fig. 4.2.1b shows timing characteristics for MATLAB implementations of the methods considered in this example.

Non-exclusion of free space, reasoning over multiple timesteps, and the resolution-to-time tradeoff motivate the use of search over discrete, pre-specified formation options as a potential planning approach. While uninformed search over a large set of options is impractical for online use, methods such as Monte Carlo Tree Search (MCTS) have proven highly successful in problems with large branching factors [12, 75]. By storing information about previously tested sequences, MCTS makes use of experience to inform its selection policy, allowing it to find better cost solutions with successive rollouts. This learning aspect means that the formulation of MCTS we propose in this work is able to balance a large, combinatoric search space with online operating requirements.

In this work, we propose a formulation of MCTS for formation planning. In Sect. 4.3, we de-

scribe our chosen search set, cost function, and how these are used in MCTS. In Sect. 4.4, we show simulation results to explore and quantify aspects of the questions we raise here, including analyzing the relationship between the size of a search set and search performance. The experiments in Sect. 4.4 show that a sample-limited MCTS implementation is able to improve performance with increased experience and that the performance of a search tree holds across environments, meaning that experience accumulated in one environment may be reused in a new environment. These results allow the use of a limited search budget to achieve fast online search within calculated performance bounds.

4.3 Proposed Methodology

We propose a formulation of Monte Carlo Tree Search (MCTS) that makes use of previous experience to perform formation planning for multi-robot formations. Our approach first defines a set of discrete *shape transforms* which modify a formation of robots by a combination of rotation, scaling, and shearing operations on a single base shape for the nominal path (Fig. 4.1.1a). We then employ MCTS to find a series of shape transforms that minimize a cost metric based on the desired formation at a timestep and the cost of changing formations between timesteps. This section describes the mathematical background, problem setup, and search details of the approach.

4.3.1 Robots, Formations, and Behaviors

A common description of multi-robot *behaviors*—position and time references for a group of robots that meet a specified action or command reference—involves specifying relationships for a

group, formation, or *shape* of robots with respect to a local reference frame and then controlling this local frame with respect to a global reference [15, 16, 78, 91]. We choose to use this representation because of its wide applicability to general multi-robot tasks, such as moving groups of robots to desired destinations in chosen formations over specified time durations, including incorporating desired motion characteristics [15].

For each robot in a group of N robots, $i \in \mathcal{I} = \{1, \dots, n\} \subset N$, a robot's position is denoted by $\bar{x}^i = [x, y, z]^T$, $\bar{x} \in \mathbb{R}^3$. A formation of robots is a group of robots having a *shape*, \mathbf{S} , with each robot in the group having a position $\bar{s} = [s_x, s_y, s_z]^T$, $\bar{s} \in \mathbb{R}^3$ in a local reference frame, \mathbb{S} . A shape of N robots, each with time varying positions within the shape, \bar{s}_t^i , is described by a vector of their respective positions: $\mathbf{S}_t = [\bar{s}_t^1, \dots, \bar{s}_t^n]$.

Using the behavior description of [15, 16] with discrete notation, the local reference frame \mathbb{S} is related to the world frame, \mathbb{W} , by a time varying rotation matrix \mathbf{R}_t , $\mathbf{R} \in \text{SO}(3)$, and position offset, $\mathbf{C}_t \in \mathbb{R}^3$. The position of a robot in the world frame can be found by composing the motion of the shape frame \mathbb{S} relative to the world frame as described by \mathbf{C} and \mathbf{R} , such that $\mathbf{x}_t^i = \mathbf{C}_t + \mathbf{R}_t \bar{s}_t^i$. A multi-robot *behavior* is then the specified positions of a shape of robots over time $t \in \{1, T\}$ as:

$$\bar{\gamma}_t = \mathbf{C}_t + \mathbf{R}_t \mathbf{S}_t. \quad (4.3.1)$$

4.3.2 Environment and Collision Checking

We choose to use a voxel-based environment representation, letting $\mathcal{O} \subset \mathbb{R}^3$ be the set of static obstacles (occupied voxel cells) in environment E . If $\mathcal{P}(\bar{\gamma}_t)$ is the set of voxels occupied by robots

for formation $(\bar{\gamma}_t)$, then those robots are in collision if $\mathcal{P}(\bar{\gamma}_t) \cap \mathcal{O} \neq \{\}$.

4.3.3 Definition of Shape Transforms

The behaviors described by Eqn. (4.3.1) relate a potentially time varying formation of robots to the world frame using a time varying rotation reference. We additionally define a behavior representation that groups the time varying elements of a formation and its inertial relationship into a single term as:

$$\bar{\gamma}_t' = \mathbf{C}_t + \mathbf{A}_t \mathbf{S}_0, \quad (4.3.2)$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ represents an affine transform, a more general representation that includes rotation as well as scale and shear, and \mathbf{S}_0 represents a nominal formation. This definition allows us to view behaviors as local, *shape-transforming* modifications made to a nominal formation that defines a fixed relationship between robots, and assists us in formulating our search approach for non-colliding formations. Upon finding a behavior $\bar{\gamma}_t$ specified by a high level planner or human user, as in [15, 16], to be in collision, we propose to find the closest non-colliding behavior $\bar{\gamma}_t'$, where $\mathcal{P}(\bar{\gamma}_t') \cap \mathcal{O} = \{\}$ for all $t \in \{1, T\}$, by searching over a set of *shape transforms*.

A *shape transform*, $\mathbf{A} \in \mathbb{R}^{3 \times 3}$, is then defined as a 12-parameter affine transform which modifies a formation of robots by a combination of rotation $\mathbf{R} \in \text{SO}(3)$, scaling $\mathbf{Sc} \in \mathbb{R}^{3 \times 3}$, and shearing

operations $\mathbf{Sh} \in \mathbb{R}^{3 \times 3}$.

$$\mathbf{A} = [\mathbf{Sc}][\mathbf{Sh}][\mathbf{R}] \quad (4.3.3)$$

$$\mathbf{Sc} = \begin{bmatrix} sc_x & 0 & 0 \\ 0 & sc_y & 0 \\ 0 & 0 & sc_z \end{bmatrix} \quad (4.3.4)$$

$$\mathbf{Sh} = \begin{bmatrix} 1 & sh_x^y & sh_x^z \\ sh_y^x & 1 & sh_y^z \\ sh_z^x & sh_z^y & 1 \end{bmatrix} \quad (4.3.5)$$

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \quad (4.3.6)$$

For chosen scaling (sc_x, sc_y, sc_z) , shearing $(sh_x^y, sh_x^z, sh_y^x, sh_y^z, sh_z^x, sh_z^y)$, and rotation (ϕ, θ, ψ) parameters, we create a set of affine transforms, \mathcal{A} . Each parameter may take a range of values (for example, scaling parameter sc_x may be drawn from a set of values, $sc_x \in K_{sc}$) and so the cardinality of \mathcal{A} is the combinatorial product of the sizes of each parameter set: $|\mathcal{A}| = \prod_{i=1}^{12} |K_i|$.

While reflections are also included in the space of affine transforms, we omit reflections from the set of allowable shape transforms. Reflecting a formation between timesteps is undesirable because either robots must travel greater distances to their assigned positions and risk inter-robot collision, or reassignment must occur to switch robots to alternate destinations in the formation to avoid wasted motion and inter-robot collision.

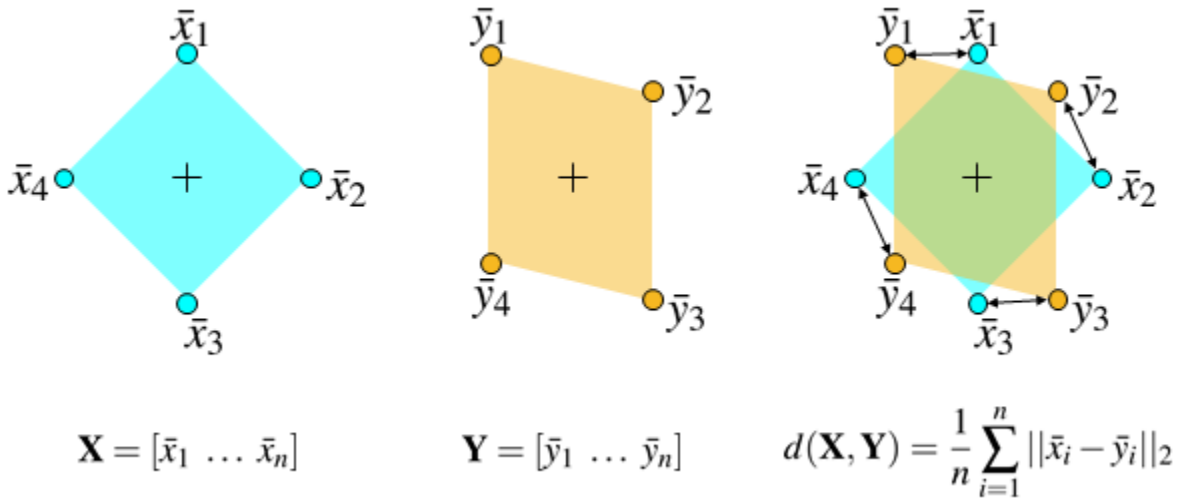


Figure 4.3.1: Average displacement across robots between two formations.

4.3.4 Cost Function

In order to determine if a formation evaluated during search is “close” to the formation specified by a desired behavior, we require an evaluation metric quantifying a notion of displacement between two formations. Given a colliding formation, we would like to minimize the distance each robot must travel to assume a non-colliding formation. Therefore, we define a cost metric based on the average distance between robot assignments between two formations.

A formation is a collection of robot positions, as described in 4.3.1: $\mathbf{S} = [\bar{s}_1 \dots \bar{s}_n]$, $\bar{s}_i = [x, y, z]^T$. Given two formations, $\mathbf{X} = [\bar{x}_1 \dots \bar{x}_n]$ and $\mathbf{Y} = [\bar{y}_1 \dots \bar{y}_n]$, we described the average displacement over all robots in the formation by:

$$d(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^n \|\bar{x}_i - \bar{y}_i\|_2, \quad (4.3.7)$$

where $\|\cdot\|_2$ is the l^2 , or Euclidean, norm. This is also illustrated in Fig. 4.3.1.

To define the cost of a candidate shape transform as part of a solution sequence, we consider the displacement of robots between the formation formed by the candidate shape transform and two other formations: the formation specified by the desired behavior at time t , and the candidate formation at time $t - 1$. The final cost of a formation at timestep t formed by a candidate shape transform is then the sum of terms:

$$\text{cost}(\mathbf{A}_t \mathbf{S}_0) = d(\mathbf{A}_t \mathbf{S}_0, \mathbf{R}_t \mathbf{S}_t) + d(\mathbf{A}_t \mathbf{S}_0, \mathbf{A}_{t-1} \mathbf{S}_0). \quad (4.3.8)$$

To evaluate a sequence of transforms, as required over multiple colliding timesteps, $\bar{\mathbf{A}} = [\mathbf{A}_1, \dots, \mathbf{A}_T]$, the total cost is the sum of the costs incurred over the sequence: all direct costs between the desired behavior and the candidate formations at equivalent timesteps, and the cost of transitioning between each formation in the proposed solution sequence:

$$\text{cost}(\bar{\mathbf{A}} \mathbf{S}_0) = \sum_{t=1}^T d(\mathbf{A}_t \mathbf{S}_0, \mathbf{R}_t \mathbf{S}_t) + \sum_{t=2}^T d(\mathbf{A}_t \mathbf{S}_0, \mathbf{A}_{t-1} \mathbf{S}_0). \quad (4.3.9)$$

4.3.5 Optimization Problem

We then state the problem of finding non-colliding behavior $\bar{\gamma}'_t$ as the problem of finding a sequence of transforms, $\bar{\mathbf{A}} = [\mathbf{A}_1, \dots, \mathbf{A}_T]$, which minimizes the deviation of $\bar{\gamma}_t$ from $\bar{\gamma}'_t$ for $t \in 1, T$ while

remaining collision free:

$$\mathbf{A}^* = \operatorname{argmin}_{\mathbf{A}^* \in \mathcal{A}} \operatorname{cost}(\bar{\gamma}_t(\mathbf{C}, \mathbf{R}, \mathbf{S}) - \bar{\gamma}'_t(\mathbf{C}, \mathbf{A}^*, \mathbf{S}_0)) \quad (4.3.10)$$

$$\text{s.t. } \mathcal{P}(\bar{\gamma}'_t(\mathbf{C}, \mathbf{A}^*, \mathbf{S}_0)) \cap \mathcal{O} = 0 \text{ for } t \in 1, T \quad (4.3.11)$$

4.3.6 Search graph formulation

As previously stated, we propose to find \mathbf{A}^* through search. We formulate the search graph as a directed tree structure, $G = \{V, E\}$, to represent ordered sequences of shape transforms where each vertex $v \in V$ represents a shape transform \mathbf{A} from the search set \mathcal{A} and directed edges $e \in E$ indicate the ordering of transforms. The branching factor of the tree is then $|\mathcal{A}|$.

4.3.7 MCTS formulation

A meaningful discretization of affine transforms results in a large search space with a high branching factor. We choose to employ Monte Carlo Tree Search (MCTS) [12] given this problem setup, as MCTS allows us to build a graph online, balancing exploration of new nodes with exploiting previous solutions, and performs well in similar search scenarios [75]. The four established components of MCTS are selection, evaluation, simulation, and back propagation. Our implementation takes the following form:

1. **Shape transform selection:** We select nodes, representing shape transforms, from the tree according to the UCT (Upper Confidence Bound for Trees) algorithm [12] to balance exploration and exploitation. That algorithm is restated below as:

$$UCT_j = \bar{X}_j + 2C_p \sqrt{\frac{2 \ln n}{n_j}}, \quad (4.3.12)$$

where \bar{X}_j is the average value of node j ; n_j is the number of simulations which have been played through node j ; and n is the number of simulations played through the parent of node j . The value C_p is a constant, $C_p = \frac{1}{\sqrt{2}}$ [12, 43]. Our rollout policy is random selection.

In our formulation, we set the value of a node, \bar{X}_j , as the ratio of the number of times that the node has been found to be the minimum cost, collision-free solution divided by the number of times the node has been sampled.

2. **Sequence evaluation:** Sequence evaluation is the computation of cost according to Eqn. (4.3.9).
3. **Sequence simulation:** In our problem formulation, "simulating play," equates to evaluate the performance of the sampled sequence of shape transforms in context, i.e., with respect to the environment. This equates to calculating $\bar{\gamma}'_t = \mathbf{C}_t + \bar{\mathbf{A}}\mathbf{S}_0$ over time horizon $t \in \{1, T\}$ and collision checking $\bar{\gamma}'_t$ with the environment.
4. **Back propagation:** Cost and sample count information are updated for all nodes in the sampled sequence.
5. **Final selection:** Sampling, evaluation, simulation, and back propagation are performed for a budgeted number of samples (equivalent to a fixed search time), and the lowest cost sequence of transforms is returned after the sample budget has been reached.

We note two additional items which we found greatly enhance search performance. First, because actions are transformations of a nominal fixed shape, we can pre-compute collision checks at each step along the path for each $\mathbf{A}\mathbf{S}_0$ where $\mathbf{A} \in \mathcal{A}$. This reduces collision checks to only a simple lookup during search, and, further, we remove the actions that are in collision from the set of available actions at each time step.

Second, MCTS using UCTs may select the same actions multiple times in sequence, particularly when the search horizon is relatively short. We found in practice that the technique of updating sample count across children to reflect multiple plays through a sequence with a known

score reduced repeated visitation of fully explored sequences and greatly improved performance.

4.4 Experiments and Analysis

To evaluate the performance of the proposed approach, we perform experiments in simulation. The following experiments in Sects. 4.4.1 and 4.4.2 show that a sample-limited MCTS implementation is able to improve performance with increased experience and that the performance of a search tree holds across environments, meaning that experience accumulated in one environment may be reused in a new environment.

Experimental setup

To evaluate search performance, we perform searches in simulation through randomly generated environments. Desired behaviors through the environment are generated using random destination selection. A formation of robots is initialized at a random starting point with a randomly chosen destination. The desired behavior is the motion formation along the shortest path between the start and end waypoints, and the final waypoint becomes the starting waypoint for the next behavior. An example of a desired behavior in a test environment is shown in Fig. 4.4.1.

As all waypoints are randomly chosen, a behavior will encounter a random number of collisions with the environment. Therefore, we look at the overall cost of a solution for a behavior as an average cost incurred over all collisions. An example of this was shown in Fig. 4.2.1a; the individual cost of every non-colliding transform is shown plotted in the figure, and the average cost of the solution sequence is reported in the figure legend. Mathematically, the average cost is

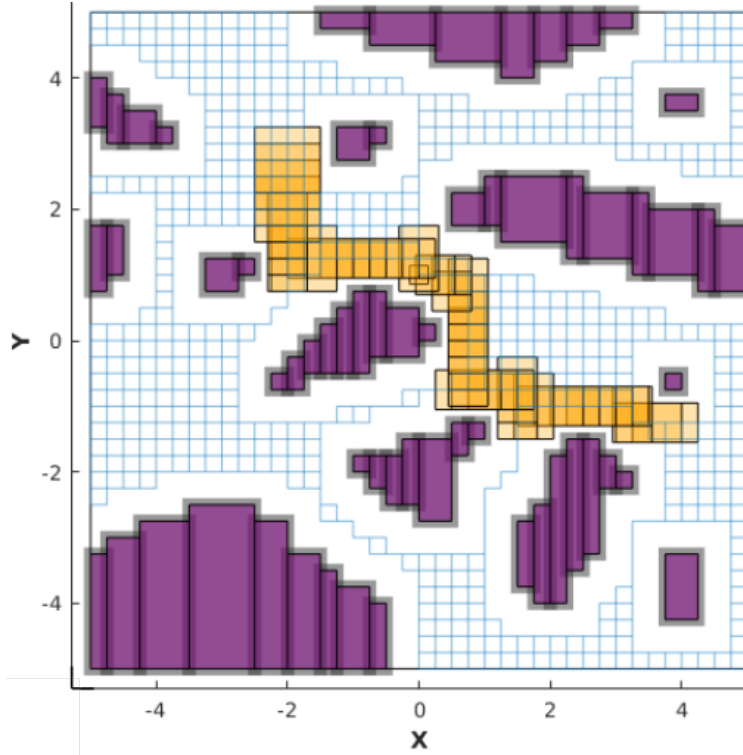


Figure 4.4.1: This figure shows a representative experimental scenario, showing a randomly generated environment and obstacles (purple). A solution sequence of formations (orange) is shown for a desired square formation moving between randomly chosen start and goal locations.

found by dividing the cost of a sequence (given in Eqn. (4.3.9)) by the number of elements in the sequence: $\text{mean}(\text{cost}(\bar{\mathbf{A}}\mathbf{S}_0)) = \text{cost}(\bar{\mathbf{A}}\mathbf{S}_0)/T$.

Error metrics

While we are interested in understanding how the proposed search formulation performs, i.e. the cost of a solution, it is more informative to understand how the cost of a solution found through search compares to the lowest cost we could expect to achieve given the chosen search set, e.g., the cost of the optimal solution composed of shape transforms from the search set, \mathcal{A} . While MCTS is complete in the limit, meaning it is guaranteed to find the optimal solution or conclude that no solution exists given sufficient samples to expand a full search tree, MCTS may not find

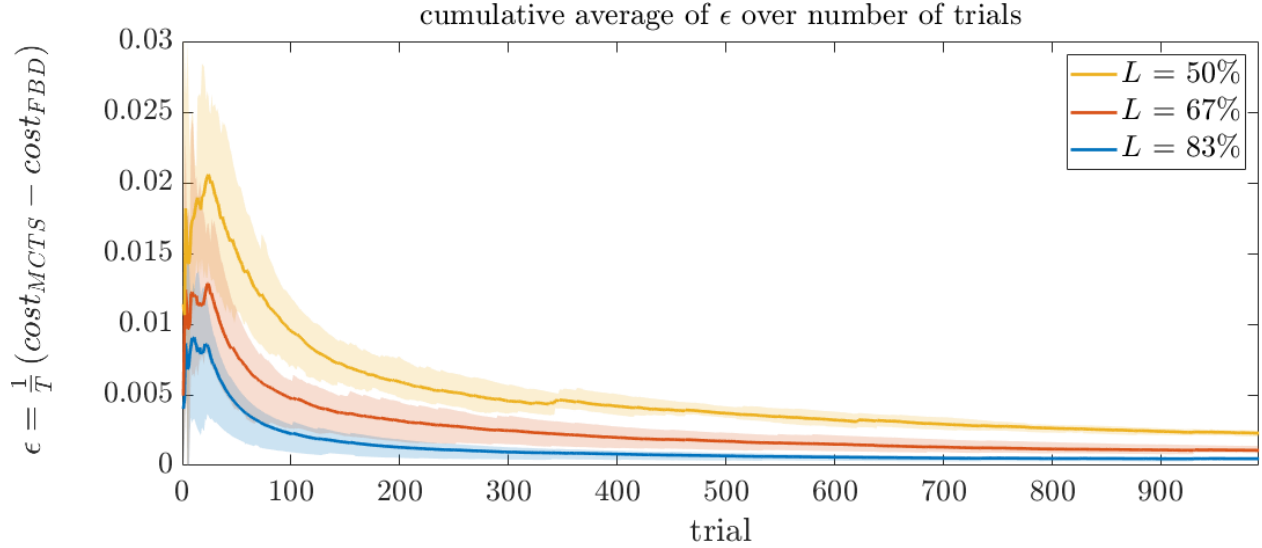


Figure 4.4.2: This plot shows the performance of search given different sample limits. For the same set of randomly generated paths, search is run with three different sample limits.

the optimal solution when the search time or number of search iterations is constrained.

The optimal solution for the discrete problem formulation can be found by performing a complete search over the entire tree of search options. Performing a full breadth and depth search (FBD search) is infeasible for online use, and is provided solely for solution quality comparison. The performance difference between a solution found through MCTS and solution determine through FBD search is therefore given as,

$$\epsilon = \frac{1}{T} (\text{cost}(\bar{\mathbf{A}}\mathbf{S}_0))_{\text{MCTS}} - \text{cost}(\bar{\mathbf{A}}\mathbf{S}_0))_{\text{FBD}}. \quad (4.4.1)$$

We include the $\frac{1}{T}$ term, forming ϵ as an arithmetic average, in order to compare average error across trials experiencing different numbers of collisions. Eqn. 4.4.1 is then used to report performance in all experiments discussed in Sects. 4.4.1 and 4.4.2, Figs. 4.4.2 and 4.5.1.

4.4.1 Method performance characteristics

To show that the proposed approach is able to find solutions and improve performance with increased experience, we show performance characteristics of the methodology for 5 different sets of 1000 random behaviors, through a randomly generated environment, in Fig. 4.4.2. The three curves shown in Fig. 4.4.2 show the performance of the methodology for three different sample budgets. The individual lines represent the mean performance over 5 trials, each of 1000 randomly generated behaviors; the shaded area shows the max and minimum range of values. All three curves decrease with the number of trials experienced, showing that the method, regardless of sample budget, improves solution cost with time. Looking across the three examples, it is clear that sample budget contributes to the learning rate; greater numbers of samples during search allow the method to explore more paths through the search tree, evaluating greater numbers of solutions. The sample budget for each trial is shown (legend of Fig. 4.4.2) as a percentage, $N_{sample} = L(B^D)$, where B^D are the possible number of leaf nodes. The value of total possible number of leaf nodes is chosen as a reference value to give a sense of scale to a sample budget.

4.4.2 Transfer across environments

Fig. 4.5.1 shows how the performance of a search tree generalizes across environments. A search tree, trained in one environment, is transferred to a new randomly generated environment. Both environments share the same voxelization parameters, meaning that voxel cell size is the same in both environments. The same search set and sample budget is used across all searches.

The yellow plot in Fig. 4.5.1 shows the performance of a search tree in Environment 1 over

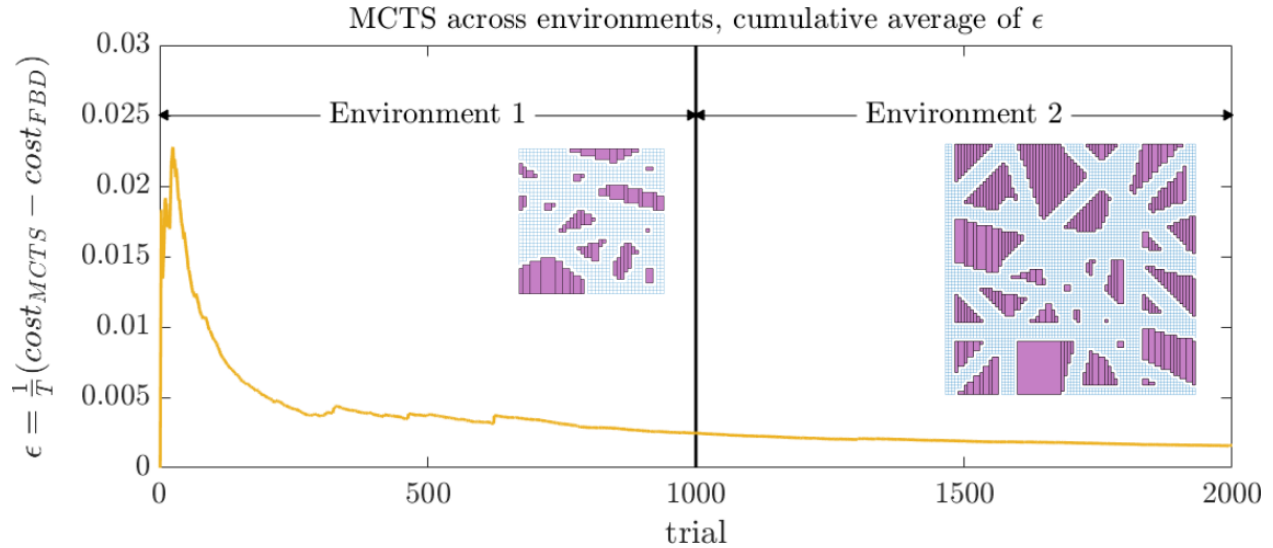


Figure 4.5.1: The yellow curve shows the performance of a search tree trained on “Environment 1” and transferred to “Environment 2.” Environments 1 and 2 are shown in overlaid on the plot, for reference. Note: Environment 2 is twice as large as Environment 1, to show an example of differences between environments and evaluate a trained search over longer paths with greater numbers of obstacles.

1000 trials, where the search tree is initialized with Trial 1. The search tree is then used to find solutions for 1000 randomly generated behaviors in a new scenario, Environment 2. As shown, the performance of the search tree remains constant between the two environments.

4.5 Discussion and Conclusion

Experiments performed in Sects. 4.4.1 and 4.4.2 illustrate the two necessary capabilities to use the proposed approach online: 1) We have verified that formation sequences (i.e., shape transform sequences) can be valued based on experience and that this experience improves performance, even allowing a limited sample budget to achieve equivalent performance to higher sample (and therefore more time consuming) search; 2) For a given environment discretization, a trained search tree holds performance across different environments.

These two attributes show that a search tree can be trained offline to achieve a desired performance threshold and that the method will perform comparably online across changing operating scenarios. Given application requirements, the results show that an operator or system designer can evaluate offline performance in representative environments for a given search set resolution and choose a search budget to achieve a desired level of performance.

Drawbacks to this method are that this approach requires offline training and that separate search trees must be trained for different nominal formations, S_0 , different search sets, \mathcal{A} , and for different resolutions of an environment voxel grid. However, this is comparable to approaches which compute motion primitive libraries for individual agents; we have, in effect, proposed a motion primitive library for multi-robot formations which leverages a trained selector to enable fast online use.

While we have only considered static environments, the finding that a trained search tree depends on environment discretization rather than the environment itself suggests that we may be able to represent dynamic obstacles as static obstacles over a forward time horizon and use the proposed method directly for dynamic obstacle avoidance. Likewise, the described approach is formulated to account for a time varying desired behavior reference, such as those described in [15, 16]. The ability of the methodology to provide high-resolution solutions is promising for being able to represent time varying behaviors with good fidelity, and we are interested in understanding how the proposed search methodology extends given the increased state space introduced by a non-static reference.

Chapter 5

Multi-robot planning via action search

The preceding chapter described human-instructed, online multi-robot planning in the specific context of a performer-directed theatric application. Advantages to the described methodology are that complex and specific instructions detailing desired inter-robot coordination can be issued by a human user within the application time constraints. However, the need for the enumeration of specific classes of desired behavior attributes, the time required to specify coordination requirements, and the restriction of operating in an open environment restrict the usability of the approach when looking at operating in different environments or for alternate applications.

In this chapter, we show that the previously described approach can be thought of as a specific instantiation of a general formulation of multi-robot behaviors. We present notation for describing multi-robot actions, and a general framework that shows that sequences of these actions can be composed online to form feasible and safe multi-robot behaviors. We then show that this approach is able to replicate performance of the specific methodology.

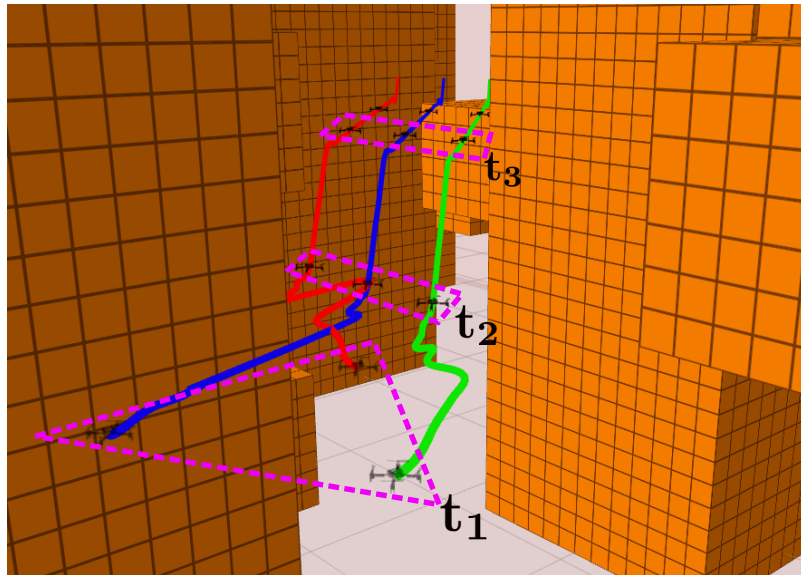
In this chapter, we begin by giving further context to the problem and presenting a brief

overview of our approach (Section 5.1). Notation and problem formulation are introduced in Sect. 5.2, and Sect. 5.3 introduces how data, as demonstrations of desired user actions, may be given as time-series of positions and incorporated into the desired notation framework. Our formulation for composing actions is a search framework, described in Sect. 5.4. To quantify how a solution returned from search represents the user-input (demonstration data), Sect. 5.5 describes several evaluation metrics. We then evaluate the performance of the methodology in simulation (Sect. 5.6) by showing performance in various cluttered environments and for different sized groups of robots.

5.1 Approach Overview

In considering the problem of planning coordinated motions online for a group of robots, there are challenges associated with not just planning kinodynamically feasible and collision free motions despite environment complexity, but also with specifying coordination policies for groups of robots. When the coordination goal is well understood or clearly expressed by quantifiable metrics, optimization approaches or rule-based policies are commonly used to plan team motions. For example, the common task of maintaining agents in formation is often formulated in terms of minimizing deviation to a desired formation, and can be solved via a constrained optimization employing sequential convex programming [5], using a vector field policy to avoid obstacles but maintain formation [91], or time-based polynomial trajectory planning over a receding horizon in order to maintain distance and bearing specifications between agents [78]. The coordination policy of these approaches (in this example, “minimize deviation from a desired reference”) is specified by human understanding of the problem domain.

Figure 5.1.1: Example of coordinated motions between robots in a formation being performed in a complex environment. The robot motions were selected during online search based on user-provided demonstration data. Robots are shown at sampled times, t_1 , t_2 , and t_3 , along a set of trajectories generated through keyframes returned via the proposed search methodology. The robots transition from a triangle formation to a line formation in order to avoid collision with an obstacle (represented by the orange voxels).



However, some applications can require coordination policies that are difficult to express via optimizable metrics, or which do not have a clear best policy. In these cases, demonstrated solutions—if available—may be used to inform planning policies. In the example of team sports such as soccer where it may be unclear how players should perform defense, [48] uses video demonstration of human players to learn agent roles. In [51], no decentralized controller is available for the posed particle assignment problem, and so a centralized planning formulation is used as a demonstration to train decentralized policies for a multi-robot team.

We seek to plan coordinated motions reflecting user preferences for a team of robots in complex and cluttered environments without having to explicitly express user objectives through a rule-based methodology. The methodology described in the preceding chapter (Chapter 3) of this thesis allows a user to specify time-varying formation objectives online via parameterized input and create dynamically feasible and collision free multi-robot plans in the form of time-based

polynomial trajectories in non-cluttered environments. However, directly finding dynamically feasible and non-colliding trajectories for the desired user input in cluttered environments is difficult to solve under online time constraints.

We propose to use plans reflecting user specified multi-robot coordination preferences generated in an open environment via [15, 16] to inform online plan generation in cluttered and complex environments. The contribution of this approach is that it allows a system to reproduce user-preferred coordination strategies in non-demonstrated scenarios (the transfer of actions from open to cluttered environments), as well as perform coordinated multi-robot motions without requiring the direct specification of a rule-based objective function or direct online user-input.

A conceptual overview of the proposed approach is visually depicted in Fig. 5.1.2, and portions of this work are additionally covered in our publication [17]. Our approach is divided into Offline and Online portions. We formulate multi-robot planning as a Markov Decision Process, where robots transition between states via actions, and introduce notation to describe a discrete representation of multi-robot states and actions (Sect. 5.2). Our approach looks at multi-robot motion data given as time series of robot positions. Sect. 5.3 describes the representation of a time series as multi-robot state-action tuples, and the creation of a search set composed of actions observed from datasets.

Sect. 5.4 describes the online navigation of a group of robots through a cluttered environment. Finding a sequence of actions that allows the robot group to navigate the desired path—input via joystick from a user—can be formulated as search over a tree of possible actions. To minimize edge expansions for fast search online, we evaluate a best-first search approach. We again leverage

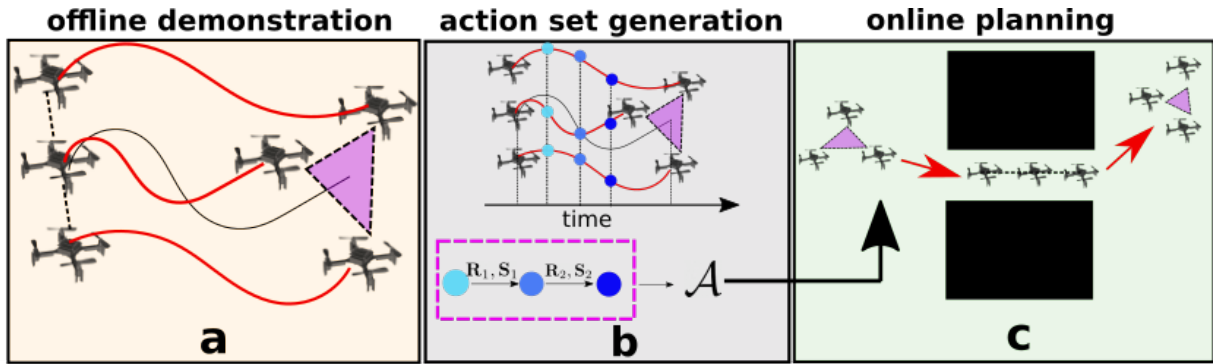


Figure 5.1.2: This figure illustrates an overview of the approach. Block (a) shows an offline demonstration of a group of robots changing formation from a line to a polygon. Block (b) illustrates the demonstration, sampled at uniform time discretizations, represented as actions performed by the formation, described as rotations and shape transformations. These actions are stored in an action set, \mathcal{A} . Block (c) shows a group of robots in a cluttered environment, executing actions from action set \mathcal{A} to transition from polygon to line to navigate between obstacles.

the provided datasets of multi-robot actions to guide search, and propose the use of a simple, probability-based heuristic for action selection.

Sect. 5.6 evaluates the proposed approach with a series of experiments showing method performance in terms of computation time and solution quality when the approach is employed in varying simulated environments. We evaluate scaling with action resolution, number of robots, and environment and group representation. We also compare to existing approaches in a representative example environment.

5.2 Notation and problem formulation

This section introduces notation and problem formulation. In the following descriptions, indexing elements are given as superscripts, while time elements are written as subscripts. For example, the state x of robot i at time t is written as x_t^i .

5.2.1 Robot, formation, and obstacle notation

The state of an individual robot, s , with respect to a local, formation reference frame \mathbb{S} is given by its position, $s = [x, y, z]^T \in \mathbf{R}^3$. The state of a group of n robots is a vector containing the states of the member robots, $\mathbf{S} = [s^1, \dots, s^n] \in \mathbf{R}^{3 \times n}$. A valid formation, or shape \mathbf{S} , is one in which no robots are in collision with each other: i.e., a minimum separation distance, dr , is required between all robots in a formation: $\|s^i - s^j\|_2 \geq dr \forall (s^i, s^j) \in \mathbf{S}$, and $\|\cdot\|_2$ describes the Euclidean distance or l^2 -norm.

Similar to state in the group reference frame, we define the state of a robot in world frame \mathbb{W} as $x = [x, y, z]^T, x \in \mathbf{R}^3$, and the state of a group of n robots as $\mathbf{X} = [x^1, \dots, x^n], \mathbf{X} \in \mathbf{R}^{3 \times n}$. The formation reference frame, \mathbb{S} , is related to \mathbb{W} by a positional offset, $C \in \mathbf{R}^3$, and a rotation, $R \in \text{SO}(3)$, so that state s of a robot in frame \mathbb{S} is expressed in frame \mathbb{W} as $x = C + Rs$, and likewise for a formation of robots, $\mathbf{X} = C + R\mathbf{S}$.

We use a voxel-based environment representation, letting $\mathcal{O} \subset \mathbb{R}^3$ be the set of static obstacles (occupied voxel cells). However, the methodology described in this work holds for any chosen environment or obstacle representation, for example, defining obstacles \mathcal{O} as a point set of observed surface data or collection of convex polygons, etc. We define $\mathcal{P}(\mathbf{X}_t)$ as the set of voxels occupied by the convex polytope defined by robot poses \mathbf{X} at time t . A formation of robots is collision free if the set of voxels covered by the formation does not intersect the obstacle set, i.e., the intersection of the two sets is the empty set: $\mathcal{P}(\mathbf{X}) \cap \mathcal{O} = \{\}$.

5.2.2 MDP over states and actions

We formulate planning for a group of robots as a Markov decision process (MDP), where our state at time t is the tuple $\mathbf{s}_t = \{C_t, R_t, \mathbf{S}_t\}$, an action at time t is the tuple $\mathbf{a}_t = \{c_t, \Omega_t, A_t\}$, and the transition function $\mathcal{T}(\cdot)$ gives $\mathcal{T}(\mathbf{s}_{t+1} | \mathbf{a}_t, \mathbf{s}_t)$ by:

$$C_{t+1} = C_t + c_t \quad (5.2.1)$$

$$R_{t+1} = \Omega_t R_t \quad (5.2.2)$$

$$\mathbf{S}_{t+1} = A_t \mathbf{S}_t. \quad (5.2.3)$$

In the above set of equations, $c_t \in \mathbf{R}^3$ describes a translation in \mathbb{W} in x , y , and z ; $A_t \in \mathbf{R}^{3 \times 3}$ describes a linear transformation of the n robot formation, $\mathbf{S}_t \in \mathbf{R}^{3 \times n}$; and $\Omega_t \in \text{SO}(3)$ describes an incremental rotation that takes R_t to R_{t+1} . We let $\mathcal{T}(\cdot)$ be deterministic, transitioning \mathbf{s}_t to \mathbf{s}_{t+1} given \mathbf{a}_t with probability 1.

The problem definition is as follows: given the current system state \mathbf{s}_t at time t , we seek to complete a path specification given only as components c and Ω of an action with the most appropriate shape space transform A to form the complete action tuple: $\mathbf{a}_t = \{c_t, \Omega_t, A_t\}$ to take the system to the desired (and feasible) state at the next time step, \mathbf{s}_{t+1} . This may be stated as:

$$A_t = \underset{A_t \in \mathcal{A}}{\operatorname{argmax}} \operatorname{Reward}(A_t) \quad (5.2.4)$$

$$\text{s.t. } \mathcal{P}(\mathbf{X}_{t+1}) \cap \mathcal{O} = \{\} \quad (5.2.5)$$

$$\|s^i - s^j\|_2 \geq dr \quad \forall (s^i, s^j) \in \mathbf{S}_{t+1}, s^i \neq s^j. \quad (5.2.6)$$

Reward(A_t) is a reward function in terms of A_t , the linear transform describing the transition of \mathbf{S}_t to \mathbf{S}_{t+1} , drawn from a set \mathcal{A} of demonstrated transforms. We discuss the specifics concerning the generation of set \mathcal{A} and details of reward calculation Reward(A_t) to Sects. 5.3 and 5.4, respectively.

5.3 Demonstration data as $\{\mathbf{s}, \mathbf{a}\}$ tuples

To select a shape transform A_t as in Eqn. (5.2.4) in line with user expectations, we require a demonstration of user-directed robot motions represented as a series of state-action tuples. Here, we generate demonstrations using the methodology of [15, 16], which takes detailed user input to produce dynamically feasible, time-based polynomial trajectories for all robots. We therefore sample the demonstration at a chosen, uniform time discretization to create a time-series of robot states. The dynamic feasibility of the demonstration trajectories means that (1) no states contain robot-robot collisions, and (2) sampling at a fixed time resolution provides actions within a bounded set, as the demonstration trajectories generated by [15, 16] respect the provided actuator limits of the physical robot platform.

We begin with a demonstration of the form $\mathbf{X}_{1:T}$ in \mathbb{W} for all robots in a group, where \mathbf{X}_t has been sampled at $t = \{1, \dots, T\}$, and $t_{i+1} = t_i + dt$. Given two sequential poses, \mathbf{X}_t and \mathbf{X}_{t+1} , and the knowledge of either \mathbf{S}_t or R_t , we can decompose \mathbf{X}_t and \mathbf{X}_{t+1} to MDP states \mathbf{s}_t and \mathbf{s}_{t+1} and the action \mathbf{a}_t which transitions \mathbf{s}_t to \mathbf{s}_{t+1} . This is a Procrustes problem [34], where we seek a transformation that maps robots from \mathbf{X}_t to \mathbf{X}_{t+1} , such that transformation Y minimizes $\|Y\mathbf{X}_t - \mathbf{X}_{t+1}\|$.

5.3.1 Position and translation components of states and actions

We equate the position component of state, C , to the geometric mean of robot positions: $\bar{\mathbf{X}}$. Given position components $C_t = \bar{\mathbf{X}}_t$ and $C_{t+1} = \bar{\mathbf{X}}_{t+1}$ at neighboring time steps, translation term c_t is found by subtracting the geometric mean of the robot group at time t from $t + 1$, a rearrangement of Eqn. (5.2.2): $c_t = C_{t+1} - C_t$.

5.3.2 Rotation component, Ω_t , of action \mathbf{a}_t

Knowledge of the position component lets us group the remaining state components, R_t and \mathbf{S}_t , together as P_t :

$$P_t = R_t \mathbf{S}_t \quad (5.3.1)$$

$$= X_t - C_t \quad (5.3.2)$$

$$P_{t+1} = R_{t+1} \mathbf{S}_{t+1} \quad (5.3.3)$$

$$= X_{t+1} - C_{t+1} \quad (5.3.4)$$

Given P_t and P_{t+1} , the rotation matrix Ω_t is the solution to the orthogonal Procrustes problem [34]: $\Omega = \operatorname{argmin}_{\Omega}$

$\| \Omega P_t - P_{t+1} \|_F$ subject to $\Omega^T \Omega = \Omega \Omega^T = I$ with $\det(\Omega) = 1$, where $\| \cdot \|_F$ denotes the Frobenius norm.

This problem is equivalent to finding the nearest special orthogonal matrix¹ to a given matrix

¹An orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors, i.e., $R^T R = R R^T = I$; a special orthogonal matrix is an orthogonal matrix whose determinant equals 1, $\det(R) = 1$.

$M = P_{t+1}P_t^T$. To find Ω_t , singular value decomposition is used to write:

$$P_{t+1}P_t^T = U\Sigma V^T \quad (5.3.5)$$

$$\Omega_t = U\Sigma'V^T \quad (5.3.6)$$

where Σ' is a modified Σ with the smallest singular value replaced by $\text{sign}(\det(UV^T))$, i.e. +1 or -1, and the other singular values replaced by 1, so that the determinant of Ω is guaranteed to be positive [29].

5.3.3 Discussion of R_0 and S_0

Poses P_t and P_{t+1} are known from Eqns. (5.3.1) - (5.3.4), and we would like to decompose this information into four state components, R_t , S_t , R_{t+1} , S_{t+1} . We have so far expressed only one constraint: that R_t evolves to R_{t+1} via pure rotation. This allows us to express R_{t+1} fully in terms of R_t via Ω_t (Eqn. (5.3.6)) exactly as in Eqn. (5.2.3). We therefore have two knowns, P_t and P_{t+1} , and three unknowns, R_t , S_t , and S_{t+1} , and so require one additional piece of information which may be provided by the knowledge of either the starting shape, S_0 , or the starting orientation, R_0 . The demonstration may be assumed to begin with R_0 equal to identity, or alternatively, S_0 may be defined via knowledge of the user's desired shape or by matching the observed positions of the robots at $t = 0$ to a library of desired formations.

5.3.4 Rotation and shape components of state

Without loss of generality, we use poses $\mathbf{P}_{t=0}$ and $\mathbf{P}_{t=1}$ to clarify the explanation of determining state rotation and shape components.

We first examine the case where R_0 has been specified. Given R_0 , Eqn. (5.3.1) may be directly solved for $\mathbf{S}_{t=0}$:

$$\mathbf{S}_{t=0} = R_0^T P_0, \quad (5.3.7)$$

and the evolution of $R_{t=0}$ by $\Omega_{t=0}$ (Eqn. (5.3.6)) gives $R_{t=1}$, yielding $\mathbf{S}_{t=1}$:

$$\mathbf{S}_{t=1} = R_1^T P_1. \quad (5.3.8)$$

If \mathbf{S}_0 has been specified, R_0 may be found similarly as to Eqn. (5.3.6), as:

$$P_0 \mathbf{S}_0^T = U \Sigma V^T \quad (5.3.9)$$

$$R_0 = U \Sigma' V^T, \quad (5.3.10)$$

and the solution of $\mathbf{S}_{t=0}$ and $\mathbf{S}_{t=1}$ proceeds as in Eqns. (5.3.7) and (5.3.8).

While we have used poses at $t = 0$ and $t = 1$ for explanation, it should be clear that knowledge of R or S at any time t enables the calculation of all components of state at $t + 1$. States at all times may therefore be calculated sequentially from knowledge state $\mathbf{s}_{t=0}$.

5.3.5 Shape transform A_t

The final component of action \mathbf{a}_t undefined is transform A_t . This is, again, a Procrustes problem, although here unrestricted to a rotation matrix and so may be solved through the least squares minimization given by [34] (additionally as noted in the example implementation of [22]):

$$A_t = \mathbf{S}_{t+1} \mathbf{S}_t^T (\mathbf{S}_t \mathbf{S}_t^T)^{-1}. \quad (5.3.11)$$

Due to the described extraction of rotation matrix Ω_t from P_t, P_{t+1} , transform A_t may be interpreted as scaling and shearing elements, where scale may be given along the x, y , and z dimensions and with the off-diagonal components describing shearing as:

$$\begin{bmatrix} s_{xx} & 0 & 0 \\ 0 & s_{yy} & 0 \\ 0 & 0 & s_{zz} \end{bmatrix}, \begin{bmatrix} 1 & s_{xy} & s_{xz} \\ s_{yx} & 1 & s_{yz} \\ s_{zx} & s_{zy} & 1 \end{bmatrix}. \quad (5.3.12)$$

We make no assumptions about the uniformity of scaling, meaning that $s_{xx} \neq s_{yy} \neq s_{zz}$, nor symmetry of shearing, i.e., $s_{xy} \neq s_{yx}$, etc, reiterating that $A \in \mathbf{R}^{3 \times 3}$ or equivalently $A \in \mathbf{R}^9$ for the values $\{s_{xx}, s_{yy}, s_{zz}, s_{xy} s_{yx}, s_{xz} s_{zx}, s_{yz} s_{zy}\}$.

Transforms A_t observed from demonstration $\mathbf{X}_{1:T}$ may be discretized (for example, casting all values to a desired resolution, dA) so that actions can be recognized as discrete and unique. A set of possible actions, \mathcal{A} used in Eqn. (5.2.4), may then be formed of all unique actions from the demonstration.

Algorithm 1: Best First Search [69] formulated for our application.

```

1 returns TreePath or failure
2  $root \leftarrow$  node with formation start state and zero reward
3  $frontier \leftarrow root$  // queue ordered by cumulative reward
4  $explored \leftarrow \{\}$  // empty list
5 while  $i < limit$  do
6   if Empty(frontier) then return failure
7    $node \leftarrow Pop(frontier)$  // highest cumulative reward
8    $explored \leftarrow node$ 
9   if CollisionFree(node) then
10    if  $node.depth == T$  then return TreePath(root, node)
11     $Children \leftarrow Expand(node)$ 
12     $CumulativeReward(Children)$ 
13     $frontier \leftarrow Insert(Children \setminus (explored \cup frontier))$ 
14  end
15 end

```

5.4 Online action search

This section describes the formulation of action selection as a tree search among possible actions, and the introduction of a simple reward measure for estimating user-preferred actions.

5.4.1 MDP as tree search

The process of selecting and evaluating the possible actions from a given state may be formulated as search over a directed graph, specifically a tree, where nodes represent states and edges represent actions. A node stores state $\mathbf{s} = \{C, R, \mathbf{S}\}$ and collision information with respect to obstacle set \mathcal{O} . Each edge represents a single action, $A \in \mathcal{A}$.

The root node is initialized at the start position in the world, $C_{t=0}$, with the starting rotation, $R_{t=0}$, and formation state, $S_{t=0}$. At each node, a maximum of $|\mathcal{A}|$ possible decisions can be made (i.e., a node has a maximum of $|\mathcal{A}|$ possible children), where $|\mathcal{A}|$ is the size of action set \mathcal{A} .

Child node state is found by applying the transition function (Eqns (5.2.1)-(5.2.3)). Therefore

children share action components c_t and Ω_t , resulting in shared state components C_{t+1} , R_{t+1} , but have different shape transforms A_t and so different shapes \mathbf{S}_{t+1} . Every child node is checked for collisions with the environment as well as for collisions between robots, and nodes in collision are marked as inadmissible.

In order to find actions quickly to meet online planning time requirements, we propose a Best First Search (BFS) approach [69] to determine a sequence of shape transforms, $A_{1:T}$, for a user input $\{c_{1:T}, \Omega_{1:T}\}$. BFS for our application is summarized in Alg. 1. Search may be terminated under three conditions: (1) when the end of the horizon is reached, $t = T$; (2) the frontier list (Algorithm 1) is empty, meaning all non-colliding nodes have been explored; (3) we have reached a maximum number of node expansions, capped so as to limit online planning time.

5.4.2 Selection policy

We suggest that the reward of a node representing transform A_t be based on the probability of seeing A_t as part of a string of actions from the demonstration. We express this by defining a window around time step t using two parameters, m and h , where m defines the number of actions prior to, and h defines the number of actions following, A_t .

$$\text{Reward}(A_t) = \frac{1}{1 + \epsilon} p(A_{t:t+h} | A_{t-m:t-1}) + t. \quad (5.4.1)$$

Given many nodes with similar or equal reward, we prefer to expand nodes that move closer to the end of the path. As search depth is correlated with time step t , we therefore add t to the reward based on $p(A_t)$.

The discount factor of $\frac{1}{1+\epsilon}$, where ϵ is any positive number, is a constant term and so does not change the relative reward value between child nodes. This term ensures that the probability of any action $p(A_t)$ is scaled to be less than one: as $p(A_t)$ is a probability, it will always lie within the bounds of $0 \leq p(A_t) \leq 1$, meaning that $0 \leq \frac{1}{1+\epsilon}p(A_t) < 1$. This reward structure simply means that this instantiation of Best First Search is effectively equivalent to Depth First Search, but with a ranked order among nodes at a given depth. This ranking allows us to choose the best node to expand, rather than randomly choosing an edge to expand or collision checking all children to find only non-colliding choices (Alg. 1).

5.5 Evaluation Metrics

This section describes three measurable attributes that allow us to evaluate a search solution, a string of shape transforms over a time horizon T , $A_{1:T}$, with respect to a demonstration. The solution is a string (a finite sequence of symbols chosen from an alphabet, a finite set of symbols), where each symbol is a unique transform, $A^i \in \mathcal{A}$, and the size of the alphabet is the chirality of set \mathcal{A} , $|\mathcal{A}|$.

Longest Common Substring (LCS) One measure for determining the extent to which a solution exactly matches (a portion of) the demonstration can be expressed by the Longest Common Substring (LCS). Given two strings F and G , of lengths L_F and L_G respectively, the LCS is the longest string which is a substring of both F and G .

We additionally look at two possible metrics for scoring the “distance” of a solution from the

demonstration.

Minimum Demonstrated Hamming Distance (MDHD) Edit distance describes the number of operations required to transform one string to another. The Hamming distance between two strings of equal length (we denote this as $d_H(\cdot, \cdot)$) is the number of positions at which the corresponding symbols differ [86], and in our application, gives a measure of how well the solution could have done (ignoring the feasibility requirements which constrained solution choice) to exactly replicate a portion of the demonstration.

The Hamming distance can only be computed between strings of equal length. For a demonstration string F of length L_F , and a solution string G with length L_G , there will be $(L_F - L_G + 1)$ possible L_G -length substrings from the demonstration. We denote the Minimum Demonstrated Hamming Distance² (MDHD) between the query string, G , and the set of substrings, f^i with length $L_f = L_G$, drawn from the demonstration string, F , as:

$$f^i \subseteq F, \quad L_f = L_G \quad (5.5.1)$$

$$f^* = \operatorname{argmin}_{f^i \subseteq F} d_H(f^i, G) \quad (5.5.2)$$

$$\text{MDHD} = d_H(f^*, G). \quad (5.5.3)$$

Minimum Demonstrated JSD (MDJSD) While it ignores action ordering, we can also view a demonstration and a solution as two probability distributions over actions. A common metric for

²Our defined “Minimum Demonstrated Hamming Distance,” identifying the minimum distance of a query string to a string set, is unrelated to “minimum Hamming distance” as defined in the literature, the smallest Hamming distance between all possible pairs of strings in a set [86].

measuring the similarity between two probability distributions is the Jensen–Shannon divergence (JSD), also known as the information radius or total divergence to the average [56]. The JSD is a symmetrized and smoothed version of the Kullback–Leibler divergence. We let F be the string of demonstration data and G the solution, and we use f_i, g_i , to denote the observed frequencies of A^i from the demonstration and the solution strings. The Kullback–Leibler divergence between strings F and G is then $\text{KLD}(F||G) = \sum_{i=1}^{|\mathcal{A}|} f_i \log_2\left(\frac{f_i}{g_i}\right)$, and the JSD is given as:

$$F = \{f_1, \dots, f_M\}, G = \{g_1, \dots, g_M\} \quad (5.5.4)$$

$$M = \frac{1}{2}(F + G) \quad (5.5.5)$$

$$\text{JSD}(F||G) = \frac{1}{2}\text{KLD}(F||G) + \frac{1}{2}\text{KLD}(F||G), \quad (5.5.6)$$

Comparing a solution to an entire demonstration is uninformative when $L_F \gg L_G$; we do not expect the solution to well represent an entire demonstration, but we would hope that the solution well represents some portion of the demonstration. Therefore, as in our calculation of the MDHD (Eqns. (5.5.2)–(5.5.3)), we calculate JSD with respect to the set of all L_G -length substrings of the demonstration to find a Minimum Demonstrated Jensen-Shannon divergence (MDJSD):

$$f^i \subseteq F, L_f = L_G \quad (5.5.7)$$

$$f^* = \underset{f^i \subseteq F}{\operatorname{argmin}} \text{JSD}(f^i||G) \quad (5.5.8)$$

$$\text{MDJS} = \text{JSD}(f^*||G). \quad (5.5.9)$$

5.6 Evaluation

In this section we evaluate the proposed search approach using the metrics proposed in Sect. 5.5 over several environments of varying complexity, and additionally examine the dynamic feasibility of trajectories interpolated through the discrete states returned from search.

Our methodology returns plans in the form of a non-colliding sequence of goal states (positions) for the robot group. To use the proposed approach with real robots, we require dynamically feasible plans trackable by a hardware platform. Some approaches use independent robot controllers to directly navigate to the goal destinations output from formation planning [5]; although this approach would also work for our problem formulation, we prefer to evaluate the dynamic feasibility of our plans by using the states returned from search as keyframes in a polynomial trajectory formulation. This allows us to quickly evaluate the dynamic feasibility and safety of the entire plan quickly online.

In the following experiments, we use the method of [66] to fit time-based polynomial trajectories to the discrete states returned from search. We evaluate plans for use with a quadrotor system requiring trajectories that are smooth and continuous up to the acceleration limits of the platform, and results are shown in Fig. 5.6.3. For each trial, robots are randomly placed in the environment and asked to follow a randomly-generated, set-length path.

Table 5.6.1 shows the solution metrics evaluating the discrete solution returned from search (Sect. 5.5). We evaluated several values of m and h to describe the length of the windowed history used in Eqn. 5.4.1, and found that using a small window of past actions performed best in terms of solution quality and computation time; we therefore show results for trials conducted with search

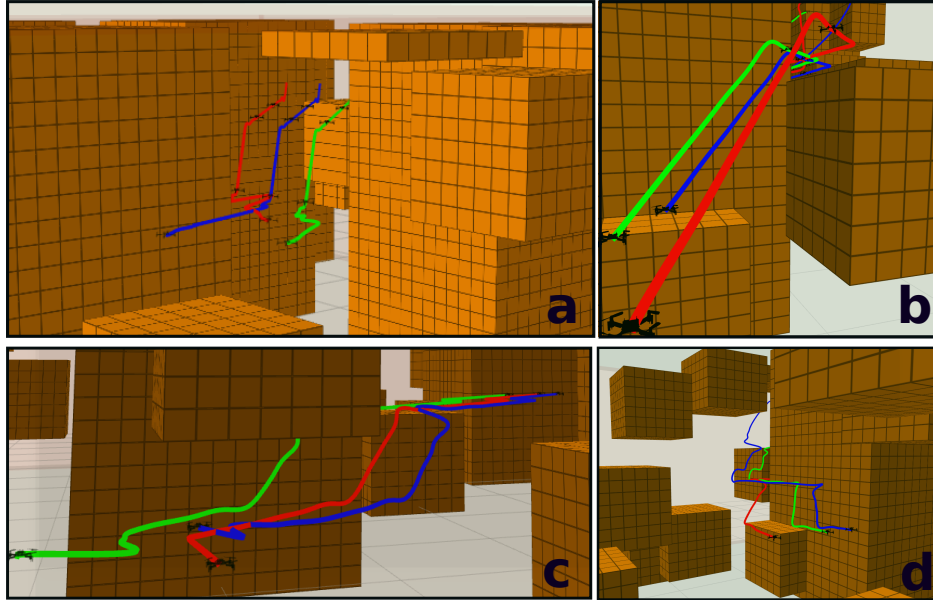


Figure 5.6.1: These four images show representative examples of coordinated actions performed by a formation of robots during search evaluation.

parameters $m = 1$ and $h = 0$.

The results in Table 5.6.1 illustrate that the proposed methodology is able to return solutions that exactly reflect portions of the demonstration even in mildly cluttered environments, as shown in Line 1. As environments become increasingly cluttered, we see that as expected, solutions are forced to deviate in greater degrees from any portion of the demonstration, and that there are a greater number of failure cases where a formation cannot fit along an intended path. However, even in the most cluttered environment, approximately 80% of the solution sequence reflects a portion of the demonstration (the LCS of Line 5 is approximately 80% of the action sequence length).

We additionally show results for the same environments represented at alternate voxel resolutions. We anticipated that searching a coarser resolution would perform faster, and we do see that in general, coarser representations show slightly lower search times and graph sizes. However, the

	Average LCS	Average MDHD	Average MDJSD	Average search time [s]	Average graph size [nodes]	Failed trials
1. Env: low, res: fine	19.000	0.000	0.000	2.196	1157.818	3
2. Env: low, res: coarse	17.227	1.545	0.044	2.184	1124.455	3
3. Env: med, res: fine	18.619	0.381	0.012	2.096	1140.286	4
4. Env: med, res: coarse	17.095	1.476	0.041	2.180	1103.476	4
5. Env: high, res: fine	15.579	2.947	0.079	2.060	1052.895	6
6. Env: high, res: coarse	11.263	6.684	0.196	1.998	1081.368	6

Table 5.6.1: Performance over varying environments. Action sequence length = 19, $|\mathcal{A}| = 92$, $dA = .01$, $dt = .25$, $m = 1$, $h = 0$. Environments “low”, “med”, and “high” are shown in Figs. 5.6.2a, 5.6.2b, 5.6.2c. The voxel resolution of “coarse” is set to 0.25m, and “fine” is set to 0.125m. We report the average values over 25 trials at each environment and voxel resolution.

performance differences in search time and graph size with respect to voxel resolution are minor, showing that the method maintains performance over higher-fidelity environment representations. This is beneficial in avoiding the decreasing solution quality incurred by coarser voxelization; with coarser representations, fewer solutions are found due to the reduced available free space.

Search times reported in Table 5.6.1 are reported as the (average) total search time to find actions along the entire specified path (including all collision checks). Trajectories generated from the discrete states returned from search roughly span a time duration on the order of sample resolution, dt , multiplied by the number of actions in the solution sequence. The search time required is therefore sufficiently less than the travel time of the robots that the methodology can safely evaluate and generate trajectories for online use. We further note that our implementation was performed in MATLAB, and that transitioning to alternate programming languages or libraries will decrease search time.

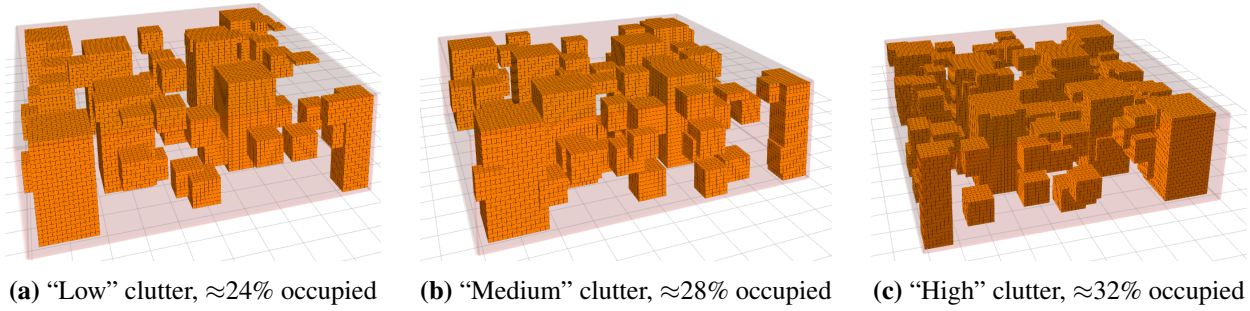


Figure 5.6.2: Environments of varying complexity used in evaluation. Environment labels of “low, med, high” corresponding to Subfigures (a), (b), and (c), respectively, correspond to Table 5.6.1. Environments span $10\text{m} \times 10\text{m} \times 3\text{m}$ in height, and are shown here with a voxel resolution of 0.125m .

5.7 Concluding remarks

The proposed approach contributes the capability to coordinate multi-robot formation planning guided by demonstration, allowing us to reflect user preferences or instructions that are not easily input during online operation. This affords us the ability to coordinate actions with respect to a wider variety or more unstructured representation of user preferences—as represented by the distri-

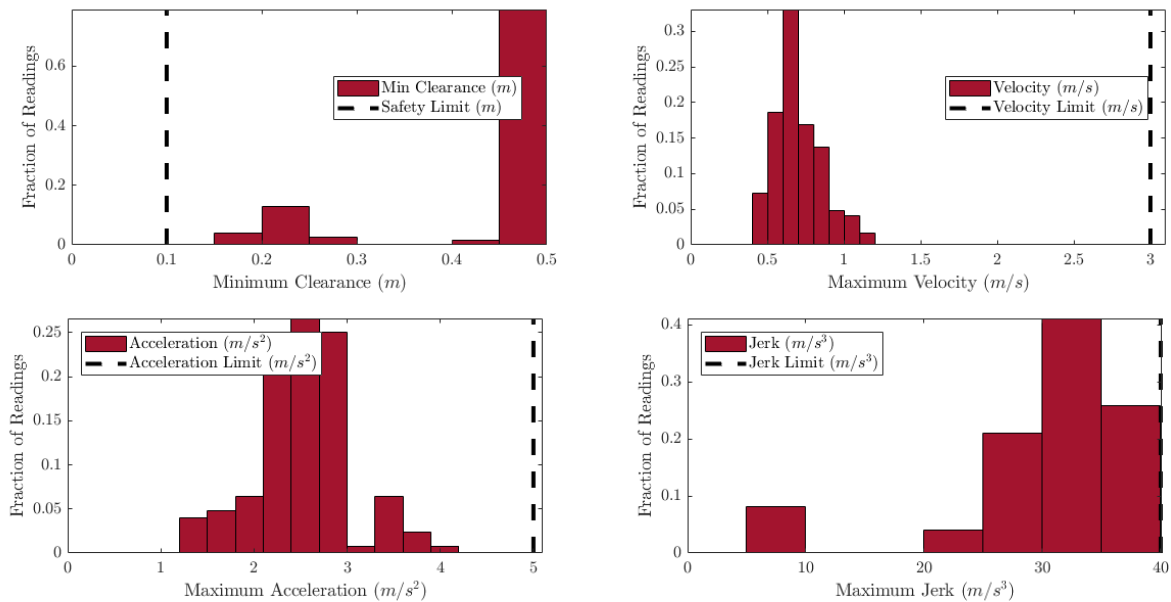


Figure 5.6.3: Trajectory characteristics showing dynamic feasibility: tests employ robots of radius 0.1m , and acceleration and jerk limits of 5m/s^2 , and 40m/s^3 , respectively.

bution of actions in a demonstration—rather than being restricted to minimizing a rule-specified objective function. Simulation results validate that the proposed method is able to find high-resolution solutions over an extended time horizon within a computation budget that allows for online operation, even in complex and cluttered environments. Further, solutions with this approach represent the user-provided demonstration data, showing that the methodology is capable of recreating the specialized behaviors shown by the system of Chapter 3.

Chapter 6

Data-driven search for joystick-directed reconnaissance

The previous chapter described a general representation of multi-robot states and actions, showing that actions can be composed online using a search methodology. With a selection policy based on action probability as observed from a data distribution, this methodology allows for the recreation of preferred motion sequences even in the presence of obstacles. In this chapter, we analyze the performance of the action-based search approach in the context of joystick-directed motion through highly cluttered environments as might be encountered in urban reconnaissance. In this evaluation, we are not seeking to specifically recreate specialized behaviors as previously analyzed. Rather, we leverage data from multiple sources to narrow down the set of possible actions from the space of all possible actions and provide information for online selection.

This chapter focuses on analyzing the methodology of Chapter 5 in greater depth, including a

discussion of method attributes, experiments in simulation analyzing the impact of varying method parameters to characterize performance, and a comparison study with respect to two other chosen approaches for online multi-robot group control in a reconnaissance scenario. Section 6.1 provides an introductory discussion, while Sect. 6.2 discusses method attributes including complexity and completeness. We then perform a series of experiments in simulation (Sect. 6.3) varying method parameters including numbers of robots, action discretization and search set size, and environment to show that the method is general and characterize parameter performance.

6.1 Introductory discussion

We seek to enable a human operator to direct a group of five to twenty aerial robots (quadrotors) at high speed through highly cluttered environments. This capability is of interest in applications such as reconnaissance, where we consider the scenario of an operator using a joystick to drive a group of robots rapidly through a building, entering via a restricted opening such as a window, steering the robot group between rooms and floors, and out of the building to the next target. In this application, we value fast, partial coverage over slow and thorough mapping; high frequency, simple interaction between the operator and the group over long range detailed plans; the freedom for an operator to ignore the collision and dynamic requirements of the robots; and the ability to operate rapidly in complex and cluttered environments. To support these desired capabilities, we therefore require a low-latency, high-frequency human interface of correspondingly low cognitive complexity for online instruction, and the ability to (re-)plan online at high speeds, generating safe (dynamically feasible and collision free) plans for all members of the robot group.

Our problem is motivated by the complex conditions inherent to operating 10-20 robots in highly cluttered and restricted areas in an application requiring human operator expertise via high-frequency interaction. Existing approaches for multi-robot control alone are insufficient to handle the sum of these conditions and requirements, but we can build on existing methods to operate in our chosen problem area. For a high rate of interaction and control, we state that for our application we are specifically interested in driving a group of robots using a joystick interface. Joystick interfaces provide simple input at high frequency, and have been shown to work for human control of a multi-robot group [91]. This approach allows an operator to provide high frequency input to a quadrotor team and uses a vector field based methodology to avoid collisions with objects and other robots. Vector field methods, however, can experience problems when there are many local minima, which happens with increasing frequency as the numbers of agents and obstacles increase. While we therefore propose an alternate motion planning strategy, we adopt a similar interface convention to [91] with additional considerations learned from single-quadrotor joystick teleoperation [87, 88].

With respect to motion planning for multi-robot teams, existing methods generally divide into optimization based and search based approaches. Optimization methods that plan trajectories for groups of quadrotors [20, 58, 67] offer numerous advantages. These methods most often generate polynomial time trajectories with respect to collision and dynamic constraints, yielding assured safe plans over the trajectory horizon. These methods further produce optimal solutions which minimize (or maximize) a desired cost function, and costs of interest may include energy [18] or formation parameters [5]. However, optimization approaches are highly time intensive and

generally infeasible for applications requiring rapid online re-planning. Optimization complexity increases with the number of decision variables, so approaches which consider individual robots in a coupled fashion become intractable with larger numbers of robots or obstacles [20, 58]. More recently, approaches have sought to limit the number of decision variables by optimizing parameters representing the group of agents subject to linear constraints representing only the local environment [5]; while this has been shown to work in online planning scenarios, the environment representation can be limiting in highly cluttered scenarios and optimizing formation parameters can still be time intensive, making it problematic for high-frequency interaction. While a full optimization problem for all robots is intractable in our operating context, we are able to incorporate optimal motion planning methods for polynomial trajectories [57, 66] within our constraints, and we leverage these methods to generate safe continuous plans from the discrete output of our search approach.

Search approaches are frequently used when the number of decision factors in a problem is large. Graph representation coupled with the use of heuristics or other informed selection strategies allow search approaches to tractably refine large decision spaces [77]. With respect to the graph representation (i.e., a chosen discretization for discrete methods, or with respect to chosen parameters), some search methods are also optimal (the path minimizes a cost function) and/or complete (will return a solution or conclude no solution exists in finite time) [85]. Finally, planning dynamic motions based on search solutions has an increased probability (or in some approaches, guarantee) of success [28]. However, coordinating actions between agents during search can be computationally challenging. Planning in the joint configuration space of agents is exponential in the number

of agents, making search with even a limited set of individual actions for each robot potentially intractable with the required number of agents or over longer search horizons; search approaches which plan for agents individually must still resolve inter-agent conflicts [81, 85]. Additionally, edge checks can be expensive depending on the constraints considered, for example, when requiring collision or motion planning computation [14]. Finally, it can be difficult to generate good heuristics to inform search [31, 89]. We employ a search based approach to make tractable the large number of options inherent to coordinating a team of robots in complex environments, and discuss our approach for mitigating search complexity and time considerations in the following sections.

Swarm approaches, or local control methods for individual agents that give rise to global behavior, are frequently used in computation limited applications because it is often more computationally tractable to allow agents to make independent decisions rather than coordinate across agents. We also include in this discussion highly decentralized approaches as frequently employed by reactive control methods, for example, the wide range of approaches stemming from velocity obstacle control methods introduced in [82, 83]. The low computation requirements of these approaches lend themselves well to high frequency interaction applications such as ours, and numerous methods of allowing a human user to direct swarm motion have been pursued [45]. However, local control methods make coordinating the group as a whole highly challenging, a problem generally exacerbated in high clutter environments. Even examples that do seek to encode group behavior at a local level [36] can lead to deadlock between agents and may not provide collision guarantees. Group cohesion and motion guarantees with respect to velocity based control methods have been

addressed by integrating motion planning with higher level planning approaches for multi-agent systems in open environments [2], with multiple aerial robots [4], and small numbers of obstacles [5]. However, these approaches employ optimization based methods to resolve constraints, the time and computation limits of which may be prohibitive for our application as previously discussed. We therefore do not consider local control strategies in this work, although these methods could potentially augment the motion planning portion of the approach.

Due to the numbers of robots considered in our application (group sizes of five to twenty robots), coupled with the desire for safety assurances and ability to coordinate motions between agents, we pursue a centralized, discrete planning approach coupled with polynomial trajectory interpolation for rapid generation of dynamically feasible and safe plans. To allow a user to rapidly interact with a robot team with a high degree of control yet correspondingly low cognitive burden, we employ a joystick interface, mapping joystick input to a desired motion for the robot team using a motion primitive strategy, building on the works of [87, 88, 91]. To find paths that maintain coordination between agents despite clutter, we evaluate a simple Best First Search [69] approach over a selected set of coordinated multi-robot actions. These actions are learned from datasets of real-world multi-agent examples, and are composed online using an informed heuristic about action value. We show that continuous, dynamically feasible plans can be generated from the discrete output of the planner [57, 66]. Our contribution is the formulation of a generalized representation of group actions which allows us to compose actions across multi-agent implementation examples and the online composition strategy. This work establishes the value of the proposed approach with respect to the application of online multi-robot teleoperation, and serves as a base-

line for comparison against further studies, such as the evaluation of alternate search methods or heuristics.

Our decision to employ a search-based methodology faces the aforementioned limitations inherent to multi-robot search, which we seek to mitigate through our approach as follows. We propose to search over a set of actions that describe the motion of a group of robots as a team, or collective. (In order to allow a user to continuously direct all robots in the group, the robots are always considered part of the same collective; we do not allow the group to split.) A collective action representation allows us to search over a set of coordinated actions between agents using a lower dimension representation than the full space of joint actions formed by considering the exponential combination of every individual agents' actions. We propose to use a discrete rather than continuous representation of group actions to create a finite set of possible search options. Although abstracting actions using a group-based representation creates a smaller search space than the joint space of all robot actions, a naive representation at any meaningful discretization still results in a very high number of actions.

We propose to (1) balance the tradeoff between action resolution and computational tractability, and (2) determine a heuristic policy for guiding search, by leveraging data from real-world multi-agent examples. For a desired action fidelity, i.e. discretization, a uniform sampling over the space of actions forms a large set without additional information indicating the potential value of any action. Using data taken from wide-ranging real-world multi-agent motions including: biological examples such as flocks of birds or human crowd data; engineered actions such as rigid pattern following in multi-agent theatrical applications; and reactive motion data exhibited by

swarm multi-robot applications, we can represent agent motions as group actions at our chosen discretization and form our search set from these observed actions. Datasets additionally provide information we can use to estimate the relative value of an action. For example, frequently observed actions may be hypothesized to be of higher value compared to less frequently performed actions. We may additionally estimate value by examining action sequences, as individual actions may have a much higher probability of having been performed based on previous actions. We therefore evaluate action sequences observed from data and use probabilistic measures of action occurrence as a heuristic to guide search.

6.2 Methodology Attributes

Our approach is characterized by our choice to search over discrete actions for groups of agents.

We realize two benefits from the decision to model actions at the group level with respect to the numbers of agents considered. One, actions at the group level allows graceful scaling with respect to search time as the numbers of agents increase. While search does need to ensure that actions produce no collisions between agents, this check can be performed after a check between the group and the environment. This reduces the number of inter-robot collision checks needed, so we scale at less than the exponential growth rate expected for inter-agent collision checking for larger team sizes. Additionally, the group-based action model means that actions are applicable across group sizes. Actions are therefore general with respect to the number of agents in the group, and the action set may be applied to groups of any numbers of agents.

A further advantage to the proposed approach is that it works in highly complex environments.

The method can find solutions in the presence of complex obstacle configurations, i.e., in areas where free space is non-convex, because the method makes no assumptions about or exclusions of available space. While we operate without assumptions about environment configuration, this approach means that the method transfers across environments, making it a general approach that can be used across operating scenarios.

Due to the chosen discrete nature of the action representation, we can provide guarantees with respect to search completeness and safety. However, the conditions required to yield guarantees may be restrictive during actual operating scenarios and so in practice, it is more desirable to relax bounds in order to eliminate computation steps and act quickly. However, all final solutions are represented as time-parameterized motion plans and are safety checked for feasibility and collision, and so we can ensure that no robot will ever start a plan without knowing if it is safe.

Disadvantages to the method are that search space is large and edge checks can be expensive. However, we can overcome these disadvantages by leveraging prior data to first, refine the space of search actions and second, by using heuristics to bias search selection, minimizing edge expansions. Despite the limits of the approach, our analysis shows that this methodology performs better than state of art approaches and in practice allows operation in previously untenable scenarios. The following subsections discuss method completeness, complexity, optimality, and safety guarantees in greater depth.

6.2.1 Method Completeness and Complexity

The structure of our chosen selection heuristic, i.e. reward function (Eqn. (5.4.1)), during search means that our search approach performs equivalently to a Depth First Search approach. Depth First search is complete when it is done in finite spaces and a list of explored nodes is kept (as is performed in our approach, see Alg. 1), so that explored nodes are never re-added to the frontier [69]. The method will expand nodes in the graph until a solution is found and will not re-add an explored node, therefore creating no infinite loops; the method will therefore terminate when a solution is found or, by exploring every node in the graph, conclude no solution exists. A toy example illustrating method completeness is shown in Fig. 6.2.1, where the algorithm is forced to expand every node in the graph to find the solution.

For a breadth b and search depth d , the time complexity of the method is therefore $O(b^d)$ in time and $O(bd)$ space; by performing lazy edge expansion (collision checking only the chosen node to expand, rather than collision checking all child nodes to determine which are feasible), however, it is possible in the best case for the method to also realize $O(bd)$ time.

Therefore, although potentially large, because we have chosen a finite discrete formulation and a complete search method, the search method is complete with respect to the enumerated search set of actions: we will return a solution as a sequence of discrete actions or conclude in finite time that no sequence of actions from the enumerated search set exists.

6.2.2 Optimality

For a problem definition that only considers finding an admissible (collision free) path from the root node to any leaf node, all leaf nodes are equivalent, and all edge costs are uniform, and so by this definition any path from root to leaf is therefore optimal and minimum cost. For any other cost consideration, optimal search methods could be considered because the discrete problem enumeration is finite.

However, our proposed search heuristic does not explicitly consider commonly desired solution qualities such as smoothness, energy, or trajectory time. Because search actions are taken from dynamically feasible examples, and because action reward is based on action ordering and so solutions generally replicate demonstrated action sequences (Chapter 5), the method implicitly captures some notions of desired solution traits such as smoothness, energy, or trajectory time. But trajectories generated through keyframe sequences found via search are not guaranteed to be optimal with respect to these commonly desired trajectory characteristics because we do not explicitly evaluate the true cost of a path with respect to these metrics during search.

We have chosen to use a Best First Search approach. Our described selection metric effectively ranks all child nodes from a given parent, and evaluates each child in order of highest reward until finding a non-colliding child. The algorithm therefore explores all child nodes from a parent before backtracking (if no child is collision free). This means that the algorithm is not guaranteed to return the most likely string of actions overall, but rather, prioritizes deeper nodes in the tree, to push towards leaf nodes as quickly as possible. This can mean that we can encounter situations where search explores actions that have not been frequently seen given preceding actions, forcing more

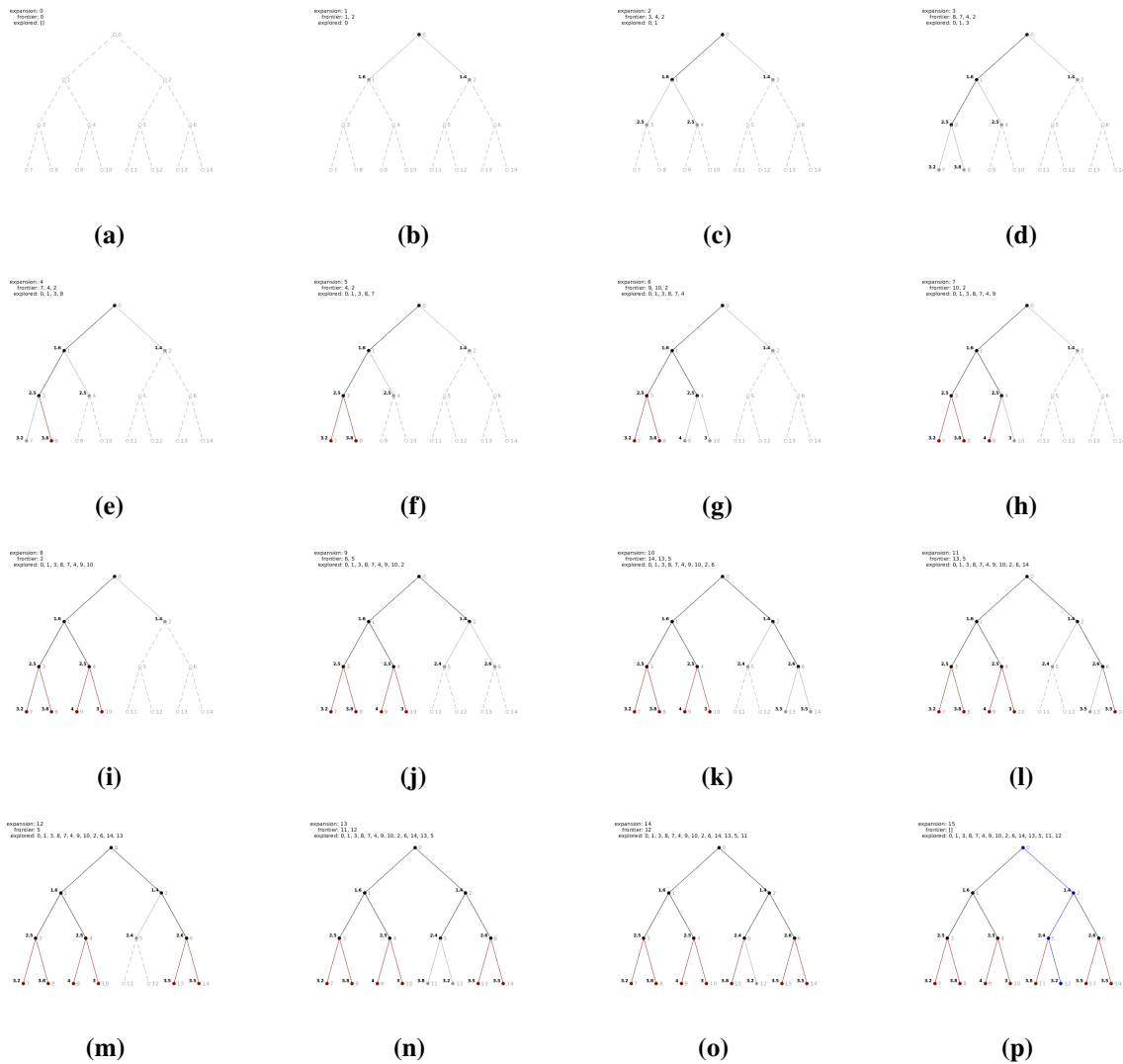


Figure 6.2.1: Toy search example illustrating completeness.

uninformed exploration. Alternative selection heuristics that downweight node depth, or exclude it entirely, could in contrast backtrack earlier in search to evaluate sequences that play out from an earlier option in the tree. This could potentially find more likely action sequences over longer horizons or avoid expensive evaluation in little explored areas. In this work, however, we evaluate the proposed selection metric in order to gain a baseline understanding of method performance and defer the evaluation of different search approaches and selection heuristics to future work.

6.2.3 Solution Safety

This section describes solution and safety guarantees of the proposed method provides.

We cannot guarantee a solution exists because we search a parameterized and discrete space. First, user input is parameterized, as the user selects motion primitives from a library [87, 88], which therefore excludes possible solution space. We also then choose to search at discrete waypoints along the user-selected motion primitive. Second, we find keyframes by searching over a discrete, linear, finite set of actions. This chosen parameterization excludes possible motions between agents, restricting motions to fall within an affine definition. Third, collision checking at all levels prioritizes safety at the extent of solution space.

We collision check the user's path input using some minimum bounding volume. A reasonable bounding volume in practice may exclude solutions. The safest, but overly-optimistic, estimate could be to consider a single robot volume along the user-specified path. Obviously however, all robots cannot collapse to the volume of a single agent, so this admits only solutions which might collapse the formation to one-robot diameter in some dimension such as a line or plane, leaving search to collision check against actual obstacle locations. The search method doesn't incorporate environment information except for edge checks, so, while a potential solution may exist to the 1-robot swept volume boundary, and the search would then find it given time to expand the whole graph, the method is unlikely to do so in a practical amount of time. A better minimum volume is a ball known to at-minimum fit all robots closely packed, as multiple scaling actions from the search set can fit robots inside such a volume.

During search, we estimate robot and group shapes using convex polytope models, so while

it may be small, there is excluded space due to overestimating a robot shape because of planar bounding volumes and convex group representations.

Guarantees with respect to the feasibility of continuous polynomial solutions fit through the discrete search solution: If we ensure the discrete search meets conditions of CAPT [81], we can guarantee a continuous trajectory solution for every robot that is dynamically feasible and collision free.

1. We consider states X_t to be at rest, with zero-valued higher order dynamic terms.
2. The edge between states X_t and X_{t+1} is collision free if the convex hull containing X_t, X_{t+1} contains no obstacles.
3. The distance between each robot's start and goal locations is greater than δ , $\|x_{t+1} - x_t\| > \delta$, where δ is a function of the robots' radii as stated in [81].

These three conditions mean that if we cannot fit a polynomial through the discrete goal points as keyframes, a dynamically feasible and collision free solution for all robots is guaranteed to exist because every instance of successive keyframes can be solved through an application of the CAPT problem [81].

In practice however, the above three conditions produce non-desirable solutions. The at-rest assumptions produce trajectories that are not aligned with motions desirable to joy-stick control, while convex-hull free-space checks increase search time and undesirably limit solution space. In all of our results, we therefore evaluate a methodology which checks only for collisions at keyframes during search.

Despite relaxation of bounding volume limits during search, we perform computation-timed feasibility and collision checks on all polynomials generated through any keyframes returned from search. We therefore guarantee that any polynomial trajectory returned is dynamically feasible

and collision free, and in the case of failure, robots will never start a non-safe trajectory. We can therefore guarantee safety but not the existence of a solution.

6.3 Evaluation

In this section, we evaluate the proposed approach with respect to computation and solution attributes. We are interested in determining the feasibility of incorporating prior data into search through a probabilistic selection policy; relative method scaling characteristics associated with elements including numbers of robots, action fidelity, and environments; and the utility of the proposed methodology for a reconnaissance scenario, especially in comparison to existing approaches. We begin with a brief look at the diversity of the actions learned from the varying sources of data, then discuss methodology scaling and solution quality as factors of varying parameters.

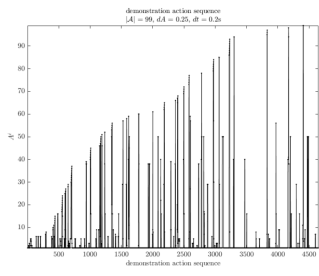
6.3.1 Data attributes

As discussed in Sect. 5.3, we create actions from the trajectories or position time series of multi-agent group datasets. In Chapter 5 we used demonstrations, or datasets, from our previous multi-robot theatric system. In this chapter we broaden the considered actions beyond the engineered actions of rigid pattern following from the theatric application to include biological multi-agent examples as well as actions exhibited by reactive multi-robot methodologies. The datasets used therefore represent multi-robot swarm actions [84], flocking data (from pigeons) [59, 60], and chosen formation patterns [15, 16].

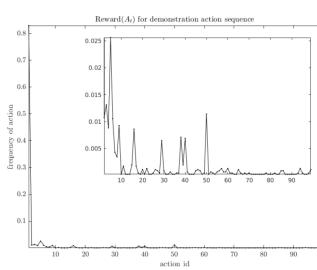
An example of an action sequence, and corresponding action distribution, from a dataset time-

series of agent positions is shown in Figs. 6.3.1a and 6.3.1b. Fig. 6.3.1c gives an idea of how actions, and more clearly, action sequences, differ across the different data sources. To measure the similarity—or conversely, the divergence—between two distributions we use the Jensen-Shannon divergence metric [56], where a measure of 0 means distributions are identical (i.e., do not diverge at all) and a measure of 1 means the distributions share no similar elements (completely diverge).

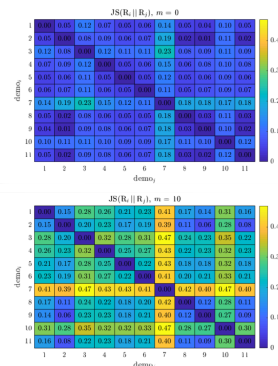
The figures give a qualitative example of what a demonstration looks like in terms of uniquely identified actions. We see that earlier actions are reused, and that not just actions but action sequences may be repeated. Holding a given action discretization common across datasets, we see



(a) An example action sequence. The data set is processed sequentially; each action, A_t , relating agent positions at $t + 1$ to t , is discretized here by $dA = .25$. The x -axis shows the sequential progression of time, and the y -axis shows the label given to each newly encountered unique action. We see that a group of agents tends to re-use actions, and as time progresses, new actions are seen less frequently. In this example, 99 unique actions are seen over a course of over 4500 datapoints.



(b) This figure shows the frequency distribution of actions from the action sequence in 6.3.1a. The normalized frequency (y -axis) is shown for each unique action in the 99 action set (x -axis). A large portion of the distribution corresponds to the identity transform, i.e., maintaining a static formation. The inset figure shows a magnified view of the action distribution, neglecting the identity transform.



(c) The top figure shows the correspondence of actions across datasets. The bottom figure shows the divergence between datasets when we consider actions in sequence—here, we consider sequences of 10 actions long. As expected, the top figure shows that while some actions are unique to different applications, many actions are common across multi-agent groups. What is clearly different between datasets is how actions were ordered to allow agents to flock or perform theatrics, etc.

Figure 6.3.1: Figs. 6.3.1a and 6.3.1b are examples of an action sequence and distribution from a dataset. Fig. 6.3.1c shows how actions differ across datasets, and that differences between datasets is more clearly evident when considering action ordering.

that actions might be repeated between datasets but that data sets begin to differ significantly when viewed in terms of sequences of actions.

6.3.2 Solution attributes

To characterize the proposed approach, including both the utility of the search policy and the quality of trajectories interpolated through the discrete output of search, we quantify several metrics as defined in Table. 6.3.1.

As mentioned in Sec. 6.2.3, we cannot guarantee the existence of a solution, but we do enforce that the method returns a solution or a notice of failure in finite time and that no robot will perform an unsafe trajectory. Therefore, we note that we impose a time limit with respect to search over actions, and we mark any trial as failing if:

1. search fails to find a solution within the specified time limit;
2. trajectories interpolated through the discrete search solution cause inter-robot collisions;
3. trajectories interpolated through the discrete search solution cause robot-obstacle collisions.

Therefore, metrics reported include method success and search time.

For every trial where search returns a discrete action sequence, a trajectory is interpolated for each robot in the group through the resulting position sequences. Evaluating every robot's trajectory at a finely discretized time resolution, we show that every trajectory for every robot is collision free by reporting the smallest experienced clearance distance between any two robots and similarly, the smallest experienced clearance between any robot in the group and any obstacle. Likewise with respect to dynamic feasibility, we report the maximum values of velocity, acceleration, and jerk experienced by any robot in the group.

Notation	Units	Definition
method success	boolean	feasible solution found
search time	seconds [s]	time to find a discrete solution action sequence
Group trajectory feasibility metrics:		
$\Delta r_{\min}^{\text{robot}}$	meters [m]	minimum clearance between any two robots
$\Delta r_{\min}^{\text{obstacle}}$	meters [m]	minimum clearance between any robot and obstacle
v_{\max}	m/s	maximum velocity of any robot
a_{\max}	m/s^2	maximum acceleration of any robot
j_{\max}	m/s^3	maximum jerk of any robot
Group trajectory quality metrics:		
d_{\max}	meters [m]	maximum distance traveled by any robot
t_{\max}	seconds [s]	maximum time required by any robot
\tilde{E}_{\max}	m/s^2	maximum energy [†] required by any robot

[†] Energy is proportional to required thrust over the course of a trajectory. To calculate \tilde{E} , we neglect mass, and sum the net acceleration required over an entire trajectory: $\tilde{E} = \sum_{t=1}^T \|a_t - a_{t-1}\|$.

Table 6.3.1: This table describes metrics quantifying both the discrete search process and the dynamically feasible trajectories interpolated through the search solution.

Finally, we also report metrics of distance traveled, energy required, and total travel time for the resulting trajectories. While we do not explicitly seek to optimize these criteria during search, these qualities reflect desirable motion attributes that have a dependency on action choice. Given trajectories for every robot in a group, we therefore report the greatest distance, the largest amount of required energy, and the longest time required by any robot across the group.

6.3.3 Experimental setup

In order to evaluate the proposed approach, we run large numbers of trials through randomly generated cluttered environments. In every trial, robots are randomly placed in the environment and are directed to follow a randomly selected motion primitive. Each motion primitive is sampled at ten evenly spaced waypoints, meaning that we perform search over a horizon of ten steps, i.e.,

to a tree search depth of ten. Environmental collision checking during search is performed by checking the convex hull formed by the robot group against environment obstacles.

In all trials across all experiments, parameters used to determine solution feasibility—search time limit and robot physical parameters (radius and higher order dynamic limits)—are kept constant. Parameters specific to any individual experiment are described in their respective sections, with a detailed explanation of random environment characterization given in Sect. 6.3.7.

In the following sections, we seek to establish first, the validity of the method: that biasing action selection based on prior data is advantageous to motion planning (Sect. 6.3.4). Following this, we examine in more detail how the methodology generalizes across action discretization, number of robots, and environment (Sects. 6.3.5-6.3.7, respectively) by evaluating method performance as respective parameters are varied. We conclude by examining method performance relative to two existing approaches for group-based multi-robot control in an example reconnaissance scenario (Sect. 6.3.8).

6.3.4 Validity of biased action selection

We do not focus on what operating scenario, application, or model generates the data from which we construct multi-robot actions. We simply leverage the existence of such data to provide information as to what actions, over the entire space of possible actions, have a more likely probability of being feasible. We additionally benefit from the fact that we take actions from dynamically proven examples. Biasing search to examine sequential actions then expresses our belief that sequential actions will, despite their reuse with respect to a new state, similarly hold feasible and

	method success [%]	average search time	average $\Delta r_{\text{robot min}}$	average $\Delta r_{\text{obstacle min}}$	average v_{max}	average a_{max}	average j_{max}	average d_{max}	average \bar{E}_{max}	average t_{max}
randomSelect	84.60	0.38	0.29	0.45	1.61	3.19	17.39	6.42	36.25	10.80
biasSelect	90.10	0.44	0.31	0.49	1.35	2.93	14.58	4.94	34.58	9.44

Table 6.3.2: This table reports the average values of metrics defined in Table 6.3.1 over 1000 trials for each search method. A group of five robots and 999 actions represented at a discretization of 0.40 were used for both methods. Values typeset in **bold** show the better value for an indicated category (i.e., lower values where minimization preferred (time, energy, etc.), greater values where maximization preferred (clearances)). (When not specified, metric units are given in Table 6.3.1.)

yield solutions with similar properties (minimizing unneeded energy, distance, or time, for example). While it is likely that additional benefit could be derived from a more informed linking of provided data and operational reuse, such as using expert demonstrations for exploration specifically in a new exploration scenario, we defer this to future work.

This section therefore examines our approach’s fundamental assumption that it is valuable to bias search on action sequence probabilities generated from dynamically feasible examples, even when we do not specifically account for the model that generated the action distribution. To evaluate the validity of the proposed methodology, we evaluate search success and time, as well as the resulting interpolated trajectories formed through the discrete solutions returned from search, using a set of 999 actions. We compare against a performance baseline where search selection assumes all actions are equally likely and so must randomly select actions as no prior information is provided to bias selection. Then, we leverage data taken from [15, 16, 59, 60, 84] (Sect. 6.3.1) to inform selection.

For each method, random selection versus biased action selection, we run 1000 trials. Each method is evaluated over the same set of randomly chosen start states and correspondingly random chosen motion primitives through the same environment. We record the metrics discussed in Table 6.3.1, Sect. 6.3.2, for every trial and report summaries of these values in both Table 6.3.2 and

Fig. 6.3.2.

In Table 6.3.2, we summarize the results of all trials for each method by reporting the average values over all 1000 trials. This shows that despite a small increase in search time, biasing action selection as described in Chapter 5, Sect. 5.4.2 leads to a higher success ratio and overall higher quality trajectories—trajectories interpolated though actions selected based on provided data are on average shorter, smoother, require less energy, take less time, and robots maintain further spacing

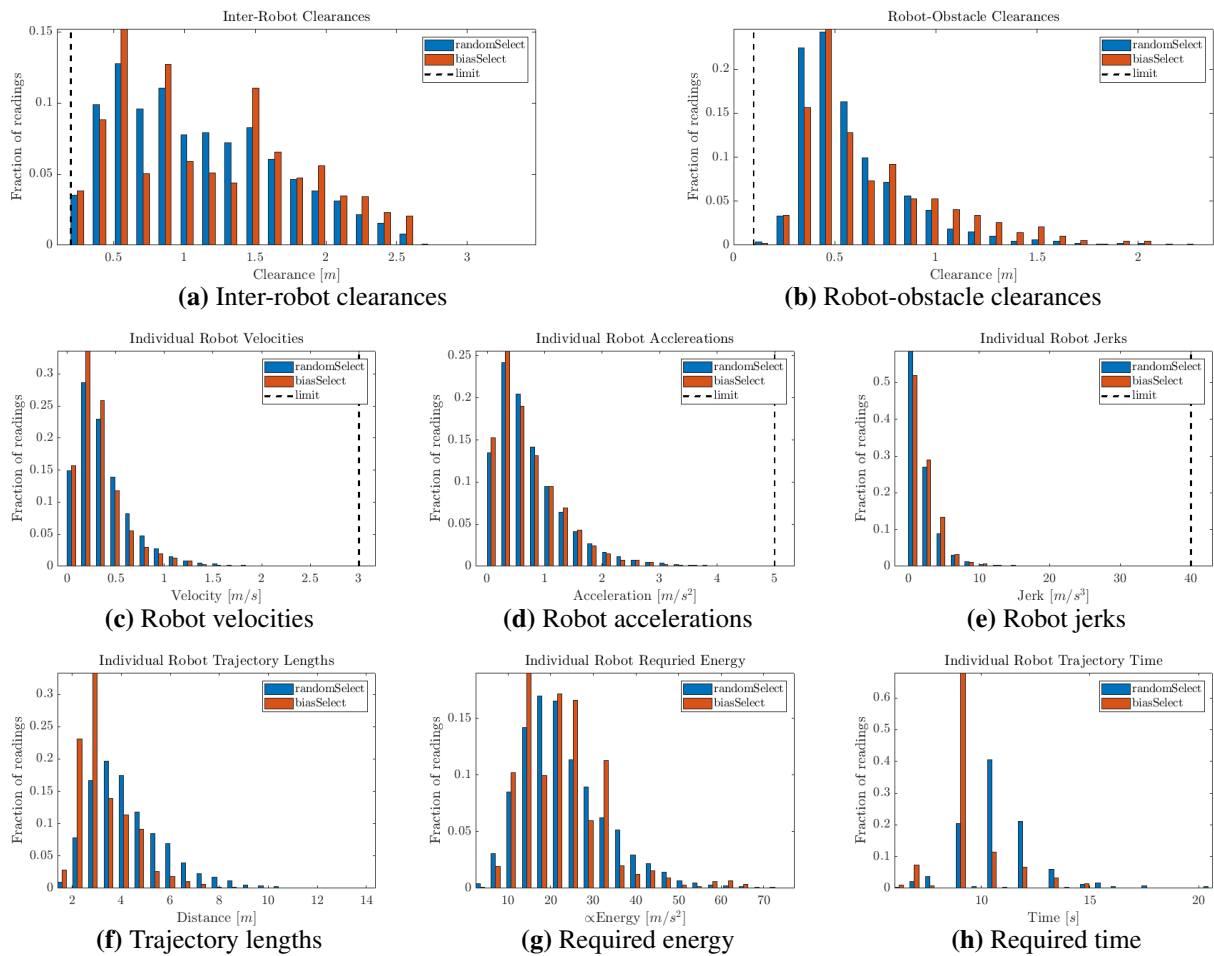


Figure 6.3.2: These figures report measurement histograms associated with all individual robot trajectories over all trials for the evaluated random and biased selection heuristics. In Figs. 6.3.2a-6.3.2e, multiple measurements taken over every robot trajectory show that all trajectories meet all feasibility requirements: all inter-robot and robot-obstacle clearances are above the defined limits (Figs. 6.3.2a-6.3.2b), and all dynamic trajectory terms are below the defined limits (Figs. 6.3.2c-6.3.2e). Figs. 6.3.2f-6.3.2h report the single value for every individual robot’s trajectory with respect to total distance traveled, energy required, and travel time.

from each other and obstacles.

Figure 6.3.2 reports the distribution of individual measurements taken over all trials for both methods. Reported as histograms of measurements, we can see that the distributions for clearance distances for biased search are shifted to the right, showing that over all trials, robots experience greater clearances compared to random selection (Figs. 6.3.2a=6.3.2b). We also see that the distributions for biased search with respect to overall trajectory length is shifted left (Fig. 6.3.2f), showing that all trajectories are on average shorter—i.e., individual robots are not made to travel through many extraneous motions as the group follows the selected motion primitive. This shorter trajectory distance results in shorter travel times (Fig. 6.3.2h), despite similar velocities and accelerations (Figs. 6.3.2c-6.3.2d); likewise, this results in less overall energy usage on average (Fig. 6.3.2g).

These results demonstrate that biasing selection to follow previously demonstrated, dynamically feasible action sequences can lead to overall better-quality motions in terms of time and energy as well as more frequently finding valid solutions.

6.3.5 Generalizing with action resolution

We wish to characterize the relationship between action resolution, dA , search set size, $|\mathcal{A}|$, and the quality of the trajectories interpolated through the resulting solutions. These factors are dependent on each other, and evaluating the performance of various parameter combinations may give a system designer insight into how large a search set or fine an action fidelity an application can afford when choosing between online run time and desired solution attributes.

To evaluate tradeoff in action discretization and search set size we evaluated a single data set at

a constant sample time. We then represented actions from the data set over a range of discretization values, dA . The size of the discretization affects the overall number of unique actions we realize from the dataset. Finer discretizations allow us to represent more actions, and the relationship is shown in the top subplot of Fig. 6.3.3a.

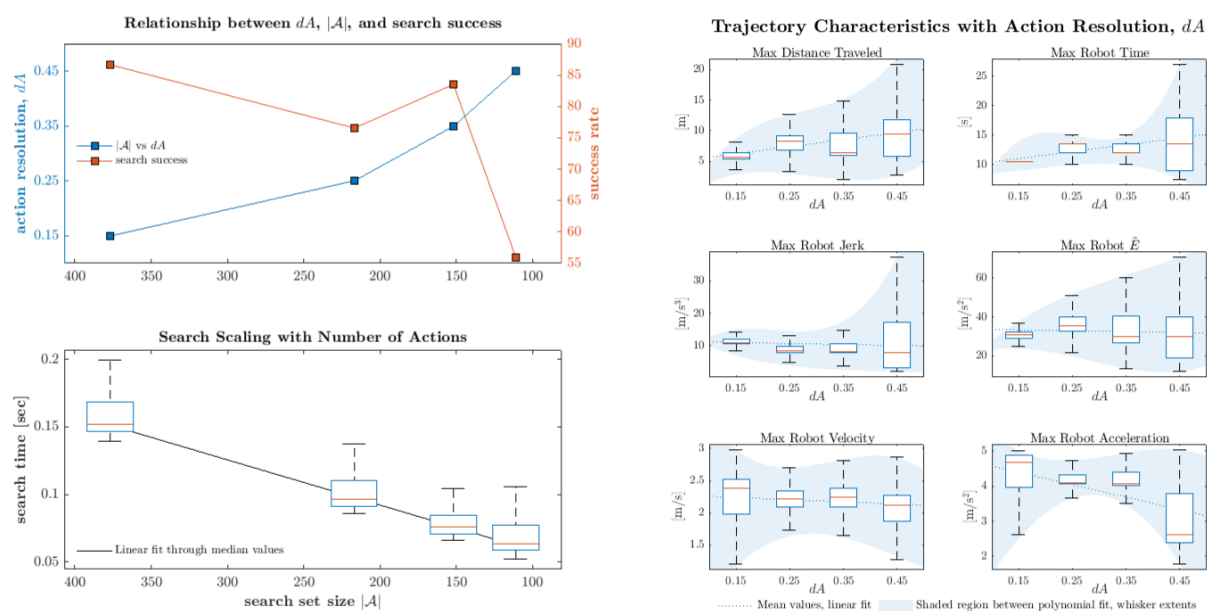
To evaluate the differences in performance between the actions sets, we performed search over the same 1000 paths through the same environment for the same group of robots, varying only the action discretization (and corresponding size of the action set).

Broadly, the results support our expectations. Finer resolution actions have a higher success rate, as they provide more options to find solutions that avoid obstacles without creating inter-robot collisions (success rate is shown in the top subplot of Fig. 6.3.3a). However, larger numbers of actions in the search set increases search time, as the search methodology evaluates the probabilities of all actions at every node. Holding other values of search constant (such as planning horizon), this only results in a linear increase in time $O(n)$, as shown in the bottom subplot of Fig. 6.3.3a.

We hypothesize that having access to finer resolution actions should result in higher quality trajectories. For example, finer resolution action sequences should allow for smoother motions, or paths that do not need to widely deflect to avoid obstacles. Although the data is noisy, the trends shown in the plots of Fig. 6.3.3b support this hypothesis. Figure 6.3.3b reports data for all trials that were successful across all dA values. This allows us to compare the different solutions over the same paths. Although the average values do not have a steep slope across dA values, values for distance and time trend as expected. What is most supportive of the relationship between action fidelity and quality, however, is that the variability of the trajectory metrics increases as action

discretization increases—this is especially noticeable on the top four subplots of Fig. 6.3.3b.

Looking at the subplots of Fig. 6.3.3b in more detail, we first look at the top row, reporting for each trial the maximum distance traveled by any robot and the maximum time required (top row, left and right subplots, respectively). The distance plot makes clear that the robots travel increasing distance with coarser actions, and that more coarsely discretized action sets have greater variability—greater likelihood of traveling greater distance. This means that to obey the imposed dynamic limits of the robots, trajectories require longer time spans to cover the increasing distance.



(a) The top subplot shows the relationship between action resolution and search set size, as well as the correlation with search success. The bottom subplot shows the distribution of search times for each trial performed at each resolution.

(b) This figure reports the statistical distribution of results for different quality metrics for the trajectories interpolated through search solutions at different action resolutions.

Figure 6.3.3: These figures report the statistical distribution of search performance and trajectory quality over varying search sets, corresponding to increasing action discretization. In Fig. 6.3.3b, the tops and bottoms of each box are the 25th and 75th percentiles of the samples, respectively. The distances between the tops and bottoms are the interquartile ranges. The line in the middle of each box is the sample median. If the median is not centered in the box, it shows sample skewness. **Note:** Action set size, $|\mathcal{A}|$, in Fig. 6.3.3a decreases moving to the right along the x -axis. This is done to align with the plots in Fig. 6.3.3b, where trials are shown ordered by corresponding (increasing) action resolution, dA .

When interpolating trajectories through search results, we base an initial estimate for a trajectory time span on the planning horizon along the desired motion primitive and discretization used in creating the action. If the initial time estimate proves to require higher order terms that violate the robot limits, we increment the estimated time span by a set value and re-interpolate. This approach is reflected in the time distributions shown. At a dA of 0.15, all trajectories are feasible within the initial time estimate. At the next two action discretizations, trajectories require a greater time increment, but the samples are skewed below the median at $dA = 0.25$, and above the median at $dA = 0.35$. A far larger time increment is needed then for trajectories calculated using a dA value of 0.45.

Moving to examine the middle row of subplots in Fig. 6.3.3b, we can see that the increased distance at coarser discretizations results in wide variability in experienced jerk. Interestingly, because of our stepped time-incrementation approach to quickly resolving dynamic feasibility, the average trend line has a slight negative slope because extra time was effectively provided for those trajectories which did not need the full step value. The variability in required energy for robots to cover trajectories likewise increases directly with coarser action resolution.

Velocity and acceleration are shown in the bottom row of subplots in Fig. 6.3.3b. Here we actually see decreasing average values with increasing action discretization, but this is not counter to our hypothesis. This downwards trend is a result of high jerk values forcing increased time scaling to realize dynamic feasibility, to the degree that velocities and accelerations actually decrease to realize dynamically feasible jerks.

6.3.6 Scaling with group size

Evaluating an action’s feasibility during search requires two types of collision checks: a check to ensure that the group of robots is collision-free with respect to the environment, and a check to ensure that no robot collides with another robot. These two collision checks use different robot and group geometry representations, and scale at different orders with the number of robots (group size).

Individual robots are modeled as spheres, and performing inter-robot collision checking scales at $O(n^2)$. With respect to the environment, voxel grid representations as we use in this work are generally fast to query. However, this carries greater risk that interpolated trajectories through the resulting positions can lead to the intersections with occupied voxels that might exist between the discrete solution states, requiring increased need for trajectory time-scaling to approach straight-line paths.

Collision checking a geometrical representation of the entire robot group reduces this risk at the expense of some loss of solution space. Using a spherical model over the group as we did for individual robots yields fast collision calculation but is overly conservative and can greatly decrease the solution space. We therefore perform all group-environment collision checks in this work by computing a convex hull around the robot group, as we find through experience that this balances collision checking time with a reasonably solution success rate.

Fig. 6.3.4a shows plots illustrating the time scaling of varying group-geometry representations in collision checking, with Fig. 6.3.4b showing associated average search success. Fig. 6.3.4c shows the respective portions of search time shared by inter-robot and group collision checks.

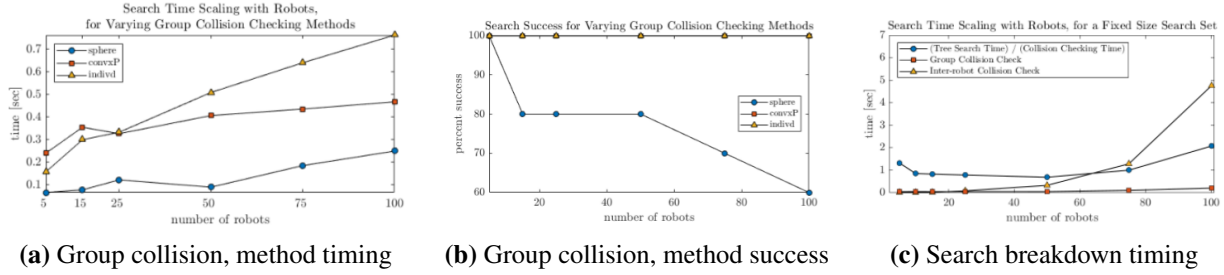


Figure 6.3.4: Fig. 6.3.4a and Fig. 6.3.4b show average timing and success values, respectively, for group collision checking methods scaling with numbers of robots. Fig. 6.3.4c shows the respective average timing values broken down for collision checking in search, using a convex group representation for group-environment collision checks, scaling with number of robots.

Group-environment and inter-robot collision checks can be performed serially at each edge-check during tree search and an edge marked as infeasible as soon as the first check returns false. Due to the different scaling orders of the respective collision checks with respect to the number of robots, the quicker, group-based check can be performed first to minimize the number of calls to the more time-consuming inter-robot collision calculation. While there is no way to avoid at least one instance of inter-robot collision checking per action selected, the group collision check can be used to avoid the worst case exponential performance we would see if we required inter-robot collision checking with every edge check.

6.3.7 Generalizing with environment complexity

Highly cluttered environments frequently invalidate action options due to collision. In this section, we evaluate how the degree of environment clutter and environment resolution affect search (Fig. 5.6.2).

To tractably generate parameter-defined highly cluttered random environments, we describe randomly generated 2.5D environments—i.e., environments experience variation in x and y dimen-

sions, but are extended to form columns in the z dimension. Obstacles are randomly generated and randomly shaped: they are defined in the (x,y) plane to have non-smooth boundaries (to within the specified voxel resolution) and are not restricted to convex shapes.

We define two parameters to describe obstacle configurations / environment: we specify voxel grid resolution, v_{res} , and obstacle spacing, $d_{\text{obs}}^{\text{ref}}$. The number of obstacles, obstacle dimensions, or percent of occupied voxels are not strictly controlled. We do ensure, however, that all obstacles are

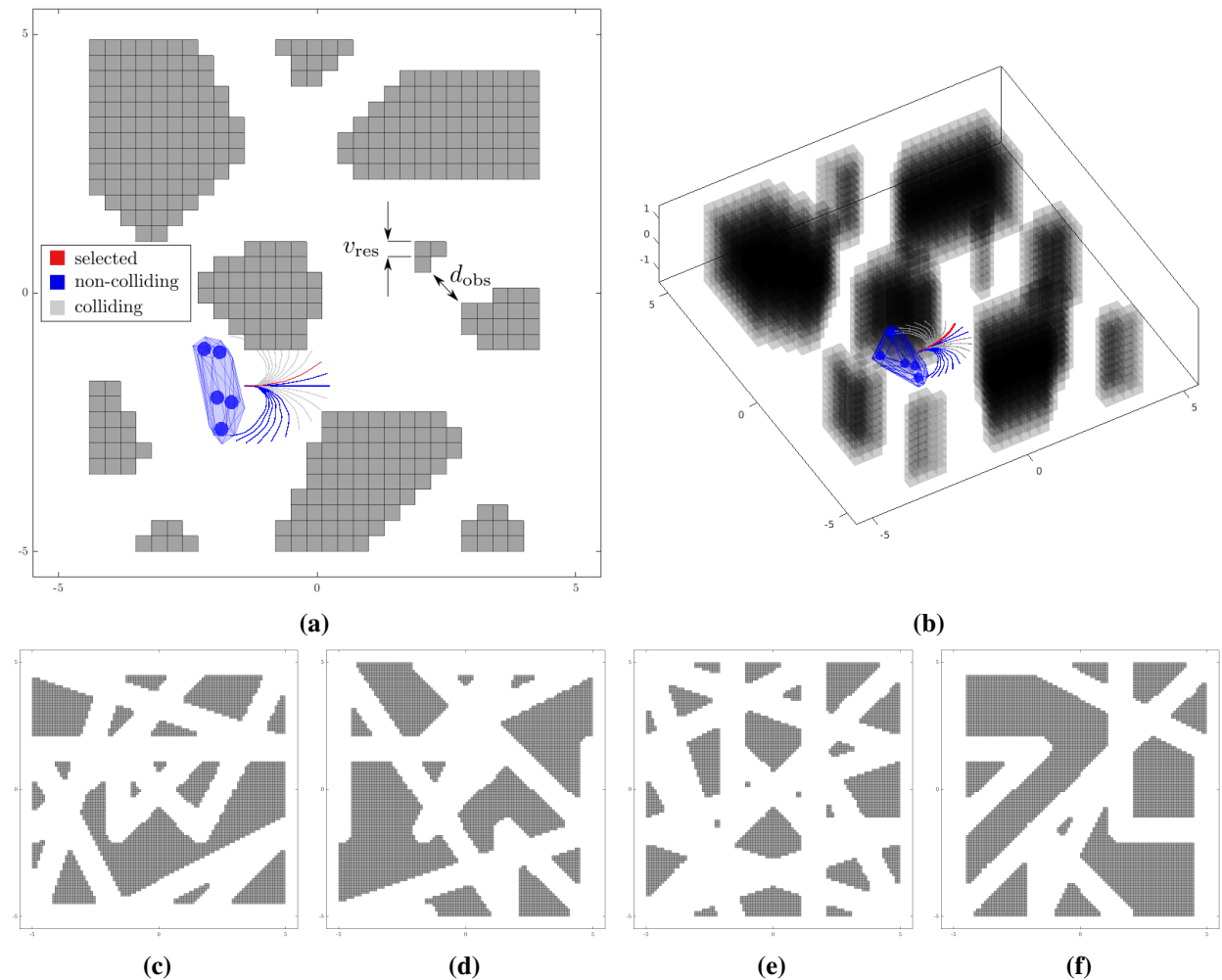


Figure 6.3.5: Top-down illustration showing environment parameters (Fig. 6.3.5a) with associated 3D view (Fig. 6.3.5b). Figures 6.3.5c-6.3.5f highlight commonly described environment features.

separated by d_{obs} to within the voxel resolution: $d_{\text{obs}} = d_{\text{obs}}^{\text{ref}} \pm v_{\text{res}}$. An example of these defined parameters in relationship to an environment is shown in Fig. 6.3.5a, and the 3D view of the same environment in Fig. 6.3.5b.

We choose this generation approach because it effectively allows the random generation of typically tested environment features such as open areas, corridors, bottle necks, and concave obstacle examples, with randomly patterned obstacle “texture” such as rough or smooth edges (again, to within the specified voxel resolution). Example environments are shown in Figs. 6.3.5c-6.3.5e.

We use a group of 5 robots, with all robots having a radius, r , of 0.1m and randomly placed

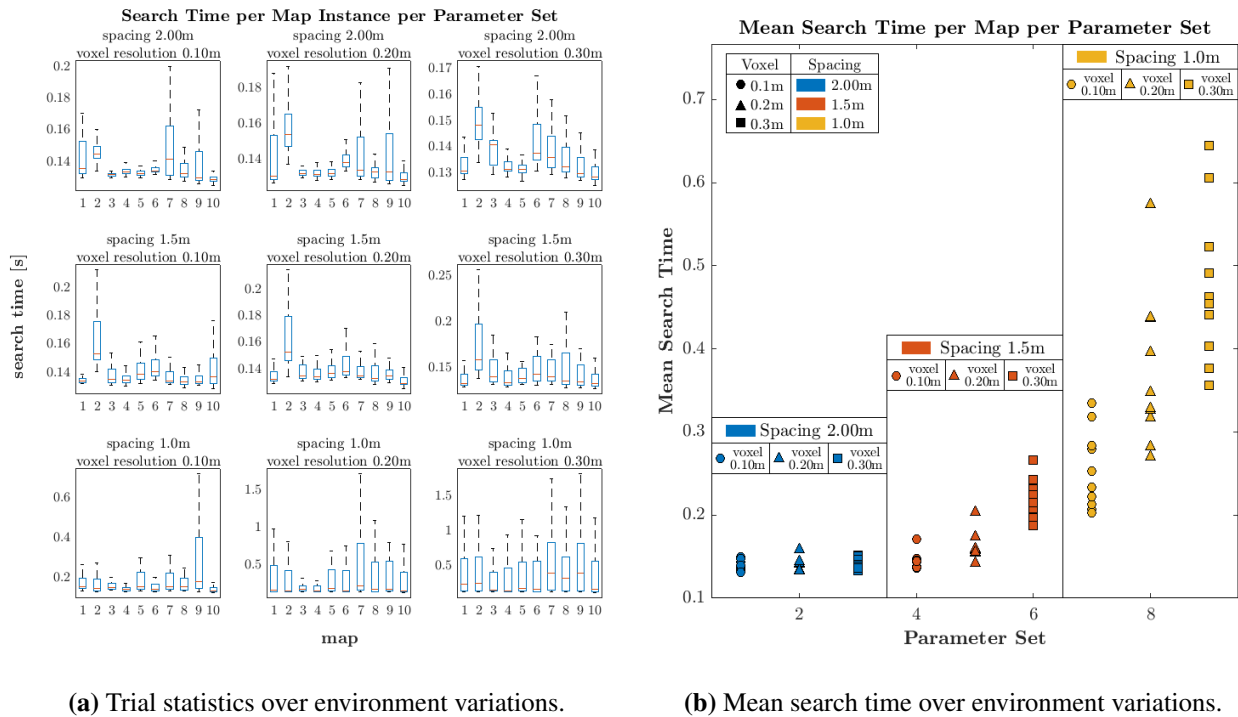


Figure 6.3.6: Fig. 6.3.6a shows the statistical distribution of trial results for all environments at each parameter combination. That search performs similarly across environments generated with the same parameters is more clearly shown in Fig. 6.3.6b, where the mean search times across all trials, per map, are shown on the same axes. Search times clearly cluster based on obstacle spacing and voxel resolution.

within a ball with a radius of 1m, $\mathcal{B}_r = 1\text{m}$. Therefore, we test voxel resolutions ranging from 0.1m to 0.3m, $v_{\text{res}} = [r, 3r]$, and obstacle spacings ranging from 1m to 2m, $d_{\text{obs}}^{\text{ref}} = [\mathcal{B}_r, 2\mathcal{B}_r]$. The group is then directed to follow randomly placed motion primitives which are collision checked against the environment using a small offset. An example is shown in Fig. 6.3.5a.

We generate ten random environments for every specified obstacle spacing parameter, $d_{\text{obs}}^{\text{ref}}$. Each environment is then represented at every voxel resolution, v_{res} . We run one hundred trials over every environment. The selected paths (the randomly placed and chosen motion primitives) are kept consistent across environments of different voxel resolutions. I.e., for Map A generated with obstacles spaced at $d_{\text{obs}}^{\text{ref}} = 1\text{m}$, the set of paths which we ask the robots to navigate is consistent across Map A represented at voxel resolutions $v_{\text{res}} = \{0.1\text{m}, 0.2\text{m}, 0.3\text{m}\}$. This allows us to see how search time scales for the same desired paths at different voxel resolutions.

The statistical distributions of all trials for each map are shown in Fig. 6.3.6a. Figure 6.3.6b, however, puts search performance across environments in perspective by plotting the mean search time across trials for each environment on the same axes. Figure 6.3.6b shows that search results clearly cluster based on voxel resolution and obstacle spacing. We can understand this as voxel resolution and obstacles spacing effectively representing the “difficulty” of an environment, and as we would expect, search time increases as free space decreases—whether free space is truly decreasing, as in cases where obstacles are spaced closer together, or in cases where obstacles are conservatively represented as with coarser voxelizations. As in 5.6, we can see that we do not need to perform a tradeoff with environment fidelity in an attempt to reduce collision checking time; rather, it is better to operate with voxel resolutions that allow robots to take advantage of available

free space.

6.3.8 Method comparison

Fig. 6.3.7 shows a scenario environment, where a group of quadrotors is driven by an operator through a cluttered obstacle field. While the view is shown top down, the environment is modeled in 3D. The desired group flight path, as directed by an operator selecting motion primitives via joystick, is shown in blue. We use this scenario to evaluate the performance of the proposed method in comparison to existing multi-robot motion planning methods.

The method of [90, 91] enables a human user to teleoperate formations of quadrotors online using joystick control and is therefore the most direct comparison work to our application. In this work, multiple vector fields are used to control (1) quadrotor deviation from a formation specification, (2) inter-quadrotor collision avoidance, and (3) quadrotor collision avoidance with respect to obstacles. The advantage to designing vector fields is that controls are effectively precomputed across an environment, making online execution very fast, and the method was demonstrated to work across scaling numbers of quadrotors as well as in the presence of low numbers of obstacles. However, the vector field methodology of [90, 91] does not perform well in the highly cluttered environments we consider in our application. It is highly difficult to balance parameters across formation, inter-robot, and robot-obstacle fields: if repulsion parameters are too strong, robots remain safe but are unable to enter narrow bottle necks or confined areas, and cannot effectively reach the goal. If repulsion parameters are lowered, robots risk collision with environment features.

The method of [5] presents an optimization based approach for controlling a formation of

	method success [%]	average search time	average $\Delta r_{\min}^{\text{robot}}$	average $\Delta r_{\min}^{\text{obstacle}}$	average v_{\max}	average a_{\max}	average j_{\max}	average d_{\max}	average \dot{E}_{\max}	average t_{\max}
Formation Velocity Fields [91]	0.00	1.55	0.15	0.00	0.73	2.10	21.66	1.14	12.22	5.99
Formation Optimization [5]	80.00	5.14	0.14	0.05	0.82	2.44	24.52	1.15	21.71	5.40
Data Leveraged Action Search (ours)	100.00	0.62	0.14	0.13	0.71	2.28	20.32	1.83	26.50	6.30

Table 6.3.3: This table reports the average values of metrics defined in Table 6.3.1 for two comparison formation-planning approaches and our proposed approach for the example scenario shown in Fig. 6.3.7. (When not specified, metric units are given in Table 6.3.1.) In this example, small robots with a radius of 0.05m were used.

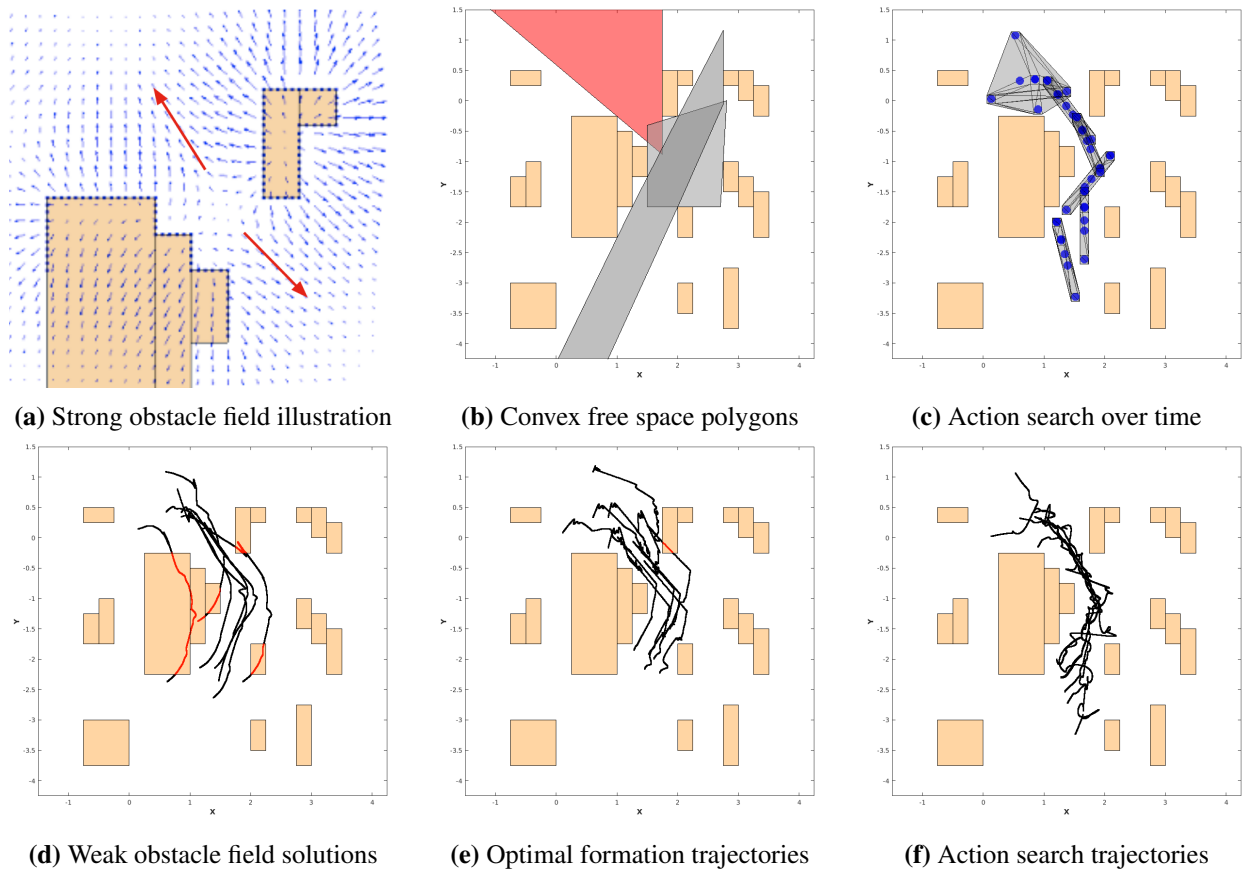


Figure 6.3.7: Approach comparisons for a group of seven robots with column-based obstacles to allow easy visual illustration; obstacles extend in the z -direction. **Subfigures are arranged vertically per method.** Figs. (a) and (d) (left column) show respective examples of the vector fields approach, where strong obstacle fields preclude passage through a gap while weak obstacle fields do not avoid robot-obstacle collisions (collisions are shown in red, Fig. (d)). Figs. (b) and (e) (middle column) show the formation optimization approach. Fig. (b) shows selected convex regions of free space used to constrain the optimization. These regions can be too constrained to allow a formation to fit between obstacles, as is the case for the red polytope. In Fig. (e), a straight line (colliding, shown in red) trajectory was carried out to move the formation across the gap to continue evaluating the approach along the desired path. Figs. (c) and (f) (right column) show our action-based search approach. While the trajectories are not as smooth as those of the comparison approaches, the method performs the fastest and avoids all obstacles and inter-robot collisions.

robots online. This approach adapts parameters governing formation scale, rotation, and translation (rather than optimizing parameters on an individual robot basis), and performs optimization calculations in a receding horizon formulation along the desired path. These concepts of reducing optimization parameters by controlling a group-based representation and operating over a local horizon enable the methodology to work within the within online time constraints for the application of [5] and parallel our approach's decisions to operate on group-level actions over a local planning horizon. The methodology of [5] was also able to operate in environments with obstacles, by first calculating a convex representation of free space over the local horizon and then using this convex bounding volume to provide linear constraints to the optimization calculation to ensure that the robot formation remains within the calculated free space. This receding horizon approach fits well with a joystick selected, motion-primitive based input model, but free space calculation can be challenging in highly cluttered environments and the tradeoff between calculation time and horizon length mean that solution fidelity is poor given similar calculation times to our method.

We evaluate the vector field based methodology of [90, 91], the optimization approach of [5], and our own data-based action search for a group of seven robots for two scenarios. In Fig. 6.3.7 and corresponding table Tab. 6.3.3, we show a short example where robots follow five user-selected motion primitives through a 2.5D environment, where obstacles extend as columns in the z -dimension. This scenario is useful because it clearly shows failure modes of the competing methods and allows easy top-down illustration of the results. In Figs. 6.3.8-6.3.9 and corresponding table Tab. 6.3.4, we show a longer example where robots follow 25 user-selected motion primitives through a full 3D environment. This example shows that our data-based action search approach

	method success [%]	average search time	average $\Delta r_{\min}^{\text{robot}}$	average $\Delta r_{\min}^{\text{obstacle}}$	average v_{\max}	average a_{\max}	average j_{\max}	average d_{\max}	average \dot{E}_{\max}	average t_{\max}
Formation Velocity Fields [91]	48.00	0.85	0.23	0.06	0.42	1.95	33.71	0.78	14.07	3.80
Formation Optimization [5]	100.00	6.17	0.14	0.08	0.46	1.82	26.80	0.80	15.17	4.01
Data Leveraged Action Search (ours)	100.00	0.44	0.13	0.18	0.93	2.64	22.89	2.11	26.10	6.53

Table 6.3.4: This table reports the average values of metrics defined in Table 6.3.1 for two comparison formation-planning approaches and our proposed approach for the example scenario shown in Fig. 6.3.7. (When not specified, metric units are given in Table 6.3.1.) In this example, small robots with a radius of 0.05m were used.

fully works in 3D and outperforms comparison approaches in this cluttered 3D scenario.

We note that in order to give a best effort comparison between methods and avoid collisions during evaluation, it was necessary for a user to select different motion primitives in order to direct the robot group when using the formation optimization approach [5] (shown in Fig. 6.3.8). (We were not able to find parameters that avoided all collisions in the vector fields approach.) While the action search method keeps the center of the robot group on the user-selected path, the optimization approach allows the robot formation to translate within the local free-space polytope. Under this approach, the robot group therefore frequently strays from where the user has directed the group to go, necessitating the user select different primitives to guide the group back to the desired direction.

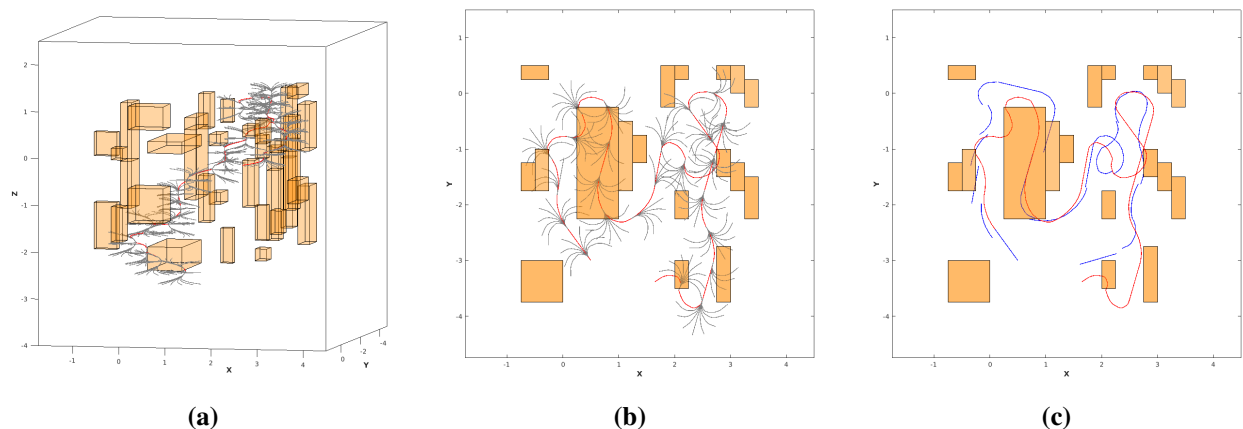


Figure 6.3.8: User-selected motion primitives traversing x , y , and z in a 3D environment. Fig. (a) shows the primitives in 3D from an oblique angle; Fig. (b) shows the same primitives using a top-down view; Fig. (c) shows that different primitives were required between the action search experiment (shown in red) and the formation optimization experiment (shown in blue) in order to navigate the groups through the environment without obstacle collision.

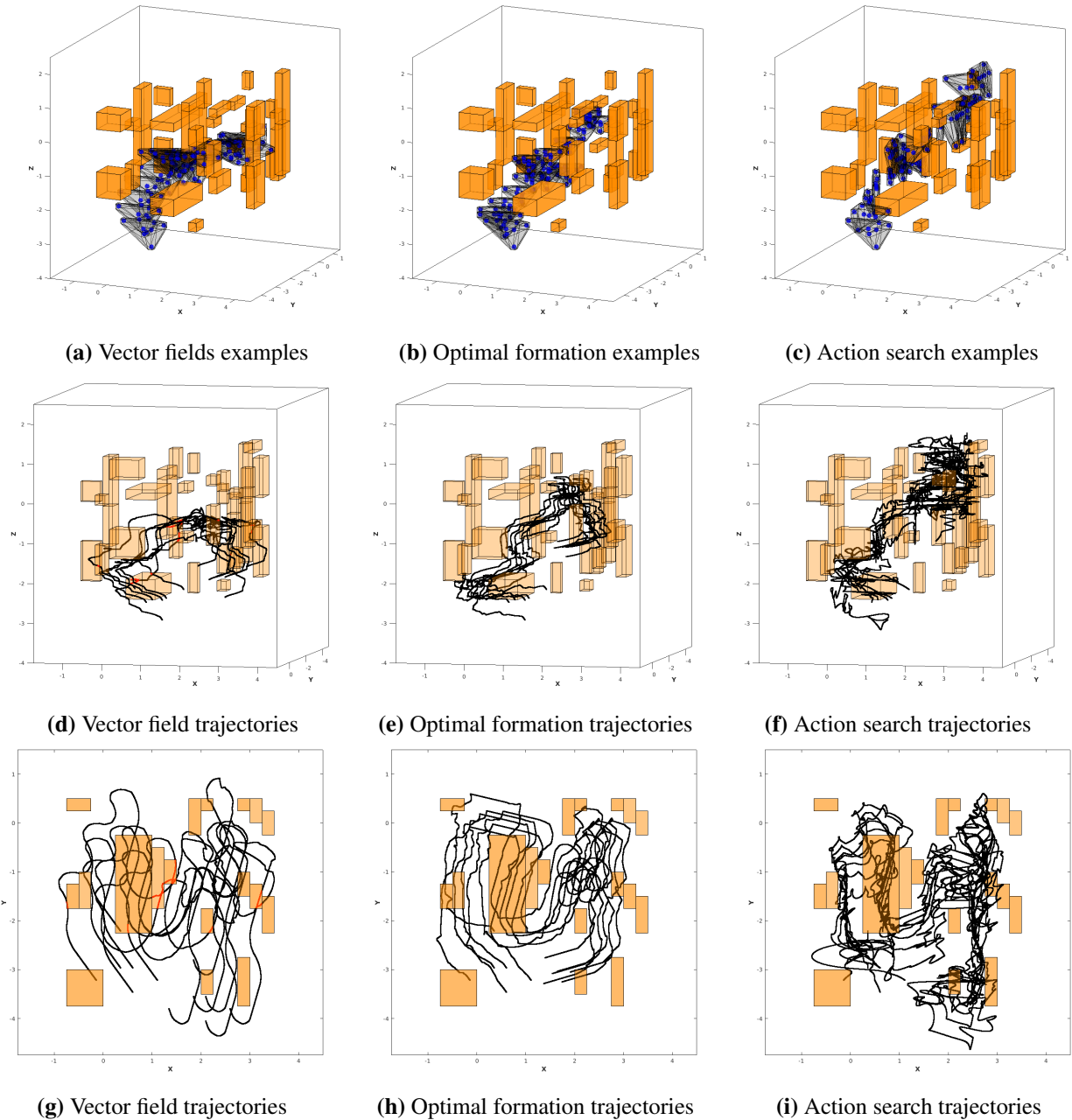


Figure 6.3.9: Approach comparisons for a group of seven robots following directed motion primitives traversing x , y , and z in a 3D environment. **Subfigures are arranged vertically per method.** The top row, Figs. (a)-(c), show snapshots of the robot for each method, respectively, at various points along the directed path. The middle row, Figs. (d)-(f), show all robot trajectories for each respective method from an oblique viewing angle; the bottom row, Figs. (g)-(i) show the same trajectories from a top-down perspective. Any trajectory portions in red denote collisions.

At the end of following a selected path, motion primitives are therefore re-applied from the center of the group. This is why in Fig. 6.3.8c the blue motion primitives are not always connected to

their preceding selected primitives; the group ended with the group center some distance away from the desired goal location. This is in contrast to the red motion primitives selected during our action search approach.

As in the 2.5D scenario (Tab. 6.3.3), the table of results from this experiment (Tab. 6.3.4) shows that our method is faster than the comparison methods and collision free.

6.4 Concluding Remarks

Plan computation in state of the art methods for multi-robot coordination, such as the vector field and formation optimization approaches examined in this chapter, depend on obstacle representation. Robot plans are calculated based on static fields surrounding obstacles in [90, 91], while [5] limits plans to remain within convex calculations of free space between obstacles. While considering obstacles in these methods works well in sparsely cluttered environments, these representations limit solution space in cluttered environments.

In contrast, our approach does not make assumptions about obstacles or available free space before computing plans. Our approach minimizes the loss of potential solution space by directly evaluating action options against the environment, and this enables us to operate in the highly cluttered and unstructured environments we anticipate in reconnaissance applications at an improved solution rate over state of art methods.

However, incorporating environment features into search policies is a significant opportunity for future work. We anticipate that being able to more intelligently search for non-colliding actions during planning would greatly improve search speed and solution quality. Even without

considering action selection based on local obstacles or environment features however, this section shows that the proposed approach enables faster operation with higher success rates than current approaches in previously untenable environments.

Chapter 7

Conclusion

This thesis addresses the problem of planning coordinated motions online for multi-robot teams in response to human operator input in two representative applications: (1) a theatrical production and (2) a reconnaissance scenario. These complex applications highlight the requirements and challenges associated with the online multi-robot planning problem. Across all applications, the objective is to translate user provided input into dynamically feasible and collision free trajectories for all robots in the team. The challenges associated with this objective broadly cover interpreting operator intent, mapping user input to feasible and safe trajectories for every agent of the team, and operating within online constraints. Specifically, all plans must account for the state of the system at the time the command is given within budgeted time and computation limits. When addressing application specific requirements, we examine constraints imposed by both the need for highly detailed user input, as requested by group behaviors in the theatric application, and under-defined user input scenarios, as when operating in response to joystick-selected motion primitives. To enable high-speed operation in the reconnaissance application, we additionally consider the

challenge of planning multi-robot motions considering highly cluttered environments.

7.1 Contribution Summary

In this thesis, we address these challenges through the development of a descriptor-based parameterization approach for users to specify multi-robot behaviors in the context of a theatrial application. This system contributes a methodology for the online specification of detailed multi-robot behaviors, and enables the performance of human-directed theatrial productions for multiple quadrotors (Chapter 3).

We contribute a formalization of multi-robot behaviors as action sequences that enables prior experience (Chapter 4) and offline data (Chapter 5) to be incorporated into a search framework for online action selection. The online composition of actions via a search approach allows for operation in environments with obstacles. We demonstrate that the generalized representation of multi-robot actions coupled with a data-driven search heuristic recreates the specialized multi-robot behaviors used in the theatrial application of Chapter 3. This illustrates that the descriptor-based input system is a subset of the generalized behavior formulation. Chapter 6 contributes characterization analysis of the approach, with comparison to existing state-of-the-art methods with respect to the specific example of a reconnaissance application.

7.2 Opportunities for Future Work

Our approach proposes a group-based representation for multi-robot actions and searches over a discrete set of these actions to create a feasible action sequence over a planning horizon. While the affine parameterization and discretization decisions constrain the space of multi-robot actions, searching over all possible actions is intractable given online time constraints. Our contributions show that the challenges of a large search space can be mitigated by leveraging prior data. This contributed methodology provides a framework that enables future lines of research to further multi-robot planning capabilities, based upon the ability of the method to incorporate data to bias solutions to reflect desired attributes. Three methods of influencing solution choice might include,

- direct evaluation of desired solution metrics during action selection;
- evaluating heuristics to reflect action value with respect to the environment;
- incorporating adaptive weighting to reflect changing user intent;

and are discussed below.

Incorporating quality metrics into search With respect to leveraging prior data, we evaluated a simple probability-based heuristic given action-sequence frequency. This heuristic biases solutions towards replicating demonstrated action sequences, and so solutions implicitly reflect attributes of the dataset. In Chapter 5, we showed this in the context of replicating theatric behaviors generated via the methodology of Chapter 3, demonstrating that motions performed via biased action search replicated the rotations and shape changes from provided demonstrations. However, to explicitly achieve solutions with desired qualities such as minimal travel time, energy, or jerk, future work

may examine approaches for incorporating an evaluation of such solution qualities during search selection.

Context-based action selection A probability-based heuristic was chosen as a reasonable metric upon which to inform online search, showing that offline data could be incorporated to yield an improved search response. However, an important line of future research is to examine alternate search heuristics. By incorporating context into search—the relationship of actions with respect to both environment and user intent—we can expect to improve action selection, leading to more quickly finding higher quality solutions.

Selection heuristics based on relevant features have shown improved success in large search domains such as [12, 75]. These methods approach search in the context of games such as Chess and Go, which have a large but defined state space in terms of board state. Chosen features, or the entire board state itself, can be valued for evaluating action selection. For example, the approach of [75] takes board state as input to a neural net and learns the value of a board state, i.e. trains the neural net serving as the selection policy, over thousands of game plays.

Unlike in game play, in motion planning it is potentially infeasible to use the entire “board state,” or complete environment-agent(s) representation, as features. This is due to first, the difficulty of enumerating such a space, and second, collecting sufficient information about such a large representation to sufficiently guide search. Future work remains to incorporate environment context into search, potentially via biasing action selection based on local environment features. Informing a prediction of how actions may lead to collision with local obstacle geometry will result in expanding fewer colliding actions during search, leading to more rapid exploration through the

search tree. Therefore, determining relevant environment features is a beneficial avenue of future research to speed online computation, and similar search-based approaches for single-robot motion planning showed a large improvement in search time when learning heuristic search weights on designer-specified features [10].

While incorporating environment context may prove highly valuable for faster search through intelligent collision avoidance, we also consider the quality of solutions with respect to user intent. Operator intent may change during use; for example, a user may want to perform a series of tasks, and so intent with respect to preferred actions could shift between tasks. Other examples include cases where a user may change their intent based on learned information during operation, or in response to changing mission operatives. The provided methodology shows action selection based on prior data, but does not provide a means to change action biases during the course of operation. A future line of research might evaluate strategies for incorporating operational context into action selection, such as through a switching model over different action distributions or through an adaptive weighting mechanism. Enabling search to incorporate specific models of user intent may facilitate system-user interaction and improve the overall efficacy of the approach in finding application or task specific inter-robot coordination policies during online operation.

Context inference Enabling context-based solution generation can be thought of as a forward, context-driven approach. However, the framework discussed in this thesis also provides a model for thinking about context recognition. In Chapter 4, we discussed learning selection weights for search trees. A search tree with learned weights therefore acts as a record of, or model for, an environment considered in conjunction with user intent.

The capability to recognize the context under which the system is operating could provide the feedback mechanism to move towards robust autonomy. For example, we envision an operating scenario where a planning system is able to discern that a given distribution of actions is not providing good solutions, and based on that performance, recognize what context—and by extension, associated action distribution—might be better applicable.

The abilities to recognize what actions might be best employed and to adapt when a plan fails are currently provided by human operators, and the need for these capabilities is a motivating factor in requiring human direction of multi-robot systems. Providing a robotic system with similar abilities can allow a system to better learn from and assist an operator, yielding a more efficient and capable system. The methodology provided by this thesis enables us to better interact with and employ multi-robot systems for present use, and provides a foundation for further development to assist human–multi-robot interaction and move towards autonomous planning for multi-robot systems in real world applications.

Bibliography

- [1] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Roland Siegwart, and Paul Beardsley. Multi-robot system for artistic pattern formation. In *2011 IEEE International Conference on Robotics and Automation*, pages 4512–4517. IEEE, 2011. [Cited in Sect. 40.]
- [2] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems*, pages 203–216. Springer, 2013. [Cited in Sects. 9 and 88.]
- [3] Javier Alonso-Mora, Stefan Haegeli Lohaus, Philipp Leemann, Roland Siegwart, and Paul Beardsley. Gesture based human–multi-robot swarm interaction and its application to an interactive display. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 5948–5953, May 2015. [Cited in Sect. 8.]
- [4] Javier Alonso-Mora, Tobias Naegeli, Roland Siegwart, and Paul Beardsley. Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39(1):101–121, 2015. [Cited in Sects. 9 and 88.]
- [5] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and ob-

- ject transport in dynamic environments via constrained optimization. *Intl. Journal of Robot. Research*, 36(9):1000–1021, 2017. [Cited in Sects. 1, 9, 10, 40, 41, 43, 62, 78, 85, 86, 88, 113, 114, 115, 116, and 118.]
- [6] Christopher Amato, George Konidaris, Ariel Anders, Gabriel Cruz, Jonathan P How, and Leslie P Kaelbling. Policy search for multi-robot coordination under uncertainty. *The International Journal of Robotics Research*, 35(14):1760–1778, 2016. [Cited in Sect. 13.]
- [7] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. The entrapment/escorting mission. *IEEE Robotics & Automation Magazine*, 15(1):22–29, 2008. [Cited in Sects. 1 and 40.]
- [8] Defence Science & Technology Organisation (DSTO) Australia and Research Development & Engineering Command (RDECOM) USA. Multi autonomous ground-robotic international challenge (magic 2010), November 2010. URL <https://web.archive.org/web/20101120203708/http://www.dsto.defence.gov.au/magic2010/>. [Accessed 11 August 2020]. [Cited in Sects. 1 and 2.]
- [9] Shishir Bashyal and Ganesh Kumar Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *IEEE Swarm Intelligence Symposium*, pages 1–8. IEEE, 2008. [Cited in Sect. 8.]
- [10] Mohak Bhardwaj, Sanjiban Choudhury, and Sebastian Scherer. Learning heuristic search via imitation. *arXiv preprint arXiv:1707.03034*, 2017. [Cited in Sect. 125.]
- [11] Daniel S. Brown, Sean C. Kerman, and Michael A. Goodrich. Human-swarm interactions based on managing attractors. In *Proc. of the ACM/IEEE Intl. Conf. on Human-Robot Inter-*

- action*, pages 90–97, Bielefeld, Germany, 2014. [Cited in Sect. 8.]
- [12] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. on Computational Intell. and AI in Games*, 4(1):1–43, 2012. [Cited in Sects. 45, 52, 53, and 124.]
- [13] Alain Caltieri and Francesco Amigoni. High-level commands in human-robot interaction for search and rescue. In *Robot Soccer World Cup*, pages 480–491. Springer, 2013. [Cited in Sect. 2.]
- [14] Michal Čáp, Peter Novák, Jiří Vokřínek, and Michal Pěchouček. Multi-agent rrt*: Sampling-based cooperative pathfinding. *arXiv preprint arXiv:1302.2828*, 2013. [Cited in Sects. 11, 12, and 87.]
- [15] Ellen A. Cappel, Arjav Desai, and Nathan Michael. Robust coordinated aerial deployments for theatrical applications given online user interaction via behavior composition. *Int. Symp. on Dist. Auton. Robotics Syst.*, November 2016. [Cited in Sects. 2, 40, 47, 48, 59, 64, 68, 97, and 102.]
- [16] Ellen A. Cappel, Arjav Desai, Matthew Collins, and Nathan Michael. Online planning for human – multi-robot interactive theatrical performance. *Auton. Robots*, 42:1771–1786, December 2018. doi: 10.1007/s10514-018-9755-0. [Cited in Sects. 40, 47, 48, 59, 64, 68, 97, and 102.]
- [17] Ellen A. Cappel, Arjav Desai, and Nathan Michael. Data driven online multi-robot formation planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Las Vegas, NV,

- USA, October 2020. [Cited in Sect. 64.]
- [18] A. Chammesdine. Flatness based trajectory planning/replanning for a quadrotor aerial vehicle. *IEEE Trans. Aerosp. Electron. Syst.*, pages 2832 – 2848, 2012. [Cited in Sects. 10 and 85.]
- [19] Abbas Chamseddine, Youmin Zhang, Camille Alain Rabbath, Cedric Join, and Didier Theil-liol. Flatness based trajectory planning/replanning for a quadrotor aerial vehicle. In *IEEE Trans. Aerosp. Electron. Syst.*, pages 2832 – 2848, 2012. [Cited in Sects. 20, 28, and 30.]
- [20] Yufan Chen, Mark Cutler, and Jonathan P How. Decoupled multiagent path planning via incremental sequential convex programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961. IEEE, 2015. [Cited in Sects. 10, 85, and 86.]
- [21] Ke Cheng, Yi Wang, and Prithviraj Dasgupta. Distributed area coverage using robot flocks. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 678–683. IEEE, 2009. [Cited in Sects. 1 and 40.]
- [22] Moo K. Chung. *Statistical and computational methods in brain image analysis*. CRC press, 2013. [Cited in Sect. 72.]
- [23] Edmund M. Clarke and Jeannette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996. [Cited in Sects. 32 and 33.]
- [24] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511. [

Cited in Sect. 31.]

- [25] Prithviraj Dasgupta, Taylor Whipple, and Ke Cheng. Effects of multi-robot team formations on distributed area coverage. *International Journal of Swarm Intelligence Research (IJSIR)*, 2(1):44–69, 2011. [Cited in Sects. 1 and 40.]
- [26] Jonathan A. DeCastro, Javier Alonso-Mora, Vasumathi Raman, Daniela Rus, and Hadas Kress-Gazit. Collision-free reactive mission and motion planning for multi-robot systems. In *Robotics Research*, pages 459–476. Springer, 2018. [Cited in Sect. 32.]
- [27] Arjav Desai, Ellen A. Cappel, and Nathan Michael. Dynamically feasible and safe shape transitions for teams of aerial robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 5489–5494, Daejeon, Korea, October 2016. doi: 10.1109/IROS.2016.7759807. [Cited in Sects. 20, 22, and 28.]
- [28] Arjav Desai, Matthew Collins, and Nathan Michael. Efficient kinodynamic multi-robot replanning in known workspaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1021–1027. IEEE, 2019. [Cited in Sects. 11, 12, and 86.]
- [29] David W. Eggert, Adele Lorusso, and Robert B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5-6): 272–290, 1997. [Cited in Sect. 70.]
- [30] Ehang. Chinese new year 2017 lantern festival, February 2017. URL <http://www.ehang.com/cn/news/228.html>. [Accessed 15 February 2017]. [Cited in Sect. 1.]
- [31] Edgar Galván-López, Ruohua Li, Constantinos Patsakis, Siobhán Clarke, and Vinny Cahill. Heuristic-based multi-agent monte carlo tree search. In *IISA 2014, The 5th International*

- Conference on Information, Intelligence, Systems and Applications*, pages 177–182. IEEE, 2014. [Cited in Sects. 11, 12, and 87.]
- [32] Kristin Giammarco and Kathleen Giles. Verification and validation of behavior models using lightweight formal methods. In *Disciplinary Convergence in Systems Engineering Research*, pages 431–447. Springer, 2018. [Cited in Sects. 32 and 33.]
- [33] Pooya Deldar Gohardani, Siavash Mehrabi, and Peyman Ardestani. RoboCup rescue simulation system 2016 champion team paper. In *Robot World Cup*, pages 565–576. Springer, 2016. [Cited in Sect. 2.]
- [34] John C. Gower and Garmt B. Dijkstra. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004. [Cited in Sects. 68, 69, and 72.]
- [35] Sandro Hauri, Javier Alonso-Mora, Andreas Breitenmoser, Roland Siegwart, and Paul Beardley. Multi-robot formation control via a real-time drawing interface. In *Field and Service Robotics*, pages 175–189. Springer, 2014. [Cited in Sect. 8.]
- [36] Liang He, Jia Pan, Wenping Wang, and Dinesh Manocha. Proxemic group behaviors using reciprocal multi-agent navigation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2016. [Cited in Sects. 9 and 87.]
- [37] Markus Hehn and Raffaello D’Andrea. Quadcopter trajectory generation and control. In *IFAC World Congress*, pages 1485–1491, 2011. [Cited in Sects. 19 and 28.]
- [38] A Hsieh and S Lacroix. Special issue: Special issue on multi autonomous ground-robotic international challenge (magic), volume 29 of. *Journal of Field Robotics*, 2012. [Cited in Sect. 1.]

- [39] Intel Corp. Intel drone light show breaks guinness world records title at olympic winter games pyeongchang 2018, February 2018. URL <https://newsroom.intel.com/news-releases/intel-drone-light-show-breaks-guinness-world-records-title-olympic-winter-games-pyeongchang-2018/>. [Accessed 11 August 2020]. [Cited in Sects. 1 and 2.]
- [40] Intel Corp., Lady Gaga, Pepsi, and National Football League. Pepsi zero sugar super bowl li halftime show, February 2017. URL <http://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html>. [Accessed 15 February 2017]. [Cited in Sect. 1.]
- [41] Jinmingwu Jiang and Kaigui Wu. Cooperative pathfinding based on high-scalability multi-agent rrt. *arXiv preprint arXiv:1911.07840*, 2019. [Cited in Sect. 12.]
- [42] Hiroaki Kitano and Satoshi Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI magazine*, 22(1):39–39, 2001. [Cited in Sect. 2.]
- [43] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006. [Cited in Sect. 53.]
- [44] Andreas Kolling, Steven Nunnally, and Michael Lewis. Towards human control of robot swarms. In *Proc. of the ACM/IEEE Intl. Conf. on Human-Robot Interaction*, pages 89–96. ACM, 2012. [Cited in Sect. 8.]
- [45] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. Human interaction with robot swarms: A survey. *IEEE Trans. on Human-Machine Syst.*, 46(1):9–26, February 2016. [Cited in Sects. 8, 9, and 87.]

- [46] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, (0), 2018. [Cited in Sect. 32.]
- [47] H.W. Kuhn. The hungarian method for assignment problem. In *Naval Research Logistics Quarterly*, pages 83–97, 2012. [Cited in Sect. 30.]
- [48] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1995–2003. JMLR. org, 2017. [Cited in Sect. 63.]
- [49] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *International Conference on Runtime Verification*, pages 122–135. Springer, 2010. [Cited in Sect. 33.]
- [50] Naomi E. Leonard, Derek A. Paley, Russ E. Davis, David M. Fratantoni, Francois Lekien, and Fumin Zhang. Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay. *J. Field Robot.*, 27(6):718–740, 2010. [Cited in Sect. 8.]
- [51] Qiyang Li, Xintong Du, Yizhou Huang, Quinlan Sykora, and Angela P Schoellig. Learning of coordination policies for robotic swarms. *arXiv preprint arXiv:1709.06620*, 2017. [Cited in Sect. 63.]
- [52] Miao Liu, Kavinayan Sivakumar, Shayegan Omidshafiei, Christopher Amato, and Jonathan P How. Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. In *2017 IEEE/RSJ International Conference on Intelligent Robots and*

- Systems (IROS)*, pages 1853–1860. IEEE, 2017. [Cited in Sect. 13.]
- [53] Yuanchang Liu and Richard Bucknall. A survey of formation control and motion planning of multiple unmanned vehicles. *Robotica*, 36(7):1019–1047, 2018. [Cited in Sect. 40.]
- [54] MagicLab and Rhizomatiks Research. MagicLab 24 drone flight, February 2016. URL http://www.magiclab.nyc/research/drone_magic/. [Accessed 14 June 2016]. [Cited in Sect. 2.]
- [55] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robot. Autom. Mag.*, 19(3):20–32, 2012. [Cited in Sect. 25.]
- [56] Ali Mehri, Maryam Jamaati, and Hassan Mehri. Word ranking in a single document by jensen–shannon divergence. *Physics Letters A*, 379(28-29):1627–1632, 2015. [Cited in Sects. 77 and 98.]
- [57] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE, 2011. [Cited in Sects. 10, 86, and 88.]
- [58] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 477–483, Saint Paul, MN, May 2012. [Cited in Sects. 10, 26, 85, and 86.]
- [59] Máté Nagy, Zsuzsa Ákos, Dora Biro, and Tamás Vicsek. Hierarchical group dynamics in pigeon flocks. *Nature*, 464(7290):890–893, 2010. [Cited in Sects. 97 and 102.]

- [60] Máté Nagy, Gábor Vásárhelyi, Benjamin Pettit, Isabella Roberts-Mariani, Tamás Vicsek, and Dora Biro. Context-dependent hierarchies in pigeons. *Proceedings of the National Academy of Sciences*, 110(32):13049–13054, 2013. [Cited in Sects. 97 and 102.]
- [61] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015. [Cited in Sect. 40.]
- [62] Estefanía Pereyra, Gastón Araguás, and Miroslav Kulich. Path planning for a formation of mobile robots with split and merge. In *International Workshop on Modelling and Simulation for Autonomous Systems*, pages 59–71. Springer, 2017. [Cited in Sect. 13.]
- [63] Shokoofeh Pourmehr, Valiallah Mani Monajjemi, Richard Vaughan, and Greg Mori. “You two! Take off!”: Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 137–142. IEEE, 2013. [Cited in Sect. 8.]
- [64] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. CrazySwarm: A large nano-quadcopter swarm. *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3299–3304, 2017. [Cited in Sect. 35.]
- [65] Pradeep Ranganathan, Ryan Morton, Andrew Richardson, Johannes Strom, Robert Goeddel, Mihai Bulic, and Edwin Olson. Coordinating a team of robots for urban reconnaissance. In *Proceedings of the Land Warfare Conference (LWC)*, 2010. [Cited in Sect. 2.]
- [66] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016. doi: 10.1007/978-3-319-28872-7_37. [Cited in Sects. 10, 19, 23, 26, 28,

78, 86, and 88.]

- [67] D Reed Robinson, Robert T Mar, Katia Estabridis, and Gary Hewer. An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments. *IEEE Robotics and Automation Letters*, 3(2):1215–1222, 2018. [Cited in Sects. 10 and 85.]
- [68] RoboCup. Robocup rescue simulation league. URL <https://rescuesim.robocup.org/>. [Accessed 11 August 2020]. [Cited in Sect. 2.]
- [69] Stuart J. Russell and Peter Norvig. *Artificial intelligence: A modern approach*. Malaysia; Pearson Education Limited, 2016. [Cited in Sects. 73, 74, 88, and 92.]
- [70] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1525–1532. IEEE, 2014. [Cited in Sects. 32 and 33.]
- [71] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. Implan: Scalable incremental motion planning for multi-robot systems. In *ACM/IEEE 7th Intl. Conf. on Cyber-Physical Syst.*, pages 1–10. IEEE, 2016. [Cited in Sects. 32 and 33.]
- [72] Cristian Secchi, Lorenzo Sabattini, and Cesare Fantuzzi. Conducting multi-robot systems: Gestures for the passive teleoperation of multiple slaves. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 2803–2808. IEEE, 2015. [Cited in Sect. 8.]
- [73] Tina Setter, Alex Fouraker, Hiroaki Kawashima, and Magnus Egerstedt. Haptic interactions with multi-robot swarms using manipulability. *Journal of Human-Robot Interaction*, 4(1):

60–74, 2015. [Cited in Sect. 8.]

- [74] Raymond Sheh, Sören Schwertfeger, and Arnoud Visser. 16 years of robocup rescue. *KI-Künstliche Intelligenz*, 30(3-4):267–277, 2016. [Cited in Sect. 2.]
- [75] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. [Cited in Sects. 45, 52, and 124.]
- [76] Adrian Stoica, Theodoros Theodoridis, Huosheng Hu, Klaus McDonald-Maier, and David F. Barrero. Towards human-friendly efficient control of multi-robot teams. In *Collaboration Technologies and Sys., Int. Conf. on*, pages 226–231. IEEE, 2013. [Cited in Sect. 8.]
- [77] Siddharth Swaminathan, Mike Phillips, and Maxim Likhachev. Planning for multi-agent teams with leader switching. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5403–5410. IEEE, 2015. [Cited in Sects. 11, 13, and 86.]
- [78] Matthew Turpin, Nathan Michael, and Vijay Kumar. Decentralized formation control with variable shapes for aerial robots. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 23–30. IEEE, 2012. [Cited in Sects. 47 and 62.]
- [79] Matthew Turpin, Nathan Michael, and Vijay Kumar. Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2013. [Cited in Sect. 19.]
- [80] Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of aerial robots. In *Proc. of Robot.: Sci. and Syst.*, 2013.

[Cited in Sects. 19, 29, 30, and 31.]

- [81] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014. [Cited in Sects. 11, 12, 87, and 96.]
- [82] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2008. [Cited in Sects. 9 and 87.]
- [83] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer, 2011. doi: 10.1007/978-3-642-19457-3_1. [Cited in Sects. 9 and 87.]
- [84] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20):eaat3536, 2018. [Cited in Sects. 97 and 102.]
- [85] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 3260–3267. IEEE, 2011. [Cited in Sects. 11, 12, 86, and 87.]
- [86] Takahiro Yamada. Principles of error detection and correction. In *Essentials of Error-Control Coding Techniques*, pages 11–37. Elsevier, 1990. [Cited in Sect. 76.]
- [87] Xuning Yang, Koushil Sreenath, and Nathan Michael. A framework for efficient teleoperation via online adaptation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 5948–5953. IEEE, 2017. [Cited in Sects. 14, 85, 88, and 95.]

- [88] Xuning Yang, Ayush Agrawal, Koushil Sreenath, and Nathan Michael. Online adaptive teleoperation via motion primitives for mobile robots. *Auton. Robots*, pages 1–17, 2018. [Cited in Sects. 14, 85, 88, and 95.]
- [89] Nicholas Zerbek and Logan Yliniemi. Multiagent monte carlo tree search. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2309–2311. International Foundation for Autonomous Agents and Multiagent Systems, 2019. [Cited in Sects. 11, 12, and 87.]
- [90] Dingjiang Zhou and Mac Schwager. Assistive collision avoidance for quadrotor swarm teleoperation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1249–1254, Stockholm, Sweden, May 2016. [Cited in Sects. 8, 113, 115, and 118.]
- [91] Dingjiang Zhou, Zijian Wang, and Mac Schwager. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Trans. Robot.*, 34(4):916–923, 2018. [Cited in Sects. 47, 62, 85, 88, 113, 114, 115, 116, and 118.]