Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 13 March 2024
DOI 10.3389/frobt.2024.1172105

Cooperative planning for physically interacting heterogeneous robots

Michael A. Sebok*[†] and Herbert G. Tanner

Department of Mechanical Engineering, University of Delaware, Newark, DE, United States

OPEN ACCESS

EDITED BY
Panagiotis Tsiotras,
Georgia Institute of Technology, United States

REVIEWED BY
Andrea Testa,
University of Bologna, Italy
Jakob Foerster,
University of Oxford, United Kingdom
Matthew Jackson,
University of Oxford, United Kingdom in collaboration with reviewer JF

*CORRESPONDENCE
Michael A. Sebok,
sebokm@udel.edu

†PRESENT ADDRESS
Michael A. Sebok, Manufacturing Demonstration Facility, Oak Ridge National Laboratory, Oak Ridge, TN, United States

RECEIVED 23 February 2023
ACCEPTED 14 February 2024
PUBLISHED 13 March 2024

CITATION
Sebok MA and Tanner HG (2024), Cooperative planning for physically interacting heterogeneous robots.
Front. Robot. AI 11:1172105.
doi: 10.3389/frobt.2024.1172105

Heterogeneous multi-agent systems can be deployed to complete a variety of tasks, including some that are impossible using a single generic modality. This paper introduces an approach to solving the problem of cooperative behavior planning in small heterogeneous robot teams where members can both function independently as well as physically interact with each other in ways that give rise to additional functionality. This approach enables, for the first time, the cooperative completion of tasks that are infeasible when using any single modality from those agents comprising the team.

KEYWORDS

heterogeneous multi-agent systems, robot planning and control, cooperative planning, hybrid automata, physical interaction

1 Introduction

While cooperative robot behavior for spatial deployment and object manipulation has been studied in length both for homogeneous and heterogeneous robot groups, the latter case can leverage both plurality and *diversity* to complete tasks that would be nontrivial or impossible with a homogeneous group of any single available robot modality. The possibility of exploiting robot heterogeneity and diversity coupled with *physical interaction* to carry out otherwise infeasible missions has not been adequately investigated, and as a result part of a collaborative system's potential remains untapped. In the function of heterogeneous human teams, teammate diversity can be instrumental when leveraged properly in a variety of application scenarios. There are instances in literature where robot heterogeneity was essential in accomplishing the task at hand. Mellinger et al. (2013) use a combination of a wheeled mobile robot and a quadrotor to extend the operational range of the latter. Another example is the work of Kiener and von Stryk (2007) in which a robotic wheelchair needs to transport a humanoid to a location where the humanoid needs to kick a ball. Yet another example is the approach of Huntsberger et al. (2003) who consider multiple planetary rovers which may not necessarily differ morphologically, but they have different capabilities. In general, however, fully leveraging diversity and heterogeneity in heterogeneous robot teams is nontrivial. The development of efficient planning and control algorithms for heterogeneous groups presents unique challenges due to the varied capabilities and operational constraints of individual agents. What is more, the intricacies of nontrivial types of physical interaction necessitate the incorporation of some of the agents' underlying continuous dynamics, which significantly complicates analysis and inhibits planning algorithm scalability.

Consider a scenario where a small heterogeneous robot team is tasked to complete a mission that is beyond the inherent capabilities of any of its members. Assume that the inherent capabilities of robot team members are enabled by appropriate underlying control

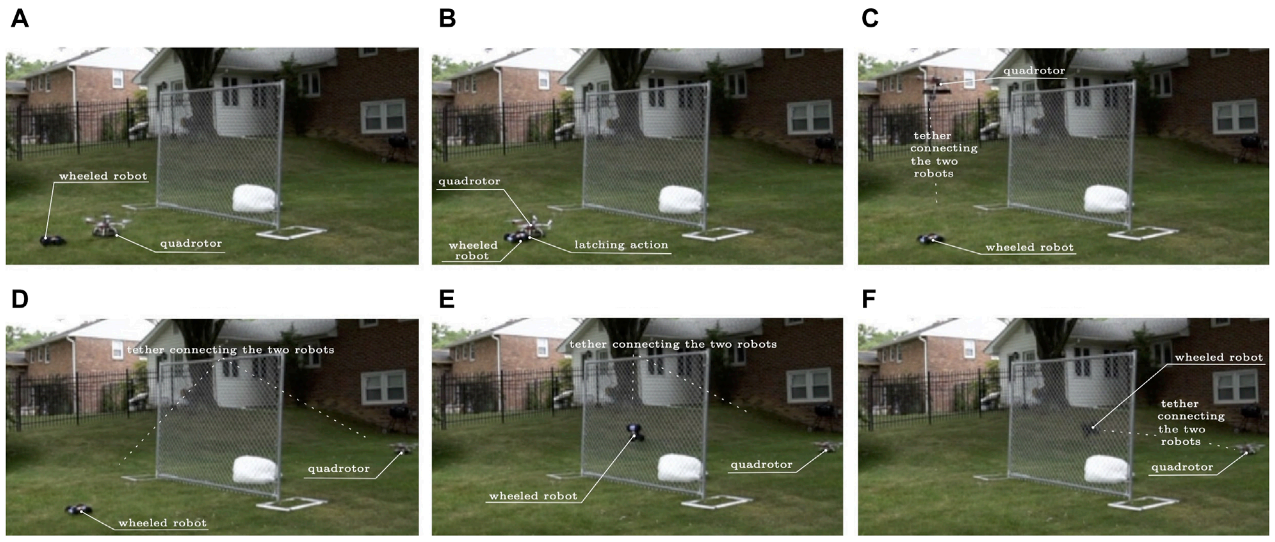


FIGURE 1
Cooperation of robots utilizing physical interaction for overcoming otherwise insurmountable obstacles. **(A)** The quadrotor lands in front of the ground vehicle; **(B)** The tip of the ground vehicle's spool attaches to the quadrotor's velcro apron; **(C)** The quadrotor takes off to fly over the fence, tethered on the ground vehicle which lets the line reel out; **(D)** The quadrotor lands on the other side of the fence; **(E)** The ground robot uses its powered spool to reel in the line and climb vertically against the fence; **(F)** The ground robot has made it over the fence and is on its way to the soft landing area on the other side of the fence.

loops. Each control loop gives rise to one of a finite collection of possible actions that the robot can carry out. The robots comprising this team are capable of physically interacting (e.g., pushing, pulling, lifting, etc.) with each other and possibly with portions of their environment.

Physical interaction between a robot and its environment has been demonstrated to be potentially advantageous for mission completion (Karydis et al., 2014; Stager and Tanner, 2020; Stager and Tanner, 2016; Stager and Tanner, 2019). In addition, some types of physical interaction between robots can significantly change the dynamics of the agents involved (e.g., when an aerial robot lifts a wheeled robot, the aerial robot's inertial characteristics change and contact constraints for the ground robot are now lifted) (Mellinger et al., 2011). Here we are focusing on the latter kind of interaction, and are particularly interested in tasks that no single modality of robots in this team, regardless its scale, can carry out on their own (e.g., wheeled robots cannot jump to the other side of a fence; see Figure 1); meanwhile, a deliberate interaction and physical coupling between the heterogeneous teammates can make this task possible. This paper describes a methodological framework for identifying the type of physical interactions needed for the desired cooperative task completion, and for planning the sequence of actions that the robot teammates need to undertake to execute the task. It should be noted that physical interaction with the environment as well as non-physical interaction between agents are also allowable within this framework. However, non-physical interactions have been extensively studied within the existing literature so this work primarily focuses on the planning difficulties unique to physical interaction.

In the case of Figure 1, the sequence of cooperative actions enabling the ground robot to climb over the fence utilizing "togglable" physical interaction with an aerial vehicle through a powered spool mechanism was manually scripted, and the robots were remotely controlled. This paper offers a methodology for *automating* such planning and mission execution processes through an algorithmic framework focusing on a subclass of heterogeneous robot groups where physical interaction between teammates can transform the dynamics of the combined system. This application of physical interaction enables planned actions that are not within the action space of any agent acting individually. Existing planning and control theoretical frameworks for heterogeneous groups are poorly equipped to model system functionality that exists due to physical interaction and fully leverage this type of heterogeneity. This paper therefore (i) narrows the gap by providing methodological innovations in modeling cooperative behavior arising from nontrivial system coupling, and (ii) demonstrates that they facilitate the discovery of cooperative robot plans that are beyond the solution space of existing approaches.

To enable and regulate coupling between the cooperating robotic agents, as well as accurately predict agent evolution in their physical workspace, a suitable hybrid dynamical system modeling formulation is adapted to capture the full closed-loop dynamics of the robot team members. Then, a new formal operation for *composing system behavior* in heterogeneous robot teams, inspired by constructs found in the field of computational and mathematical linguistics, is introduced and exploited to create a comprehensive model of system dynamics and available behaviors. This (discrete) behavior composition approach offers two distinct advantages: it curbs the growth of computational complexity when robot teams increase in size, and it exposes new behaviors that become possible when individual team members couple with each other. Subsequently, discrete abstractions of the closed-loop hybrid agent dynamics are utilized to transfer the planning problem to a purely discrete space. When formulated in this way, and with guidance from appropriate objective functions that draw from the continuous system dynamics, discrete optimization algorithms can discover new solutions to cooperative planning problems that until recently would be considered infeasible.

This cooperative planning methodological framework is validated in two heterogeneous system case studies. The first is motivated by the scenario of Figure 1 and is also utilized in the remainder of this paper as a running example which strongly leverages physical interaction between individual robots. This study utilizes a pairing between an unmanned aerial vehicle (UAV) and an unmanned ground vehicle (UGV) through a novel electromechanical design that allows optional and controlled UAV–UGV tethering. The second case study leverages a different type of physical interaction with a wheeled UGV and a walking UGV that join together magnetically to give rise to a mobile manipulator.

2 Related work

2.1 Physical interaction in multi-agent systems

Physical interaction between robots is commonly used to extend the functionality of multi-robot systems. Tethered connections between robotic vehicles are only one example of how physical interaction can be realized and leveraged. In this particular direction, for instance, one can bring up the case of Miki et al. (2019), who feature a tethered UAV/UGV system that can be used to climb obstacles and which bears conceptual similarities to the first case study treated in this paper. Others have used a tether simply to deliver additional power in a UAV/UGV system (Ogusu et al., 2020; Sutera et al., 2020). Tethers were also used in a multi-UGV system to lower one of the vehicles over the edge of a cliff (Pirjanian et al., 2002) and a reversible connection (Casarez and Fearing, 2016) which allows two crawling vehicles to traverse a vertical step. Outside the tethered robotics area, we find the work of Mathews et al. (2010) who leverage a supervisory UAV in coordinating a group of UGVs to connect together to form a bridge. The aforementioned cases are merely a small sample of examples in literature where physical interaction realized via mechanisms subject to complementarity constraints has been utilized in the context of cooperative multi-robot behavior. However, most of these attempts tend to focus primarily on the system dynamics of physical interaction *without* addressing the question of how such interaction can be utilized as a modality within a versatile high-level planning framework.

2.2 Optimal planning and control of hybrid systems

The simultaneous and coupled evolution of discrete and continuous dynamics featured in hybrid dynamical systems present enormous challenges to optimal control and planning (Cassandras et al., 2001; Egerstedt et al., 2006; Zhu and Antsaklis, 2015). In optimal control formulations like the ones mentioned, the sequence of discrete modes through which the system goes is typically *a priori* given; in this paper, this sequence is one of the key variables to be solved for. In general, existing methods for coordinating optimal behavior in heterogeneous (hybrid) multi-agent dynamical systems can generally be placed on a spectrum where on one end one finds purely discrete planners and on the other continuous optimization methods. One early example of the former for heterogeneous systems was proposed by Parker (2001) and enabled groups of ground vehicles to complete tasks such as cooperatively pushing and repositioning boxes. Fully discrete methods focus on optimizing over the set of discrete states while abstracting or approximating the dynamics of the agents on those states. Planners that work in the continuous space, on the other hand, can account for more complex dynamics but tend to struggle with hybrid mode switching, especially when the underlying system model changes drastically in terms of its continuous, discrete, or hybrid behavior. One example of a continuous approach to hybrid system planning is that of Posa et al. (2014) which efficiently handles complementarity (contact) constraints without having to distinguish between different discrete modes and engage in any sort of combinatorial analysis. Another example of an optimal planning and control method for hybrid systems that lies in between the two ends of the spectrum and could be applicable in the context of problems addressed in this paper is the hybrid optimal control method of Zhao et al. (2020). It is capable of handling both discrete and continuous dynamics, albeit of relatively small dimensions and with the latter having to be expressed in polynomial form. Still, because of its capacity to integrate nonlinear continuous dynamics it maps closer to the continuous end of the spectrum of available planning algorithms. On the reciprocal end, one may note the GRSTAPS algorithm (Messing et al., 2022), which is primarily discrete, employing a high-level abstraction of the agent dynamics to derive optimal plans in multi-agent systems. Discrete planning algorithms fare better for systems with numerous agents and tasks, whereas continuous planners are more capable of handling continuous dynamics and may have a better chance of producing feasible motion plans.

2.3 Existing limitations and proposed solution

Existing planners are challenged when called to plan for heterogeneous multi-robot systems that require leveraging physical interaction between their teammates to complete tasks. The contact mechanics and its impact on the component system dynamics place significant challenges to existing approaches, often leading to paradoxical solutions or failing to identify feasible ones (Johnson et al., 2016).

Continuous planners consistently fail to capture the emergent behaviors enabled by physically interacting agents. And while discrete planners are considerably more efficient and scale better with the number of agents involved compared to continuous ones, the often over-simplifying approximations of the continuous dynamics of the agents can either lead to infeasible plans or miss a whole range of solutions that are enabled explicitly by the continuous dynamics that have been abstracted away. Planners that combine discrete and continuous aspects of system behavior are still severely limited in the classes of system dynamics they can handle, the size of the overall system in terms of number of states, and the lack of guarantees for finding feasible solutions.

The novel planner described in this paper attempts to retain some of the computational efficiency of discrete planners while taking the continuous dynamics of each location in the hybrid automaton into account—the caveat here, and the feature that facilitates more dynamically consistent abstractions for subsequent discrete planning, is the closure of control loops at the continuous layer.

3 Hybrid automata for cooperative behavior

One commonly employed modeling formalism for modeling an individual member of a multi-agent system is the *deterministic finite automaton*. However, such an automaton is a purely discrete model of computation and as such has severe limitations with respect to capturing the physical dynamics of a robot. On the other hand, its simplicity allows for plans to be generated in an efficient manner. A finite automaton can be defined as follows:

Definition 1: A finite automaton is a tuple $\mathcal{A} = \langle L, A, L_0, F, \Delta \rangle$ comprised of

L	a finite set of locations;(i)
A	a finite set of labels;(ii)
L_0	a finite set of initial (starting) locations
F	a finite set of final (end) locations;(iii)
Δ	the transition function.(iv)

(i) the discrete states of the automaton;
(ii) the alphabet of (input) symbols that trigger transitions between discrete states and thus label the edges of the automaton graph that represents the discrete behavior of the system;
(iii) a subset of L on which when the automaton is, it is said that it has accepted a finite string made of labels in A;
(iv) a function $\Delta : L \times A \to L$ which determines the new state of the automaton based on the current state and input label.

There is a particular discrete operation on *finite* automata that combines them in a way that yields an outcome system, the behavior of which is neither the intersection nor the union of the behaviors of its factors. In addition, the number of locations in the resulting automaton increases linearly (rather than exponentially) with respect to the number of locations of the components. The motivation for such an operation is found in its utility for language identification in the limit in the context of mathematical linguistics (Heinz and Rogers, 2013). In this paper, the idea behind this mechanism will find a new application in the domain of robotic planning. Let us first introduce the original construction:

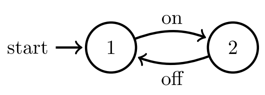
Definition 2: (cf. Heinz and Rogers, 2013). Given two finite automata $\mathcal{A}_1 = \langle L_1, A, L_{01}, F_1, \Delta_1 \rangle$, $\mathcal{A}_2 = \langle L_2, A, L_{02}, F_2, \Delta_2 \rangle$, their *join* is a finite automaton $\mathcal{A}_1 \sqcup \mathcal{A}_2 = \langle L_1 \cup L_2, A, L_{01} \cup L_{02}, F_1 \cup F_2, \Delta_1 \cup \Delta_2 \rangle$.

This *join* operation enables the composition of individual automata representing each robotic agent in the multi-agent system into a single joined automaton which represents a comprehensive and efficient model of system capabilities. However, classical finite automata are not sufficient to model the complex dynamics that arise in some cooperative multi-agent systems and which must be considered to generate feasible planning solutions. To this end, the mathematical formulation in this paper leverages *hybrid automata* (Figure 2) and features a unique hybrid automata-based planning method. This method is computationally efficient and robust enough to compute plans for systems where physical interaction between agents is absolutely necessary for task completion. Hybrid system modeling formulations are attractive in this context because they allow one to capture salient features of the continuous robot dynamics that are important for task completion. They also offer appropriate mathematical handles to high-level discrete planners and optimizers. One of the multiple challenges in working with multi-agent systems in a hybrid framework is analytical *and* computational complexity. While the latter may be easy to see on the outset, part of the difficulty associated with the former is the need for incorporating (generally nonlinear) continuous dynamics as well as the possibility that the discrete structure of the overall system can experience changes during the course of the plan evolution. This paper builds on a particular modeling formalism for hybrid automata (van der Schaft and Schumacher, 2000):

Definition 3: A hybrid automaton consists of a tuple $\mathcal{H} = \langle L, X, A, W, E, \mathrm{Init}, \mathrm{Inv}, \mathrm{Act} \rangle$ with components as follows:

L	a finite set of locations;(i)
X	the continuous state space;(ii)
A	a finite set of labels;(iii)
W	the continuous communication space;(iv)
E	a finite set of events;(v)
Init	a set of initial states;(vi)
Inv	the invariants of continuous dynamics;(vii)
Act	activities, i.e., continuous vector fields;(viii)

(i) the nodes in the hybrid automaton graph;
(ii) a subset of $\mathbb{R}^n$ where the system's continuous variables belong to;
(iii) the discrete input alphabet that trigger transitions in the automaton's graph and label its edges;
(iv) the domain of external variables that affect its behavior;
(v) the automaton's transitions, as tuples of the form ⟨ initial location, label, guard condition, resets on continuous variables, new location ⟩;
(vi) a set of initial states $(\ell_0, x_0) \in L \times X$ at which the hybrid automaton can be initiated;
(vii) the subsets of X, one for every $\ell \in L$ which remain invariant under the dynamics imposed by $\mathrm{Act}(\ell)$.



$$\begin{aligned} L &:\{1,2\}, X : \{x\} \\ A &:\{\text{on}, \text{off}\}, W : \{t,u\} \\ E &:\{1, \text{on}, x \le 23, x_f = x_i, 2\}, \quad \{2, \text{off}, x \ge 24, x_f = x_i, 1\} \\ \text{Init} &:\{1, x_i\} \\ \text{Inv} &: \text{Inv}_1 : x \ge 23, \text{Inv}_2 : x \le 24 \\ \text{Act} &: \text{Act}_1 : \dot{x} = f(x,t), \text{Act}_2 : \dot{x} = f(x,t) + u \end{aligned}$$

FIGURE 2
A standard wall thermostat as a typical example of a hybrid automaton: There are only two locations which correspond to the heater being either on or off. There is a single state variable, the indoor temperature x as well as two communication variables, the input heat u and the external temperature t. The hybrid automaton is constructed from these variables as a set of equations that govern system behavior and license discrete transitions between the two system locations.

(viii) a mapping of activities Act:$L \to TX$ which associates a continuous vector field in TX to each location in L.

In the way we utilize Definition 3 in this paper, we understand the invariants of locations as the limit sets (post conditions) of the activities labeled in the events leading to these locations, and the domains (pre conditions) of the activities of events that depart these locations. Note also that this hybrid model features no explicit (continuous) control inputs. Instead, we utilize the set W of communication variables to parameterize pre-defined feedback control action at each location, and pass on to the hybrid system relevant information about its environment and its teammates. The guard condition is key to determining the currently enabled transitions in the automata based on the current system state. We allow the guard conditions to be parameterized both by the continuous state space X as well as the communication variables W, a fact that aids in the construction of prerequisite conditions for collaborative action. We assume that the pre-defined continuous controllers in each location faithfully implement the high-level behavior prescribed by Act(ℓ), e.g., convergence to a desired region of the state space. One example of a hybrid system model is that of a wall thermostat controlling the heat within a home (Figure 2).

As written, Definition 2 is not suited for use with hybrid automata because it is constructed to operate on purely discrete systems wihtout any continuous dynamics. To address this issue, we have introduced Definition 3 to enable cooperative hybrid behavior. Thus, when heterogeneous systems are designed to interact with each other, their models reflect this capacity. One way of incorporating collaborative behavior is to include in each system (cooperative) events that can (only) be triggered by other systems. These events must be disabled within individual agent models so that they cannot be triggered by one agent acting alone. These modified events that result from collaborative actions are lacking a specified initial location, since they are only enabled in cooperation with another system which provides the initial location and matching label. The system, thus, could never trigger those events on its own, but the capacity is built into its model for the prospect of interaction with other systems. Labels "borrowed" from the label sets of other systems provide the framework for collaborative events that are only enabled when the requisite individual systems are joined together.

With this in mind, we implement the former blocking mechanism, which allows for events to incorporate labels external to the system. This is facilitated by expanding its label set to include the labels of the other components, i.e., setting $A = A_1 \cup A_2$ as a common label set. This way, the component hybrid automata match the modeling specifications of the finite automata in Definition 2 (which share a common label set). The underlying assumption here is that $\mathcal{H}_1$ and $\mathcal{H}_2$ may not be able to utilize *every* label in A to trigger events; there could be some events that remain dormant when each system operates in isolation. Additionally, there may be certain locations within the individual hybrid automata that are essentially unreachable when operating in isolation as they might only be reachable under some sequence involving one of the aforementioned dormant events.

With the understanding that execution evolves in a *turn-based* manner, i.e., the component systems take turns executing actions and do not evolve concurrently—just as in the case of the finite automata join operation of Definition 2— we are now in position to extend the latter definition to hybrid dynamical systems:

Definition 4: Given two hybrid automata $\mathcal{H}_1 = \langle L_1, L_{01}, F_1, X_1, A, W_1, E_1, \text{Init}_1, \text{Inv}_1, \text{Act}_1 \rangle$, $\mathcal{H}_2 = \langle L_2, L_{02}, F_2, X_2, A, W_2, E_2, \text{Init}_2, \text{Inv}_2, \text{Act}_2 \rangle$, their *join* is a hybrid automaton $\mathcal{H}_1 \sqcup \mathcal{H}_2 = \langle L_1 \cup L_2, L_{01} \cup L_{02}, F_1 \cup F_2, X_1 \times X_2, A, W_1 \times W_2, E_1 \cup E_2, \text{Init}_1 \cup \text{Init}_2, \text{Inv}_1 \times \text{Inv}_2, \text{Act}_1 \times \text{Act}_2 \rangle$.

Note that the outcome of the operation the above definition does not strictly conform to the specifications of Definition 3 in the sense that $\text{Init}_1 \cup \text{Init}_2 \notin (L_1 \cup L_2) \times (X_1 \times X_2)$ but this difference is inconsequential given the turn-based evolution of the machine, and does not interfere with recursive applications of the operation. What is important is that with the union of locations and the union of events, the join system can now enable those previously dormant events. The mechanism of Definition 4 thus gives rise to a richer cooperative hybrid system behavior, which—as the colloquial saying goes—is "bigger than the sum of the parts."

Now that we have a suitable operation for capturing unique new cooperative behavior involving interacting hybrid dynamical systems through another hybrid automaton, the formalism can be leveraged to create models of multi-agent systems where cooperative interaction is essential to task completion. The (discrete) event dynamics of a hybrid automaton $\mathcal{H}$ can be visually represented in the form of a directed graph, the nodes and edges of which are labeled by L and A, respectively. In order to illustrate the functionality of the operation introduced in Definition 4, consider the automata in Figure 3. In Figure 3, events (transitions) that are not labeled

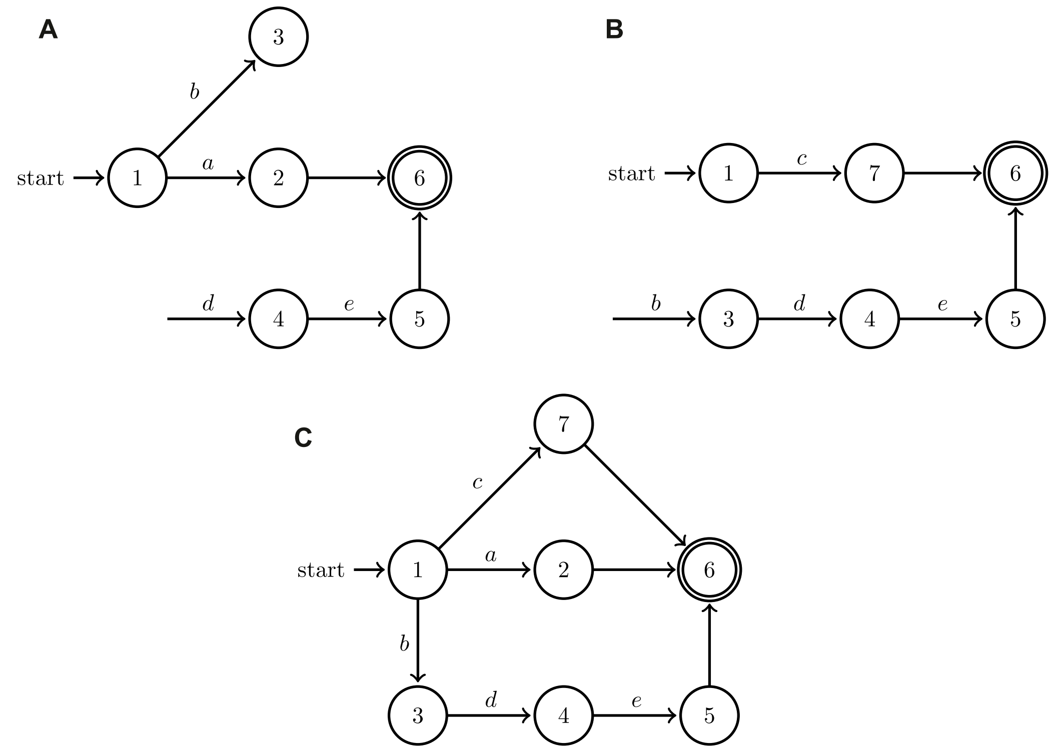


FIGURE 3
A minimal example of the application of the join operation to the graphs of two hybrid automata depicted in **(A,B)**; the outcome of the join operation is showcased in **(C)**. Both of the two factor automata share some input symbols (labels) and allow for discrete state resets.

express a decision on the part of the system to terminate execution at that point by transitioning to a final state. Notice in Figure 3 the mechanism for blocking cooperative behavior through event withholding when systems operate in isolation: location 4 (and consequently also 5) is unreachable in both systems; in system (A) the event associated to the transition from location 3 to location 4 is not included, and similarly in system (B) the event from 1 to 3 is missing. However, when the two systems join, their events combine and now both 4 and 5 can be reached through some collaborative action [first (A) moves to 3 with b, and then (B) moves to 4 with d].

This formulation, therefore, can capture both the behavior of individual systems when they operate in isolation as well as the new cooperative behaviors that become enabled when they are composed using the join operation under the turn-based assumption. Notice also that the outcome of the join operation [system (c) in Figure 3], is not significantly bigger compared to $\mathcal{H}_1$ and $\mathcal{H}_2$. The size of this outcome scales *linearly* with the size of the location sets of its components.

While this paper only focuses on systems with two agents for representation brevity and clarity, Definition 4 directly extends recursively to any finite number of collaborative hybrid systems.

4 Cooperative plans with physical interaction

The modeling formulation of Section 3 captures the mechanics of deliberate physical interaction between heterogeneous robots and exposes cooperative plans that classical transition system composition operations ignore. Now, it is the job of an appropriately guided search algorithm to shift through the finite space of possible cooperative turn-based plans to single out the ones that allow the multi-agent system to achieve its objective.

It needs to be emphasized here that a naive approach utilizing a generic graph search is bound to fail: for example, the shortest path from initial to final locations in the join automaton of Figure 3C would be a or c, but that is not a valid solution to any task specification requiring collaborative action e. The search needs to be guided with information from the continuous dynamics of the hybrid automata and cannot be conducted using solely the discrete information contained in the automata graphs. This section outlines the implementation of efficient search methodology to find effective cooperative solution plans which can then be implemented by the agents in a heterogeneous system.

4.1 Discretization of the workspace

The first step in the planning process is to discretize the continuous shared robot workspace. This discretization is dictated by the structure of the guards and the invariants of the hybrid join automaton. Workspace cells produced by the discretization process are associated to locations of the join system and the assignment may not be one-to-one, i.e., cells may belong to multiple locations. This mapping of cells to locations results in each location having a set of cells assigned to its guard and another set assigned to its invariant.

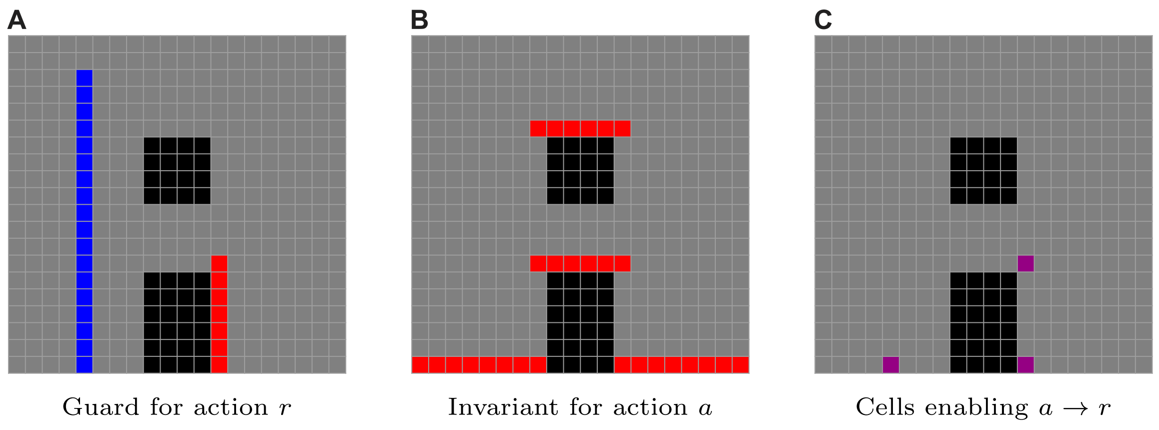


FIGURE 4
In this example drawn from the UAV–UGV case study, analyzed in more detail in Section 5, a transition from a rolling activity *a* to a reeling activity *r* for a UGV is enabled at the intersection **(C)** of the guard of *r* **(A)** with the invariant of *a* **(B)**.

Informally, the guard set represents locations where a particular activity can be (forcibly) triggered (activities are linked to locations, the latter pointed to by labeled edges in Figure 3C). Similarly, the invariant represents the reachable space of the current activity (i.e., closed-loop dynamics). Cells associated with the same location that are mutually reachable using (possibly different parameterizations of) the low-level controller corresponding to a given activity can be grouped together into collections of cells subsequently referred to as *supercells*. Assume that the workspace is divided into n cells with individual cells denoted q_i with $i{:}1 \to n$. For each location $\ell \in L$ and associated label $a \in A$, supercells are constructed as a set of adjacent cells with overlapping faces (i.e., $\{q_i, q_j, \ldots\}$) that all satisfy the continuous contraints of a particular invariant or guard. Each supercell associated with $\mathrm{Inv}(\ell)$ is denoted $I^r_\ell = \{q_i, q_j, \ldots\}$ and each supercell associated with $\mathrm{Guard}(a)$ is denoted $G^r_a = \{q_i, q_j, \ldots\}$ with $\ell \in L$, $a \in A$ and $r{:}1 \to m$ where m is the number of supercells associated with that invariant or guard. Constructed supercells are labeled with their corresponding action and an index since there can be several disjoint supercells within the guard or invariant of a particular location. Let the superset of all invariant supercells be denoted S_I and the superset of all guard supercells be denoted S_G. Then, the individual invariant supercells collectively form set $S_I = \{I^1_a, I^2_a, I^1_b, \ldots\}$ and the individual guard supercells form set $S_G = \{G^1_a, G^2_a, G^1_b, \ldots\}$. In this paper, the relationship between the set of supercells for the guards of and an action and the set for the invariants is one-to-one so that each guard always has an associated invariant. Individual cells are permitted to belong to multiple supercells corresponding to different locations of the hybrid automata, e.g., $q_i \in I^1_a, q_i \in I^2_b$. Exploiting this continuous state abstraction significantly reduces computation time when searching through the discretized workspace for an action sequence. For an event to be enabled and the corresponding transition to occur, the system must find itself at an intersection of the current invariant with the guard of the event's destination location. The planner then optimizes over a sequence of labels (i.e., events) that take the system between different supercells, focusing on regions where a guard supercell for one location intersects with an invariant supercell of another location (see Figure 4).

4.2 The search for the right action sequence

A key input to our discrete planner is the set of cells where a transition between hybrid automaton locations is licensed. This happens when a supercell associated with the guard of one location intersects with a supercell associated with the invariant of another location. The set of transition cells for a pair of supercells I^i_a, G^j_b can be written as

$$Q_{I^i_a \to G^j_b} = \left\{ q_i \mid q_i \in I^i_a \cap G^j_b \ \&\ \langle \ell_i, \cdot, \cdot, \cdot, \ell_j \rangle \in E \right\}$$

where ℓ_i is the location associated with supercell I^i_a and ℓ_j is the location associated with G^j_b. For every pairing of $I^i_a \in S_I$ and $G^j_b \in S_G$ such that $I^i_a \cap G^j_b \neq \emptyset$, there exists one or more cells where a transition is enabled between the location associated with action a and the location associated with b. Only one of these transition cells is considered by the planner when calculating the cost to transition between the corresponding supercells. This "ideal" transition cell, denoted $q_m^{a_i \to b_j}$, is the cell $q_i \in Q_{I^i_a \to G^j_b}$ which minimizes the future cost h_j evaluated at the cell centroids. If multiple cells fit this specification, one can be selected without significant impact on the nature of the final solution, since within the (landing) supercell individual cells are mutually reachable. Note that, before a transition between supercells occurs, the robot is located in cell $q_m \in I^i_a \cap G^j_b$ and after transition is located in the same cell within the invariant of the new supercell $q_m \in I^j_b$ until the next transition is enabled. With slight abuse of notation, to simplify notation in the description of our planner implementation, we represent the supercell transition $I^i_a \to G^j_b \to I^j_b$ as $\ell_i \to \ell_j$. With this in mind, we construct a matrix of transitions T consisting of tuples of the form $\{\ell_i, \ell_j, q_m\}$, containing every pair of supercells ℓ_i, ℓ_j licensed for transition, and the minimum cost q_m for that transition.

The planner developed for this problem (Algorithm 1) is a modified version of a forward $A*$ algorithm that optimizes over the subset of cells where transitions between supercells can occur—instead of considering all the cells in the workspace. This algorithm is now informed of the (short-term) cost a_{ij}

```
Inputs:
  $T, M$
Initialize:
  $upper \leftarrow \infty$
  $c_m \leftarrow \infty \forall \ell \in M$
  $open \leftarrow \{\ell_1\}$ $\ell_1$ is the initial location
while $open \neq \emptyset$ do
  Select sequence $\{\ell_1, \ell_2, \ell_n, \ldots\}$ with minimum running cost $c_i$ and remove from $open$
  for all $\ell_i, \ell_j, q_m \in T \mid \ell_i = \ell_n$ do
    $c_j = c_i + a_{ij}(q_m) + h_j(q_m)$
    if $c_j < upper$ and $c_i + a_{ij}(q_m) < c_m(\ell_j)$ then
      Add $\{\ell_1, \ell_2, \ell_i, \ell_j, \ldots\}$ to $open$
      $c_m(\ell_j) = c_i + a_{ij}(q_m)$
    end if
  end for
  if $\ell_j = \ell_f$ then
    $c_f = c_j$
    $upper = c_j$
  end if
end while
```

Algorithm 1. $A*$ Planner with Supercells.

of reaching the transition cell between (super) cells ℓ_i and ℓ_j using a continuous function of the join hybrid system's state. A design choice adopted in our implementation for quantifying the cost of such transitions is based on a particular type of scalar potential functions of the continuous state of the join hybrid automaton. These functions are known as *navigation functions* (Rimon and Koditschek, 1992) and they can serve as surrogates for a combined metric of "distance to goal" and "distance from constraint boundaries." Just as in a typical $A*$ algorithm, the planner tracks the running total cost c_i for each candidate sequence of transitions between supercells in a priority queue typically referred to as the *open* set. Additionally, there is a parameter termed *upper*, which tracks the cost of the current best sequence that reaches goal location ℓ_f.

The planner tracks the current minimum cost c_m (initial cost set to ∞) to reach each location (supercell) ℓ in the workspace within matrix M which contains tuples of form $\{\ell, c_m\}$. When extending the candidate sequences, the planner first selects the sequence with minimum running cost c_i and removes it from the set *open*; it then considers the actual cost of the next transition a_{ij} as well as a monotone estimate of the future (long-term) cost h_j for reaching the goal state ℓ_f from ℓ_j, for each ℓ_j reachable from ℓ_i, with h_j computed in a similar way as a_{ij}—see Section 5.2 for a concrete example of how a_{ij} and h_j can be defined with the help of a navigation function. (We note that there is considerable freedom in defining the cost functions for a_{ij} and h_j, and it would make sense for those structures to be determined based on the desired optimization criteria for a given task.) These candidate extensions to the minimum cost sequence are evaluated using their running cost calculated as $c_j = c_i + a_{ij}(\ell_j) + h_j(\ell_j)$. Candidate sequences which satisfy two conditions: (i) that $c_j < upper$, and that (ii) $a_{ij} + h_j < c_m$, where c_m is the current minimum cost to reach location ℓ_j, are added to *open*. When the planner finds a sequence that reaches the goal state so that $\ell_j = \ell_f$, it compares the cost for that sequence, c_f, to *upper*. If $c_f < upper$, then that sequence of transitions becomes the current minimum cost sequence to reach the goal and c_f is set as the new *upper*. This process continues until there are no more sequences in *open* and the best transition sequence with minimum cost c_f has been determined.

One note of caution is that the planner is not expected to monotonically decrease this "distance to goal surrogate" since a myopic and dynamics-agnostic gradient descent (despite any navigation function properties) offers no guarantee of task completion. A sufficiently *forward-looking* search algorithm, involving a combination of a running cost a_{ij} and a future cost h_j for each sequenced location ℓ_j is able to overcome a local cost increase along a path to the goal. An important by-product of the search for this path is the sequence of cells that mark the transitions between the activities in the solution sequence. This information is subsequently used to parameterize the underlying control laws associated with each activity, and thus implement the derived plan within the complete hybrid system.

5 Case study 1: a tethered UAV–UGV system

This section illustrates a slightly more complex instantiation of the planning problem shown in Figure 1. Again, we have a UGV and a UAV with the capability of attaching an actuated tether between them. The goal for the system is to move the UGV from a location on one side of a vertical wall, to one on the other side. The new instantiation here includes an additional obstacle over the vertical wall (as if the wall has a window that the robots can go through). In addition, the UAV has now an overhead perching location, which serves as a secure anchoring point for the tethered UGV, to lower itself down to the ground by unwinding its tether. In this problem setup, there are several interesting challenges: among them, the nonlinear continuous dynamics of the two vehicles, which approximately capture the behavior exhibited when the UGV swings in a pendulum fashion suspended by its tether and when it moves up or down from an elevated position; the obstacles that need to be avoided; the need to manage and account for the shape of the flexible tether within the cluttered workspace; and methods to identify the workspace-tether relative configurations that enable tension forces to be applied to the UGV in the desired directions.

The UGV hybrid automaton has a label set that includes the subset $A_g = \{a, d, a', r', f'\}$ where a (respectively, a') expresses a (respectively, tethered) rolling behavior and d denotes latching of the tether end on the UAV, while r' refers to controlled vertical ascent/descent using the tether with its other end suspended at an elevated position and f' corresponds to a transient swinging motion while tethered until momentum is absorbed by impact with a workspace boundary. The UAV has the a label set that includes the subset $A_a = \{\delta, \beta, \gamma'\}$ where δ enbales the UAV to fly to an aerial waypoint and hover there, β is the landing action where the UAV comes to rest on one of the workspace boundaries and γ' express the tethered UAV perching at its goal location to provide an anchoring point for the free end of the UGV tether. In this implementation, we allow for the UGV to *swing* (possibly impacting workspace boundaries) as it reels up or down with its tether and model this behavior in the UGV continuous dynamics.

The continuous dynamics of the combined UAV-UGV system, imagined to evolve on a vertical plane for simplicity, are defined on a 10-dimensional state space X comprised of tuples of the form $\{x,y,\phi,l,\theta\}$ and their first derivatives. The variable tuple (x,y,ϕ) parameterizes the $SE(2)$ pose of the UAV on the vertical plane of motion, and the pair (l,θ) (length/angle) provide a polar parameterization of the Cartesian position of the UGV on the same plane, relative to a *hinge point* (see Section 5.1) where the other end of the tether is attached to—during untethered horizontal motion, a virtual hinge point can be introduced on the semiaxis of UGV motion direction.

The hybrid system modeling formalism adopted (Definition 3) includes a set W of continuous communication variables, which for this particular case study can be realized as the set $\{q_h,u,u_1,u_2\}$ comprised of UAV control inputs u_1 and u_2, UGV wheel/winch acceleration u (depending on the hybrid mode), and the (piecewise constant) location q_h of the hinge point relative to which the UGV position parameterization is derived.

The set of events E represents the edges within the join automaton graph as well as the conditions on the state variables which license each transition. In addition to the set of events themselves, it is necessary to construct functions for the state reset (jump) and guard conditions for each transition within the hybrid automaton. The event set E for the UAV/UGV system is written as:

$$\begin{array}{ll}
E_1 = \langle 1,\delta,G_\delta,J_{(1,9)},9\rangle & E_2 = \langle 9,\beta,G_\beta,J_{(9,3)},3\rangle \\
E_3 = \langle 3,\delta,G_\delta,J_{(3,9)},9\rangle & E_4 = \langle 4,\gamma',G_{\gamma'},J_{(4,5)},5\rangle \\
E_5 = \langle 1,a,G_a,J_{(1,2)},2\rangle & E_6 = \langle 3,a,G_a,J_{(3,2)},2\rangle \\
E_7 = \langle 3,d,G_d,J_{(3,4)},4\rangle & E_8 = \langle 5,r',G_{r'},J_{(5,6)},6\rangle \\
E_9 = \langle 5,f',G_{f'},J_{(5,7)},7\rangle & E_{10} = \langle 5,a',G_{a'},J_{(5,10)},10\rangle \\
E_{11} = \langle 6,a',G_{a'},J_{(6,10)},10\rangle & E_{12} = \langle 6,f',G_{f'},J_{(6,7)},7\rangle \\
E_{13} = \langle 7,a',G_{a'},J_{(7,10)},10\rangle & E_{14} = \langle 7,r',G_{r'},J_{(7,6)},6\rangle \\
E_{15} = \langle 7,f',G_{f'},J_{(7,7)},7\rangle & E_{16} = \langle 10,f',G_{f'},J_{(10,7)},7\rangle \\
E_{17} = \langle 10,r',G_{r'},J_{(10,6)},6\rangle & E_{18} = \langle 3,\ltimes,G_\ltimes,J_{(3,8)},8\rangle \\
E_{19} = \langle 5,\ltimes,G_\ltimes,J_{(5,8)},8\rangle & E_{20} = \langle 2,\ltimes,G_\ltimes,J_{(2,8)},8\rangle \\
E_{21} = \langle 6,\ltimes,G_\ltimes,J_{(6,8)},8\rangle & E_{22} = \langle 7,\ltimes,G_\ltimes,J_{(7,8)},8\rangle \\
E_{23} = \langle 10,\ltimes,G_\ltimes,J_{(10,8)},8\rangle &
\end{array}$$

The set of guards within the join automaton is constructed as constraints on the set of state variables and also the communication variables, especially in regards to collaborative activities. All velocities must be equal to zero before any transition can be initiated in the automaton. This is a condition that arises from the turn-based nature imposed on the join hybrid automaton. Guards for both vehicles are parameterized by values of their corresponding navigation function at their current Cartesian position as compared to boundary constants with $\tau \to 1$ and $\epsilon \to 0$. Additionally, the UGV guards are parameterized by function $U(x,y)$ which acts as a measure of actuator authority for the vehicle. This function contains UGV mass m, the gravitational vector $\vec{g}$ and the gradient of the navigation function $\nabla\varphi$:

$$\left\| m\,\vec{g} \times \frac{\nabla\varphi}{\|\nabla\varphi\|} \right\| \triangleq U(x,y) < u_{\max}$$

Depending on the value of $U(x,y)$ at a particular location, the UGV will either be able to move along the workspace boundary using its primary locomotion or it must use the tether winch to pull itself upwards. In the guards for all movement actions excluding f', the action can only be initiated when when the UGV is in contact with the workspace boundary so that $\varphi_a(x,y) < \tau$. For all actions which require the tether, the UAV must be perched at the anchoring point so that $q_a = q_a^f$. For landing action β to be enabled, the UAV must be away for the workspace boundary so that $\varphi_a(x,y) < \tau$. The opposite is true for latching action γ' where the UAV always starts on the boundary with $\varphi_a(x,y) \geq \tau$. Additionally, for landing action β to occur, the UAV should be aligned with the x-position of the UGV x_g with this position calculated from the communication variables as $x_g = q_h^x + l\cos\theta$. Finally, the value of the UAV and UGV navigation function must be less than ϵ in order to reach the terminal location via $\ltimes$. Thus the set of guards is written as:

$$\begin{aligned}
G_\delta &= \{(x,y,\phi) \in X_a \mid \dot{x} = \dot{y} = \dot{\phi} = 0\} \\
G_\beta &= \left\{(x,y,\phi) \in X_a \mid \dot{x} = \dot{y} = \dot{\phi} = 0,\ \varphi_a(x,y) < \tau, \|x - x_g\| < \epsilon\right\} \\
G_{\gamma'} &= \{(x,y,\phi) \in X_a \mid \dot{x} = \dot{y} = \dot{\phi} = 0,\ \varphi_a(x,y) \geq \tau\} \\
G_a &= \left\{(l,\theta) \in X_g \mid \dot{l} = \dot{\theta} = 0,\ \varphi_g\left(x_g,y_g\right) \geq \tau,\ |u| < U\left(x_g,y_g\right)\right\} \\
G_{a'} &= \left\{(l,\theta) \in X_g \mid \dot{l} = \dot{\theta} = 0,\ \varphi_g\left(x_g,y_g\right) \geq \tau,\ |u| \right. \\
&\quad \left. < U\left(x_g,y_g\right),\ q_a = q_a^f\right\} \\
G_d &= \left\{(l,\theta) \in X_g \mid \dot{l} = \dot{\theta} = 0,\ \varphi_g\left(x_g,y_g\right) \geq \tau,\ |u| < U\left(x_g,y_g\right),\right. \\
&\quad \left. \|x - l\cos\theta\| < \epsilon,\ \|y - l\sin\theta\| < \epsilon\right\} \\
G_{r'} &= \left\{(l,\theta) \in X_g \mid \dot{l} = \dot{\theta} = 0,\ \varphi_g\left(x_g,y_g\right)\right. \\
&\quad \left. \geq \tau,\ |u| \geq U\left(x_g,y_g\right),\ q_a = q_a^f\right\} \\
G_{f'} &= \left\{(l,\theta) \in X_g \mid \dot{l} = \dot{\theta} = 0,\ |u| \geq U\left(x_g,y_g\right),\ q_a = q_a^f\right\} \\
G_\ltimes &= \left\{(x,y,\phi) \in X_a, (l,\theta) \in X_g \mid \dot{x}\right. \\
&\quad = \dot{y} = \dot{\phi} = \dot{l} = \dot{\theta} = 0, x = x^f, y = y^f, \\
&\quad \left. \varphi_a(x,y) < (1-\tau), \varphi_g\left(x_g,y_g\right) < \epsilon\right\}
\end{aligned}$$

For the tethered case study, the reset conditions which form the set of jumps takes on two forms. All UAV transitions and terminating transitions admit the trivial reset where none of the state variables change in value. However, for most of the UGV transitions, the resets are utilized to reparameterize the UGV state varables according to the currently active hinge point. More details of these hinge points are covered in Section 5.1. Assume that the currently active hinge point is q_h^c and the current state of the UGV is denoted by the tuple (l_c,θ_c). Any transition which invokes a change in the active hinge point will also result in a jump in the value of the state variables. For actions involving the tether, the hinge points denote the points in the workspace from which the tether will appear to be suspended. Actions such as a' admit a virtual hinge point where the transition to the next action occurs and according to which the dynamics are parameterized. Similarly, for action d, the hinge points will become the location of the UAV which has landed in preparation for latching where the UGV must attach the tether. If the next point in the ordered sequence of hinge points is denoted $q_h^{c+1} = (x_h^{c+1}, y_h^{c+1})$ and the current UGV location is denoted $q_c = (l_c\sin\theta_c, -l_c\cos\theta_c)$, then the length l jumps to the Euclidean distance $d(q_h^{c+1}, q_c)$ between the UGV and the new hinge point. These same variables are also utilized to reset the angle θ relative to the new hinge point location. The aforementioned jump conditions for each of the location pairs can then be written as:

$$J_{(1,9)}: \begin{cases} x \mapsto \{x_c\} \\ y \mapsto \{y_c\} \\ \phi \mapsto \{\phi_c\} \end{cases} \qquad J_{(1,2)}: \begin{cases} q_h \mapsto q_h^{c+1} \\ l \mapsto \left\{d\left(q_h^{c+1}, q_c\right)\right\} \\ \theta \mapsto \left\{\arctan\left(\dfrac{l_c \sin\theta_c - y_h^{c+1}}{l_c \cos\theta_c - x_h^{c+1}}\right)\right\} \end{cases}$$

$$J_{(2,8)}: \begin{cases} q_h \mapsto \{q_h^c\} \\ l \mapsto \{l_c\} \\ \theta \mapsto \{\theta_c\}, \end{cases}$$

$$\begin{aligned} &J_{(1,9)} = J_{(9,3)} = J_{(3,9)} = J_{(4,5)} = J_{(3,8)} = J_{(5,8)} \\ &J_{(1,2)} = J_{(3,2)} = J_{(3,4)} = J_{(5,6)} = J_{(5,7)} = J_{(5,10)} = J_{(6,10)} \\ &\qquad = J_{(6,7)} = J_{(7,10)} = J_{(7,6)} = J_{(7,7)} = J_{(10,7)} = J_{(10,6)} \\ &J_{(2,8)} = J_{(6,8)} = J_{(7,8)} = J_{(10,8)} \end{aligned}$$

The set of invariant spaces for each location Inv is constructed in a similar fashion to the set of guards. In the initial state Location 1, the system is at rest and all state variables are given an initial value which parameterizes the initial positions of the two vehicles. Similarly, the system will remain at rest in the goal state marked by Location 8, the UGV will be at its goal coordinates and the value of the navigation function for both agents will be close to zero so that $\varphi(x, y) < \epsilon$. The UGV axis is locked in Locations 4, 6, and 10, and it will not undergo angular motion so that $\dot{\theta} = 0$. For Location 7, there is navigation function constraint $\varphi_g(x_g, y_g) < \tau$ which ensures that the UGV only remains in this location until contact is made with a workspace boundary. In all locations where the UGV is in motion, a transition and associated reset on the continuous state must occur if $l \leq 0$. The UAV will only undergo vertical motion in Location 3 so the value of x remains constant and ϕ will always equal zero. Additionally, Location 3 and Location 9 share the condition that the UAV only remains in that location while it is away from the boundary of the workspace so that $\varphi_a(x, y) < \tau$. The same stipulation holds for Location 5 with the added condition that the UAV has not reached the goal location so that $\varphi_a(x, y) \geq \epsilon$. Writing these conditions into a set of mathematical constraints, the invariant set is written as:

$$\begin{aligned} \mathrm{Inv}_{(1)} &= \{(x, y, \phi) \in X_a, (l, \theta) \in X_g \mid \dot{x} = \dot{y} = \dot{\phi} = \dot{l} = \dot{\theta} = 0, \\ &\qquad x = x^i, y = y^i, l = l^i, \phi = 0, \theta = 0\} \\ \mathrm{Inv}_{(2)} &= \left\{(l, \theta) \in X_g \mid \varphi_g\left(x_g, y_g\right) \geq \epsilon,\ |u^g| < U\left(x_g, y_g\right), l > 0\right\} \\ \mathrm{Inv}_{(3)} &= \{(x, y, \phi) \in X_a \mid \dot{x} = \phi = 0,\ \varphi_a(x, y) < \tau\} \\ \mathrm{Inv}_{(4)} &= \left\{(l, \theta) \in X_g \mid \dot{\theta} = 0,\ \varphi_g\left(x_g, y_g\right) \geq \tau,\ l > 0\right\} \\ \mathrm{Inv}_{(5)} &= \{(x, y, \phi) \in X_a \mid \epsilon \leq \varphi_a(x, y) < \tau\} \\ \mathrm{Inv}_{(6)} &= \left\{(l, \theta) \in X_g \mid \dot{\theta} = 0,\ |u^g| > U\left(x_g, y_g\right), l > 0\right\} \\ \mathrm{Inv}_{(7)} &= \left\{(l, \theta) \in X_g \mid \varphi_g\left(x_g, y_g\right) < \tau,\ |u^g| \geq U\left(x_g, y_g\right), l > 0\right\} \\ \mathrm{Inv}_{(8)} &= \{(x, y, \phi) \in X_a, (l, \theta) \in X_g \mid \dot{x} = \dot{y} = \dot{\phi} = \dot{l} = \dot{\theta} = 0, \\ &\qquad \varphi_a(x, y) < \epsilon, \varphi_g\left(x_g, y_g\right) < \epsilon, x_g = x_g^f, y_g = y_g^f\} \\ \mathrm{Inv}_{(9)} &= \{(x, y, \phi) \in X_a \mid \varphi_a(x, y) < \tau\} \\ \mathrm{Inv}_{(10)} &= \{(l, \theta) \in X_g \mid \dot{\theta} = 0,\ \varphi_g\left(x_g, y_g\right) \\ &\qquad \geq \tau,\ |u^g| < U\left(x_g, y_g\right), l > 0\} \end{aligned}$$

The continuous dynamics (vector fields), and their assignment to corresponding locations (with reference to Figure 5) can be succinctly described as follows:

$$\mathrm{Act}(2) = \begin{cases} \ddot{l} = l\dot{\theta}^2 - g\cos(\theta) + \dfrac{u}{m} \\ \ddot{\theta} = -\dfrac{2\dot{l}\dot{\theta}}{l} + \dfrac{g\sin\theta}{l} + N \end{cases}$$

$$\mathrm{Act}(5,9) = \begin{cases} \ddot{x} = -\dfrac{u_1}{m}\sin\phi \\ \ddot{y} = \dfrac{u_1}{m}\cos\phi - g \\ \ddot{\phi} = u_2 \end{cases} \qquad \mathrm{Act}(3) = \begin{cases} \ddot{x} = \ddot{\phi} = 0 \\ \ddot{y} = \dfrac{u_1}{m}\cos\phi - g \end{cases}$$

$$\mathrm{Act}(6) = \begin{cases} \ddot{l} = -g\cos\theta + T - \dot{l}\delta(t) + \dfrac{u}{m} \\ \ddot{\theta} = -\dfrac{g}{l}\sin\theta - \dfrac{2\dot{l}\dot{\theta}}{l} - l\dot{\theta}\delta(t) \end{cases}$$

$$\mathrm{Act}(4) = \begin{cases} \ddot{x} = \ddot{y} = \ddot{\phi} = \ddot{\theta} = 0 \\ \ddot{l} = \dfrac{u}{m} \end{cases} \qquad \mathrm{Act}(10) = \begin{cases} \ddot{l} = \dfrac{u}{m} \\ \ddot{\theta} = 0 \end{cases}$$

$$\mathrm{Act}(1,8) = \{\ddot{x} = \ddot{y} = \ddot{\phi} = \ddot{l} = \ddot{\theta} = 0.$$

In these equations N denotes the ground reaction force to the UGV, T is the tether tension, m is the UGV mass and g is the constant of gravitational acceleration. Location 2 corresponds to the UGV moving horizontally under traction force u and also allows for the robot to free-fall after driving off the edge of a surface until it impacts another workspace boundary. Similarly, Location 10 corresponds to the UGV moving horizontally under traction force u while tethered—the UGV is unable to free-fall assuming there is no excessive slack in the tether. Location 4 has the same dynamics for the UGV but in addition has the UAV stationary which is necessary during the process of UGV latching its tether to the UAV. Location 6 involves vertical displacement with horizontal swinging and also includes the possibility of collisions with workspace boundaries. Location 3 corresponds to UAV vertical motion (take-off/landing), while Locations 5 and 9 describe general UAV flight between waypoints and the perching of the UAV on a surface. The latter two locations have identical underlying dynamics and are only differentiated based on the action's end objective. Finally, Locations 1 and 8 mark the initial and final state of the system where both vehicles are stationary. Note that when the UGV swings suspended by the tether, its motion can be modeled as that of a pendulum with varying length and damping terms that capture the effect of possible inelastic collisions with workspace boundary, the latter activated through the action of a Dirac function $\delta(t)$.

The concrete dynamics control loops are closed as follows. For the movement of the UGV toward a selected location (hinge point), driving the input force u via a simple PD on the error on state l typically suffices. The control of the UAV is realized via a differential flatness trajectory tracking control law, on UAV inputs u_1 and u_2. For this trajectory tracking law, the collision-free (x, y) reference trajectory in the differentially flat space is constructed via the diffeomorphic transformation referenced in the following subsection.

5.1 Determining hinge points

The UGV dynamics are parameterized by tether *hinge points*, the location of which is critical for determining what type of motion

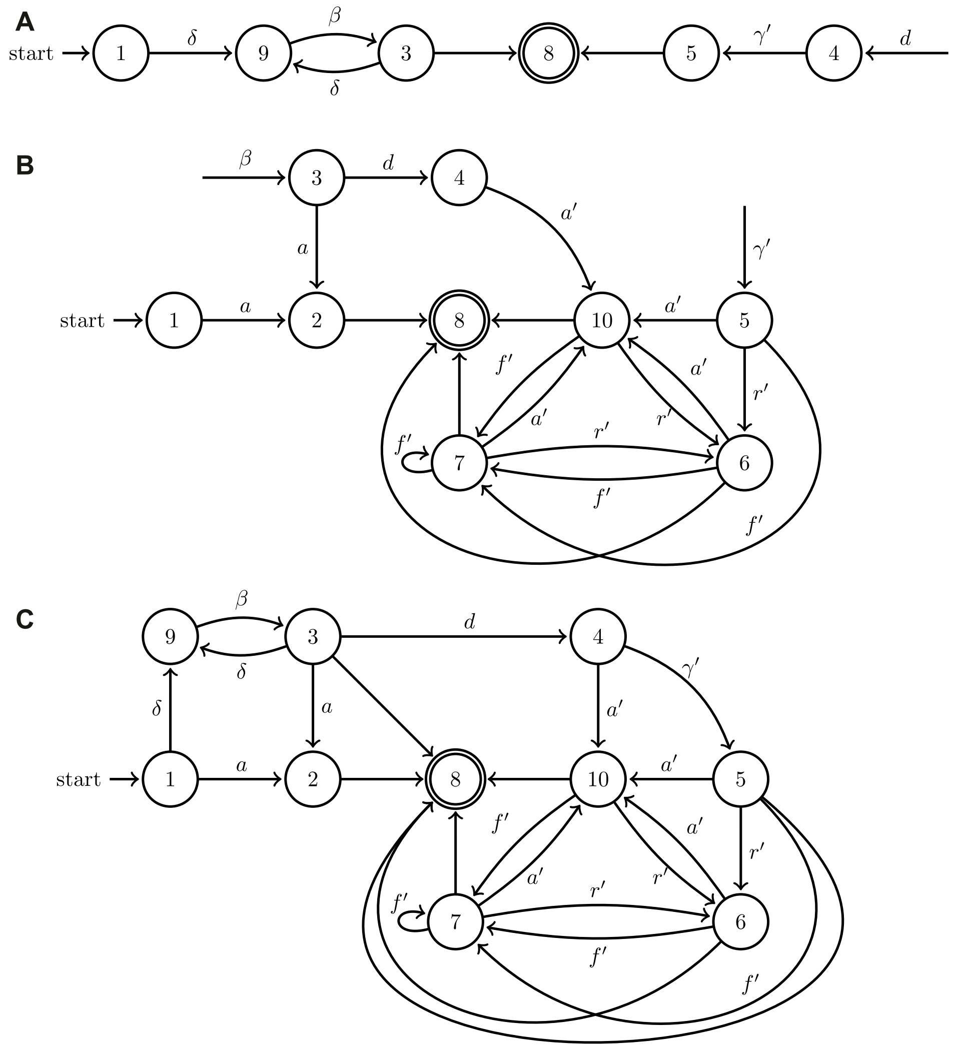


FIGURE 5
A minimal example of the application of the join operation to the directed graphs associated to the two hybrid automata modeling the UAV **(A)** and UGV **(B)** systems depicted in Figure 1, as well as the outcome of their join operation **(C)**.

the UGV undergoes. Hinge points denote locations on the edges of obstacles from which the ground vehicle can swing in a quasi-pendulum motion. Where these hinge points are placed depends on how the tether has been threaded through the workspace and how it deforms under the tension applied by the mass of the suspended UGV. Whether the tether can provide the necessary tension to support the weight of the vehicle, in turn, depends on its shape and relative configuration with respect to the workspace boundaries. While determining the precise configuration of a highly deformable, distributed parameter system such as a cable, chain, or tether unnecessarily complicates the planning problem and falls beyond the scope of this paper, an approximate yet realistic representation of possible tether configurations as a function of the motion history of the tethered vehicles is still necessary. One method of tether deformation approximation has been described in more detail by Sebok and Tanner (2019).

There are, in fact, infinitely many conceivable configurations for a flexible tether to weave among an arrangement of static obstacles. Yet, this multitude of configurations falls into a finite set of relevant *homotopy* classes (Bhattacharya et al., 2010; Kim et al., 2014), depending on the workspace geometry. Paths in each homotopy class can be diffeomorphically mapped onto each other. Our planning algorithm for this case study systematically enumerates those homotopy classes, and utilizes an iterative algorithm to select a valid tether path primitive (class representative) while eliminating, for instance, extensively long paths and those that tangle or wrap the tether around obstacles.

The collection of hinge point locations forms a set Q_h. In order to determine the location of these hinge points, the tether primitive identified for each homotopy class is deformed so that it tightly conforms to the edges of the obstacles in the workspace. A Bezier spline curve is then constructed using the points of the tether primitive (Figure 6). This 2D B-spline curve can be expressed in terms of its two Cartesian components $x_c(t)$ and $y_c(t)$, parameterized by $t \in [0,1]$. Here we assume that the (UAV) end of the tether has been anchored at point (x_a, y_a). This location is contained in the set of hinge points Q_h for every possible tether configuration. The remaining hinge points are situated where the

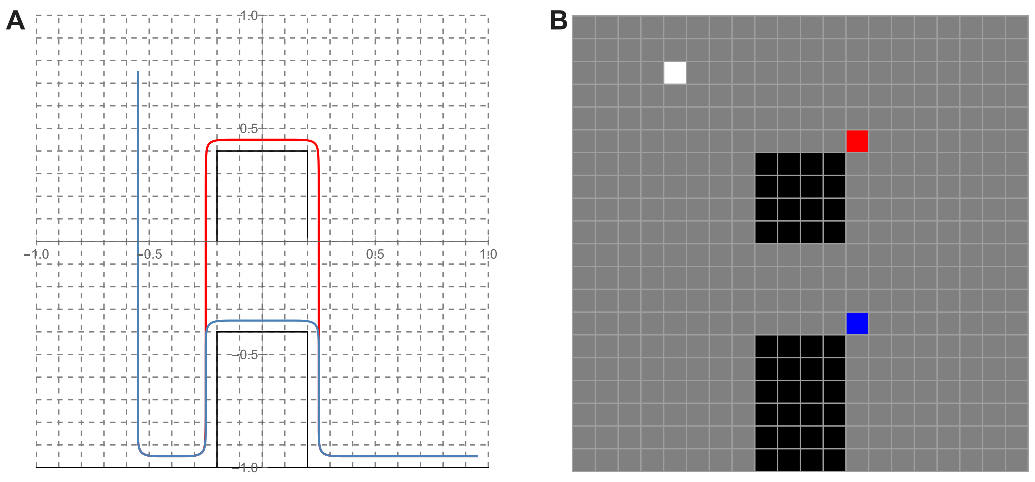


FIGURE 6
(A) Two representatives (red and blue) of homotopy tether curves realized as Bezier curves used as primitives; **(B)** The (color coded) hinge points associated with each of the tether primitives and the white cell marking the anchor point common to both.

tether is suspended as determined by the geometry of an obstacle, and these locations can be identified as the points of maximum curvature along the deformed tether curve. Hinge points are thus those pairs $(x_h(t), y_h(t))$ corresponding to values of $t \in [0,1]$ that locally satisfy.

$$\max_{t\in[0,1]} \frac{\mathrm{d}^2 x_c(t)}{\mathrm{d}t^2} \cdot \frac{\mathrm{d}^2 y_c(t)}{\mathrm{d}t^2} \quad \text{subject to} \quad \begin{cases} \dfrac{\mathrm{d}^2 x_c(t)}{\mathrm{d}t^2} < 0 > \dfrac{\mathrm{d}^2 y_c(t)}{\mathrm{d}t^2}, & x_h > x_a \\ \dfrac{\mathrm{d}^2 x_c(t)}{\mathrm{d}t^2} > 0 > \dfrac{\mathrm{d}^2 y_c(t)}{\mathrm{d}t^2}, & x_h < x_a. \end{cases}$$

The collection of all the pairs $(x_h(t), y_h(t))$ for the locally maximizing values of $t \in [0,1]$ forms a finite set $Q_t \subset Q_h$. Including the anchor point, the complete set of hinge points for a given homotopy class is written as $Q_h = Q_t \cup \{(x_a, y_a)\}$.

5.2 Application: search for an action plan

The planner that searches for an (appropriately parameterized) turn-based sequence of labels that mark an action plan for the robots to take to complete the assigned task. This sequence consists of licensed transitions between supercells of the discretized workspace of the join hybrid system. An analysis of the guards and invariants of the join hybrid automaton for this system leads to the definition of 13 different supercells (out of 400 individual cells—see Figure 6), with each supercell associated to a unique system activity, and 19 possible transitions between those supercells, as licensed by the join hybrid automaton.

The planner weighs the different options for possible transitions out of a given supercell based on cost functions evaluated on continuous state space variables of the join hybrid system. To facilitate computation, this continuous domain is quantized using the centroids of the discretized workspace cells. Several possibilities exist for cost functions to guide the planner's search; a good trade-off between greedy and longer-range planning is offered by a combination of a short-term with a long-term cost function. To see how the planner's assessment is carried out in this particular case study, assume that the system is currently at cell ℓ_i and considers transitioning to some cell ℓ_j which belongs to a different supercell than the one it is currently in. Now assume that navigation function φ is available on the continuous state space of the join hybrid automaton, assuming its minimum at the region where the system task specification is satisfied, and being uniformly maximum at the system's workspace boundary. With some abuse of notation, we will write $\varphi(\ell_i)$ to express the value of the navigation function at the centroid of the transition cell ℓ_i that the system currently occupies, and similarly denote $\varphi(\ell_j)$ as the corresponding value at the cell currently being assessed. (Note that $\varphi(\ell_f) \equiv 0$) With $D(\ell_i, \ell_j)$ denoting the Euclidean distance between the centroids of cell i and j, the local and future cost functions that quantify the cost-benefit ratio of a transition between 2 cells are given as

$$a_{ij} = \frac{D(\ell_i, \ell_j)}{|\varphi(\ell_i) - \varphi(\ell_j)|} \qquad h_j = \frac{D(\ell_j, \ell_f)}{\varphi(\ell_j)},$$

respectively. In this particular implementation, the action-specific cost of initiating UAV motion is set at a value higher than the maximum cost of a UGV action, to reflect the fact that UAV motion is energetically way more "expensive" compared to UGV motion. This setting practically ensures that the UAV will be deployed only if the UGV cannot move where it needs to go on its own.

After comparing label sequences, the planner settles on $\delta\, \beta\, d\, \gamma'\, f'\, f'$. This plan involves the activity where the UAV reaches a waypoint (via δ) and subsequently lands (via β) at a designated rendezvous location for tether latching; the UGV subsequently moves in to latch its tether on the UAV (via d); the UAV then lifts off and goes to perch at a designated elevated spot (via γ'); the UGV ascends to the first hinge point by reeling in the tether and swinging until it impacts the wall (via f'); and finally, the UGV ascends to the anchoring location and reels out the tether to descend towards the goal location (via f'). The reason why f' appears repeated is because the associated activity is parameterized differently each time: in the

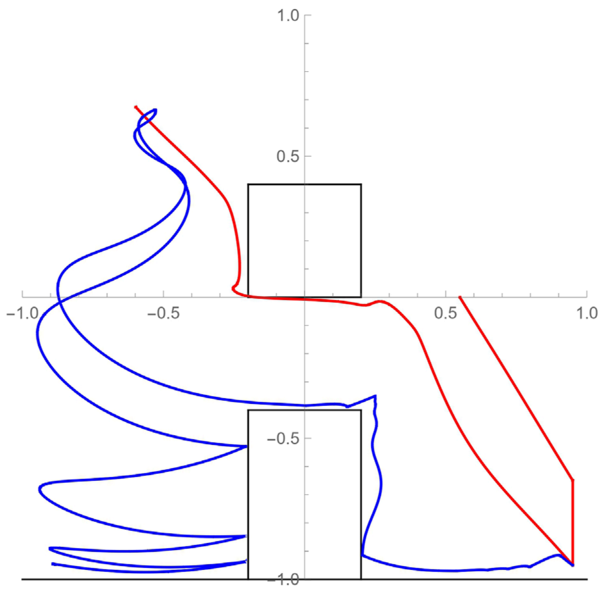


FIGURE 7
Simulated paths of the UAV (red) and UGV (blue) motion through the constrained workspace as they implement the cooperative action plan. The motion path of the UAV from an initial hovering position at (0.55, 0) to its perching configuration (−0.55, 0.75) consists of three distinct segments: reaching waypoint (0.95, −0.65) directly above designated landing spot; landing; and tracking a reference trajectory to the perching location avoiding the overhead obstacle. The motion path of the UGV consists of the following segments: swinging motion to the left from initial location (0.95, −0.95) until initial impact at the foot of the vertical wall after the UAV has perched; tethered-assisted ascent on the side of the wall; a swinging ascending motion from the top of wall to the UAV's perching point; and finally, a swinging descent to the desired position with periodic collisions on the left side of the wall.

former case, the hinge point is at the upper rightmost corner of the vertical wall obstacle while the UGV ascends, whereas in the latter case the hinge point switches to the perched position of the UAV and the UGV performs controlled ascent/descent while suspended from the UAV's perched position. This is an specific example of why a naive, purely discrete search over the join hybrid automaton graph representation is very likely to miss the desired solutions. A numerical implementation of this plan is showcased in Figure 7. The motion paths of the two vehicles are color coded.

5.3 Extension to a heterogeneous system with three agents

In principle, and absent computational considerations, the proposed methodology is subject to no limitation on the number of collaborative agents. In fact, it is modular in the sense that new agents can be added subsequently. The planning methodology can leverage the added capability as long as the existing agents are modeled to interact with the new modalities. To illustrate this, and without explicit analytical description for the sake of brevity, consider the above UAV/UGV system with an additional agent consisting of a common robotic manipulator. This robotic manipulator is located near the goal position for the UGV and can grasp some object located to the goal location and deposit it into a container on the UGV.

The robotic manipulator label set A_r would consist of two actions: ζ represents picking up an object and η represents placing an object in a new location. In this new system, the UAV automaton from Figure 5A remains unchanged. The UGV is updated as shown in Figure 8A to reflect that action ζ can occur after UGV action a'. Taking the *join* of the hybrid automaton graph for the robotic manipulator (Figure 8B) with the automata for the UAV and UGV, the join hybrid automaton is produced with the discrete components shown in Figure 8C. This join hybrid automaton for the extended three agent system contains both the individual functionality of the three agents as well as their collaborative possibilities.

5.4 Comparison to possible alternative methodologies

A couple of alternative methodologies were also employed to solve the cooperative planning problem of this section, neither of which produced a valid solution to the planning problem. The first one is based on a hybrid system optimal control design framework, available for systems with dynamics in polynomial form (Zhao et al., 2020). The second was a more general multi-robot motion and task planner called GRSTAPS (Messing et al., 2022). These two solvers represent approaches at the two ends of the methodological spectrum, with the optimal control formulation putting more emphasis on the continuous dynamics, and GRSTAPS primarily leveraging abstraction and discretization to scale up.

It was found that both approaches had difficulty addressing this problem, with challenges seeming to stem from dynamics nonlinearities. The major limitation of the hybrid optimal control approach is that it can only generate solutions for hybrid systems containing polynomial dynamics. This becomes a critical issue when attempting to generate meaningful planning comparisons for the UAV/UGV system as the nonlinear dynamics of this system are not adequately represented in polynomial form approximations. This resulted in this planning approach only being able to generate plans for a simplified system utilizing only the UGV. The hybrid optimal control approach was incapable of producing an action sequence and control outputs when the specified goal required use of the swinging action f'.

On the other hand, GRSTAPS required a particularly fine workspace discretization to be able to reasonably keep track of the nontrivial nonlinear system dynamics. The potential upside of this planner is its ability to efficiently perform task planning, scheduling, and motion planning. However, the primarily discrete nature of this algorithm limits the planner's ability to account for the dynamic restrictions which are essential to constructing a feasible action sequence.

In conclusion, none of the aforementioned alternative methodologies were capable of leveraging the unique new behaviors that are made possible through physical interaction amongst the component systems when only presented with the individual robot dynamics (as in Figure 3). In fact, even when the outcome of the join operation was *fed directly* as input to these planners, they still struggled: the optimal control algorithm could not produce meaningful sequences of inputs for the UAV and UGV while GRSTAPS

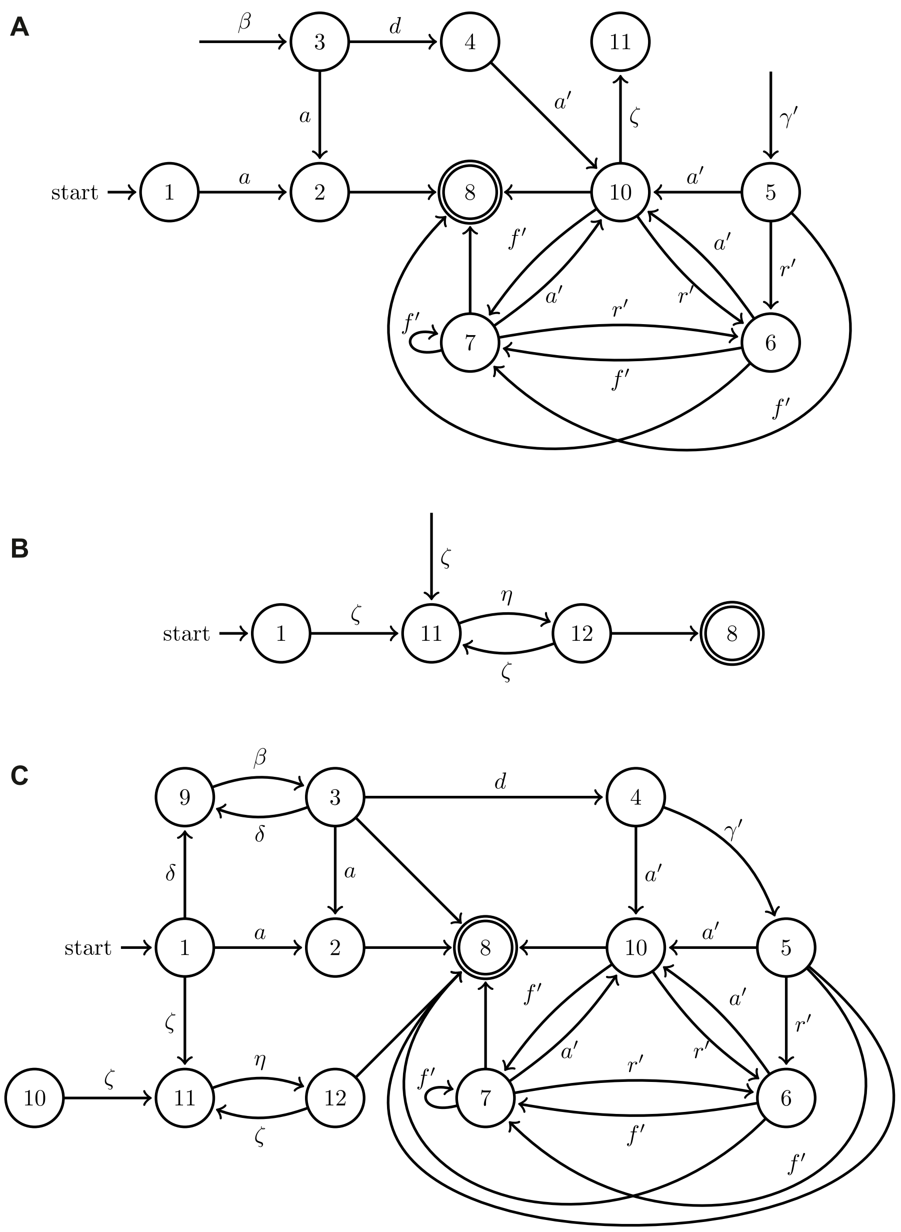


FIGURE 8
An example of the extension of the join operation to the directed graphs associated with the three hybrid automata for an updated model of the UAV **(A)**, a model of a robotic manipulator **(B)**, and UGV model (Figure 5A), as well as the outcome of their join operation **(C)**.

only produced valid cooperative plans when, *in addition* to the join automaton, it was also explicitly presented with the same workspace partition and discretization utilized above. The poor performance of these two planners on the UAV/UGV system meant that it was impossible to create a meaningful comparison between the existing planners and the methodology outlined within this work.

6 Case study 2: the dual UGV system

The previous section illustrated in some detail how the proposed planning methodology can be applied to coordinate a heterogeneous robot team comprised of a UAV that can tether itself to a UGV. This section aims to reinforce the point that the presented methods are not tailored to a particular multi-robot system or a mechanism of physical interaction between agents. While it is true that it may not always be possible to formulate any arbitrary multi-robot planning problem into one that fits the proposed framework, in principle the methodology is applicable to a reasonably wide class of small-scale heterogeneous multi-robot systems.

To this end, this section considers a different heterogeneous multi-robot configuration consisting of two UGVs. Here the UGVs are heterogeneous due their different locomotion modalities and motion degrees of freedom. The first robot (Figure 9A; Figure 10) is a (nonholonomic) wheeled robot with a differential drive. The second robot (Figure 9B) is a walking mechanism that utilizes a

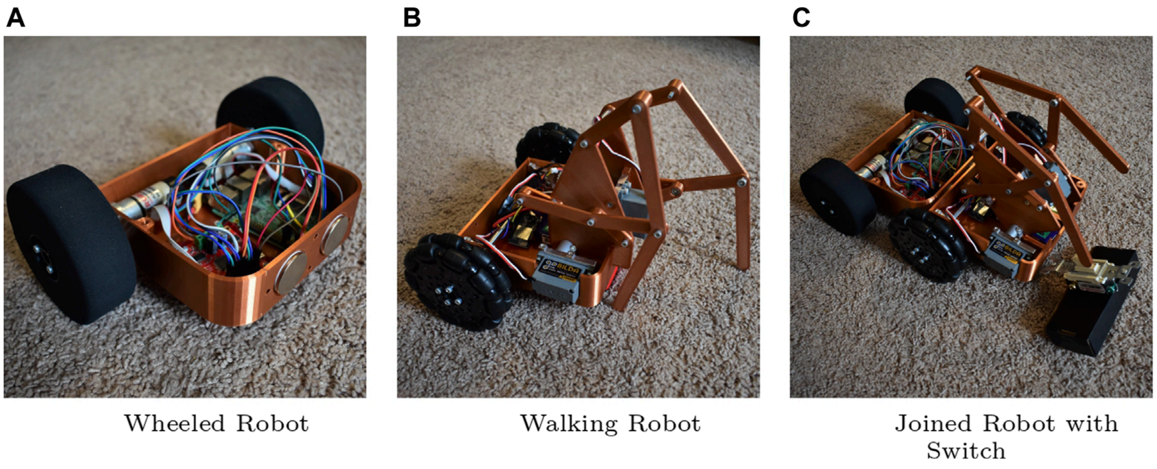


FIGURE 9
Another heterogeneous system consisting of **(A)** a wheeled robot and **(B)** a walking robot. Using magnets embedded in the robot frames, the vehicles can join together to interact with objects such as a light switch **(C)**. A video demonstrating a sequence involving the two UGVs joining and then moving together can be found at: http://research.me.udel.edu/ btanner/videos.html.

Klann linkage mechanism to locomote, pulling two unactuated wheels in the back for balance. This second robot has only one degree of freedom (it cannot turn); it does, however, have a second actuator capable of tilting its legs along a vertical plane when somehow supported on its other side. This external support can be realized by magnets located on the front of the rolling robot and the rear of the walking robot. Note that neither of the agents individually can manipulate the environment, but when connected by means of the magnets (Figure 9C), the raised legs become end effectors and the contraption takes the form of a basic two degree of freedom (DOF) manipulator. This allows the joined robot to perform other tasks such as flipping a switch or pressing a button.

The planning approach of the previous sections (without the tether analysis) can be applied to this system too. For this case, assume that the task at hand is to have the robots flip a switch like the one shown in Figure 9. Here, the planning algorithm coordinates the two robots to join and form the mobile manipulator, and then steers it toward its final objective. The details of the computational implementation are omitted since the smaller number of degrees of freedom in this problem and the simplicity of the individual robot kinematics render the problem relatively straightforward compared to the case study of Section 5; the hybrid automata models and synthesis are however described at a similar level of detail as in the UAV–UGV case in the following section.

6.1 Hybrid automata models

The walking UGV has a label set $A_w = \{a, c, i, j\}$ where a expresses the basic one-dimensional walking motion, c is the connection action where the walking UGV joins itself to the wheeled UGV using the magnetic connection, i represents the interaction of the joined vehicle with the environment and j expresses the two-dimensional motion of the joined vehicle in the horizontal plane. Similarly, the wheeled UGV has label set $A_r = \{r, p, i, j\}$ with r being the basic two-dimensional motion action in the horizontal plane, p being an action which positions the wheeled UGV behind the walking UGV in prepartion for connection and i, j being the same as in the walking UGV. Note that i and j are labels for both component hybrid systems; however, the associated activities can only be triggered after the individual robots have joined into a single vehicle.

The continuous dynamics of this dual UGV system are defined within a 5th dimensional state space X composed of tuples of the form $\{\phi_w, \theta_w, x_r, y_r, \theta_r\}$. The Cartesian coordinate system is established so that that that the origin is positioned at the initial position of the walking UGV and the x-axis is aligned with its orientation. Thus, the wheeled UGV's initial position is parameterized relative to the initial position of the walking UGV. In this case, (x_r, y_r, θ_r) parameterize the $SE(2)$ pose of the wheeled UGV on the plane of motion while the pair (ϕ_w, θ_w) parameterize both the Cartesian position of the walking UGV along the x-axis as well as the rotational position and inclination of the 2 DOF manipulator when the agents are joined together. Specifically, ϕ_w tracks the net rotations of the Klann linkage while θ_w provides its inclination angle relative to the ground. When the robots are detached and the walking motion is enabled, θ_w will remain constant so that $\theta_w = 0$.

For the dual UGV system, the set W of continuous communication variables $\{q_i, u_1^w, u_2^w, u_1^r, u_2^r\}$. comprised of walking UGV control inputs u_1^w and u_2^w, wheeled UGV control inputs u_1^r and u_2^r, and the (piecewise constant) location q_i of the current position within the plane where the joined robot must interact with its environment.

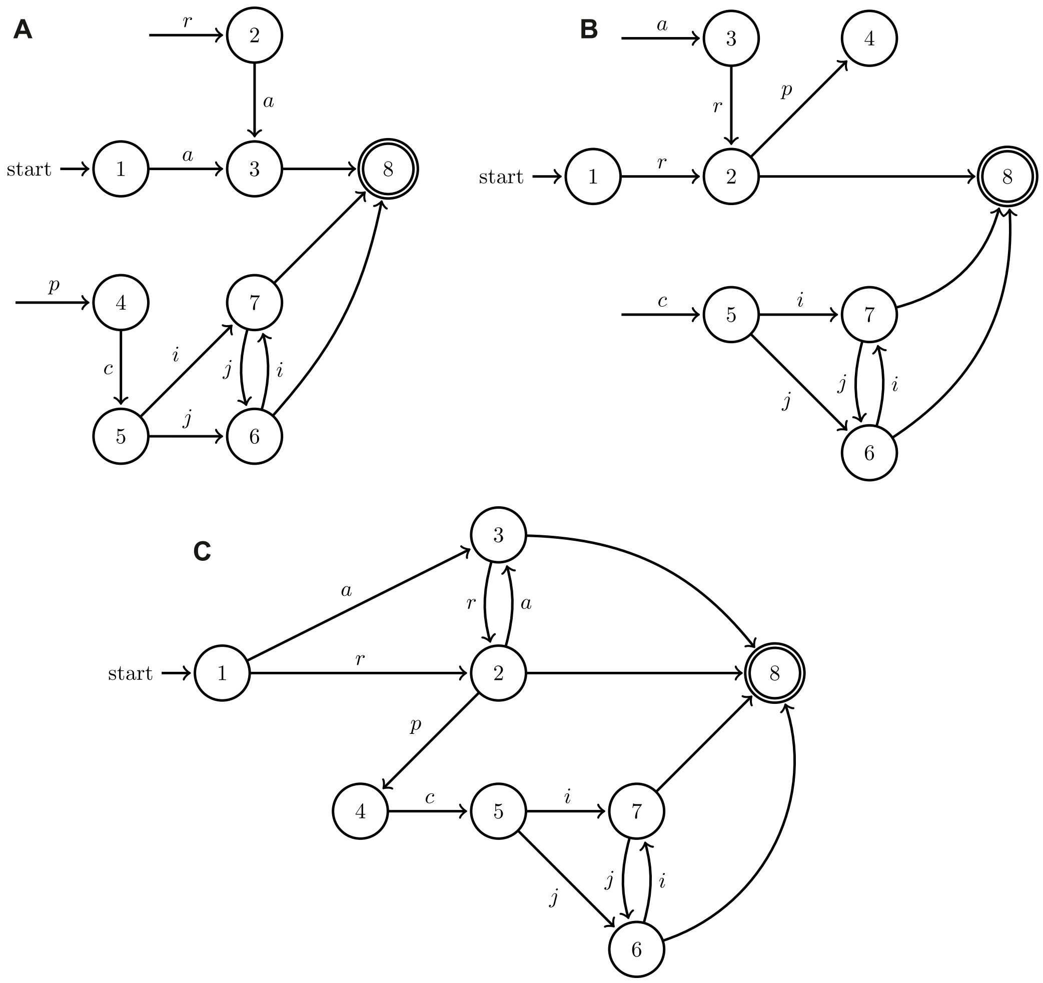


FIGURE 10
A minimal example of the application of the join operation to the directed graphs representing two hybrid automata with the walking UGV **(A)**, wheeled UGV **(B)** of Figure 9 as well as the outcome of their join operation **(C)**.

The event set E representing all transitions in the graph of the join automaton can be written as:

$$
\begin{array}{ll}
E_1 = \langle 1, a, G_a, J_{(1,3)}, 3\rangle & E_2 = \langle 2, a, G_a, J_{(2,3)}, 3\rangle \\
E_3 = \langle 4, c, G_c, J_{(4,5)}, 5\rangle & E_4 = \langle 5, j, G_j, J_{(5,6)}, 6\rangle \\
E_5 = \langle 5, i, G_i, J_{(5,7)}, 7\rangle & E_6 = \langle 7, j, G_j, J_{(7,6)}, 6\rangle \\
E_7 = \langle 6, i, G_i, J_{(6,7)}, 7\rangle & E_8 = \langle 1, r, G_r, J_{(1,2)}, 2\rangle \\
E_9 = \langle 3, r, G_r, J_{(3,2)}, 2\rangle & E_{10} = \langle 2, p, G_p, J_{(2,4)}, 4\rangle \\
E_{11} = \langle 3, \ltimes, G_\ltimes, J_{(3,8)}, 8\rangle & E_{12} = \langle 6, \ltimes, G_\ltimes, J_{(6,8)}, 8\rangle \\
E_{13} = \langle 7, \ltimes, G_\ltimes, J_{(7,8)}, 8\rangle & E_{14} = \langle 2, \ltimes, G_\ltimes, J_{(2,8)}, 8\rangle
\end{array}
$$

Again, the key constructions within the set of events are the set of guards and resets (jumps) for each transition. Under the assumption that the system will reach a steady state before the next action, the Klann mechanism must be at rest for an action to be triggered which is written $\dot{\phi}_w = 0$ for the guard of each action. This assumption also implies that all velocities must equal zero for the wheeled UGV before any transition can occur. For actions that require the Klann linkage for locomotion, the mechanism must be in contact with the ground so that $\theta_w = 0$. The position of the walking UGV along the x-axis can be determined by the conversion $x_w = s\frac{\phi_w}{\pi}$ where s is the stride length of the Klann linkage. This conversion allows for comparisons to determine where the walking UGV is positioned to the wheeled UGV. For movement action a, the walking UGV must not be in the same location as the wheeled UGV. For the positioning action p to occur, $y_r = 0$ to ensure that the robot is along the horizontal x-axis where the walking UGV is positioned. For connection action c to occur, the wheeled UGV must be properly positioned on the x-axis and aligned with the walking UGV magnets. For joint actions i and j, the Klann linkage must be raised to not interfere with movement so that $\theta_w > 0$. Also, for the joined vehicle to interact with an object with i, the vehicle position (measured using the wheeled UGV coordinates q_r) must be sufficiently close to the interaction location q_i. Finally, the system must be at its goal position and at rest for the terminating transition $\ltimes$ to be enabled. With these conditions in mind, the guards of the join automaton are written as follows:

$$
\begin{aligned}
G_a &= \left\{(\phi_w, \theta_w) \in X_w \mid \dot{\phi}_w = \theta_w = 0, x_r \neq s\frac{\phi_w}{\pi} \parallel y_r \neq 0\right\}\\
G_c &= \left\{(\phi_w, \theta_w) \in X_w \mid \dot{\phi}_w = \theta_w = 0, \theta_r = 0, y_r = 0, x_r \neq s\frac{\phi_w}{\pi}\right\}\\
G_r &= \left\{(x_r, y_r, \theta_r) \in X_r \mid \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0\right\}\\
G_p &= \left\{(x_r, y_r, \theta_r) \in X_r \mid \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0, y_r = 0\right\}\\
G_i &= \left\{(x_r, y_r, \theta_r) \in X_r \mid \dot{\phi}_w = \theta_w = \dot{x}_r = \dot{y}_r = \dot{\theta}_r\right.\\
&\quad \left.= 0,\ \theta_w > 0,\ \|q_r - q_i\| < \epsilon\right\}\\
G_j &= \left\{(x_r, y_r, \theta_r) \in X_r \mid \dot{\phi}_w = \theta_w = \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0,\ \theta_w > 0\right\}\\
G_{\ltimes} &= \{(\phi_w, \theta_w) \in X_w, (x_r, y_r, \theta_r) \in X_r \mid \dot{\phi}_w\\
&\quad = \dot{\theta}_w = \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0,\\
&\quad x_r = x_r^f, y_r = y_r^f, \theta_r = \theta_r^f\}
\end{aligned}
$$

For the dual UGV case study, there are no resets on the state variables which allows every transition to admit the trivial jump condition:

$$
J_{(\ell_i, \ell_j)}: \begin{cases} \phi_w \mapsto \{\phi_{wc}\}\\ \theta_w \mapsto \{\theta_{wc}\}\\ x_r \mapsto \{x_{rc}\}\\ y_r \mapsto \{y_{rc}\}\\ \theta_r \mapsto \{\theta_{rc}\} \end{cases} \qquad \forall \ell_i, \ell_j \in L
$$

The set of invariant spaces for each location Inv(ℓ) is constructed in a similar fashion to the set of guards. While the guards parameterize conditions on the state and communication variables for a particular transition to occur, the invariant spaces are solely functions of the state variables and represent conditions for the automaton execution to remain within the location reached by its associated transition label. In the initial state Location 1, the system is at rest, all velocities are zero and walking UGV state variables are initialized to zero. The wheeled UGV state variables are initialized to some initial value relative to the initial location of the walking UGV. Additionally, the system will also remain at rest in the goal state marked by Location 8. While the wheeled UGVis within Location 2 and Location 6, the derivative of one of the state variables must be non-zero so that the vehicle remains in motion. Additionally for the wheeled UGV to remain in Location 6, the position of the vehicle must not be located at the desired interaction location. Within Location 4, the wheeled vehicle is only allowed to rotate so that it always has angular velocity but zero linear velocities. While the walking UGV is within Location 3, the inclination angle of the mechanism θ_w remains fixed at 0 and the angular velocity of the arm rotation must take on some non-zero value. In Locations 5 and 7, either the angular velocity of the linkage or of the mechanism rotation must maintain a non-zero value. Additionally, in Locations 6 and 7, the arm mechanism is lifted to accommodated the joined vehicle so that $\theta_w > 0$ within those locations. Finally, within Location 7, the joined vehicle is sufficiently close to the prescribed interaction location and velocities for the wheeled UGV variables are zero as all movement will be performed by the walking UGV portion of the vehicle. Taking all of these conditions into account, the invariant set is written as:

$$
\begin{aligned}
\mathrm{Inv}_{(1)} &= \{(\phi_w, \theta_w) \in X_w, (x_r, y_r, \theta_r) \in X_r \mid \dot{\phi}_w\\
&\quad = \dot{\theta}_w = \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0,\\
&\quad \phi_w = \theta_w = 0, x_r = x_r^i, y_r = y_r^i, \theta_r = \theta_r^i\}\\
\mathrm{Inv}_{(2)} &= \left\{(x_r, y_r, \theta_r) \in X_r \mid \dot{x}_r > 0 \parallel \dot{y}_r > 0 \parallel \dot{\theta}_r > 0\right\}\\
\mathrm{Inv}_{(3)} &= \{(\phi_w, \theta_w) \in X_w \mid \theta_w = 0, |\dot{\phi}_w| > 0\}\\
\mathrm{Inv}_{(4)} &= \left\{(x_r, y_r, \theta) \in X_r \mid \dot{x}_r = \dot{y}_r = 0,\ \dot{\theta}_r > 0\right\}\\
\mathrm{Inv}_{(5)} &= \left\{(\phi_w, \theta_w) \in X_w \mid |\dot{\phi}_w| > 0 \parallel |\dot{\theta}_w| > 0\right\}\\
\mathrm{Inv}_{(6)} &= \left\{(\phi_w, \theta_w) \in X_w, (x_r, y_r) \in X_r \mid \dot{\theta}_w = 0,\ \theta_w > 0,\right.\\
&\quad \left.\dot{x}_r > 0 \parallel \dot{y}_r > 0 \parallel \dot{\theta}_r > 0,\ q_r \neq q_i\right\}\\
\mathrm{Inv}_{(7)} &= \left\{(\phi_w, \theta_w) \in X_w, (x_r, y_r, \theta_r) \in X_r \mid \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0,\right.\\
&\quad \left.|\dot{\phi}_w| > 0 \parallel |\dot{\theta}_w| > 0,\ \theta_w > 0,\ \|q_r - q_i\| < \epsilon\right\}\\
\mathrm{Inv}_{(8)} &= \{(\phi_w, \theta_w) \in X_w, (x_r, y_r, \theta_r) \in X_r \mid \dot{\phi}_w\\
&\quad = \dot{\theta}_w = \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0\}
\end{aligned}
$$

The activities (continuous vector fields) for the two UGV hybrid systems can be expressed as follows:

$$
\mathrm{Act}(1,8) = \{\dot{x}_r = \dot{y}_r = \dot{\theta}_r = \dot{\phi}_w = \dot{\theta}_w = 0 \qquad \mathrm{Act}(5) = \begin{cases} \dot{\phi}_w = u_1^w\\ \dot{\theta}_w = u_2^w \end{cases}
$$

$$
\mathrm{Act}(2) = \begin{cases} \dot{x}_r = u_1^r \cos\theta\\ \dot{y}_r = u_1^r \sin\theta\\ \dot{\theta}_r = u_2^r \end{cases} \qquad \mathrm{Act}(6) = \begin{cases} \dot{x}_r = u_1^r \cos\theta\\ \dot{y}_r = u_1^r \sin\theta\\ \dot{\theta}_r = u_2^r\\ \dot{\phi}_w = \dot{\theta}_w = 0 \end{cases}
$$

$$
\mathrm{Act}(3) = \begin{cases} \dot{\phi}_w = u_1^w\\ \dot{\theta}_w = 0 \end{cases} \qquad \mathrm{Act}(7) = \begin{cases} \dot{x}_r = \dot{y}_r = \dot{\theta}_r = 0\\ \dot{\phi}_w = u_1^w\\ \dot{\theta}_w = u_2^w \end{cases}
$$

$$
\mathrm{Act}(4) = \begin{cases} \dot{x}_r = 0\\ \dot{y}_r = 0\\ \dot{\theta}_r = u_2^r \end{cases}
$$

The above equations for Act(ℓ) thus capture the continuous dynamics of the dual UGV system within the join hybrid automaton. We see that the wheeled UGV is modeled using standard unicycle dynamics while the walking UGV is a basic single integrator model. Note that the joined UGV inherits the unicycle dynamics of the wheeled UGV while the walking UGV inputs now control the movement of the 2-DOF manipulator. Location 2 corresponds to the wheeled UGV moving within the 2-Dimensional plane and similarly Location 3 corresponds to the walking UGV moving horizontally along the x-axis. Location 4 is a rotational only action of the wheeled UGV which orients the magnets on the front of the robot so that they are aligned with the magnets on the rear of the walking UGV along the x-axis. Location 5 corresponds to the magnetic connection between the two UGVs performed by the walking UGV, and the walking UGV also raising its legs off the ground. Location 6 represents the motion of the joined vehicle with the wheeled UGV component providing the locomotion towards interaction locations. Location 7 corresponds to the manipulator on the joined vehicle interacting with the environment while the vehicle remains stationary. Finally, locations 1 and 8 mark the initial and final state of the system where both vehicles are stationary.

6.2 Application: search for an action plan

The construction of an action plan for the dual UGV system proceeds in a similar fashion to the process outlined in the UAV-UGV study. The $A*$ planner weighs the possible transitions based on cost functions evaluated on continuous state space variables of the join hybrid system and the continuous domain is divided and quantized using the centroids of the resulting workspace cells. While there are still multiple valid selections for cost functions as inputs to the planner, the reduced system complexity allows for the selection of simplified cost functions without effecting planner convergence.

Suppose that the planner is given the system at cell ℓ_i and considers transitioning it to some cell ℓ_j; the cell containing the goal location (here: the location of the light switch) will be denoted ℓ_f. With $D(\ell_i, \ell_j)$ expressing the Euclidean distance between the centroids of cell i and j, the short and long-term cost functions that quantify the cost-benefit ratio of a transition between 2 cells are given more simply (no workspace obstacles are considered here):

$$a_{ij} = D\left(\ell_i, \ell_j\right) \qquad\qquad h_j = D\left(\ell_j, \ell_f\right),$$

respectively. For this system these action-specific costs may only apply to events labeled with symbols in (a, r, j). For the remaining three events, namely, positioning p, connection c, and interaction i, fixed costs of the form

$$a_{ij} = C \qquad\qquad h_j = 0,$$

suffice, given that these as generally fixed duration cooperative events. For simplicity, positioning p is set to take place at some fixed distance behind the current location of the walking UGV.

Comparing event sequence costs, the planner this time settles on the sequence *rpcji*. This plan represents the wheeled UGV driving (r) to a location behind the walking UGV and rotating itself to align itself to connect with other vehicle (p); the walking UGV making the magnetic connection with the wheeled UGV and lifting the manipulator mechanism (c); the joined UGV driving towards interaction location q_i (j); and finally, the joined UGV interacting with a light switch at the designated location (i).

6.3 Physical implementation

Realizations of the walking robot and wheeled robot designs were achieved using a mix of 3D printed and off-the-shelf components as seen in Figure 9. Local controllers corresponding to each location were implemented using microcontrollers on each of the robotic agents. A Raspberry Pi on the wheeled robot acts as the coordinating controller in this case and sends the required actions to the walking robot controller to ensure synchronization between the two vehicles. The walking robot is controlled by an ESP32 and commands are received via a Wi-Fi connection with the Raspberry Pi. In testing, the vehicles were able to magnetically connect, move to the interaction location, and flip a standard light switch mounted within the workspace of the walking robot manipulator. In this manner, the UGVs are capable of successfully implementing the sequence calculated by the $A*$ planner and completing the desired task.

7 Conclusion

Cooperative planning for tasks not feasible without physical interaction and nontrivial mechanical coupling between heterogeneous robotic agents is particularly challenging and pushes existing multi-robot planning and control methodologies to their limits. The approach outlined in this paper overcomes this challenge by (a) incorporating aspects of the underlying continuous dynamics that capture the intricacies of physical interaction between the heterogeneous robot teammates and expressing their effects through a hybrid dynamical system modeling framework; (b) subsequently abstracting these continuous dynamics into discrete modes resting on appropriately parameterized control loops, and finally and arguably more importantly, (c) adapting and introducing a novel composition operation for hybrid dynamical systems which is capable of expressing cooperative group behaviors that are neither the union nor the intersection of those of its group members. The combined heterogeneous multi-robot system planning and control architecture is capable of revealing as solutions new cooperative behaviors that were not previously achievable. Yet more work is needed to further develop and extend this approach to enable more automation in the modeling phase, allow concurrent subsystem actions, and achieve higher computational efficiency. Additionally, future work will attempt to adapt the turn-based nature of automata execution to allow for agents to perform concurrent actions. The relaxation of this restriction is likely nontrivial within the confines of the current modeling framework.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

MS developed and refined the planning methodologies and performed the analysis of the case studies. HT supervised the work and provided the conceptual framework for the design of the systems under study. MS and HT wrote and revised the initial draft of the manuscript. All authors contributed to the article and approved the submitted version.

Funding

This work has been supported in part by the U.S. Army through a subcontract by PPG under Collaboration Agreement Number 2022016-142122.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2024.1172105/full#supplementary-material

References

Bhattacharya, S., Kumar, V., and Likhachev, M. (2010). "Search-based path planning with homotopy class constraints," in *Proceedings of the twenty-fourth AAAI conference on artificial intelligence*, 1230–1237.

Casarez, C. S., and Fearing, R. S. (2016). "Step climbing cooperation primitives for legged robots with a reversible connection," in *2016 IEEE international conference on robotics and automation*, 3791–3798.

Cassandras, C., Pepyne, D., and Wardi, Y. (2001). Optimal control of a class of hybrid systems. *IEEE Trans. Automatic Control* 46, 398–415. doi:10.1109/9.911417

Egerstedt, M., Wardi, Y., and Axelsson, H. (2006). Transition-time optimization for switched-mode dynamical systems. *IEEE Trans. Automatic Control* 51, 110–115. doi:10.1109/tac.2005.861711

Heinz, J., and Rogers, J. (2013). "Learning subregular classes of languages with factored deterministic automata," in *Proceedings of the 13th meeting on the mathematics of language (MoL 13)* (Sofia, Bulgaria: Association for Computational Linguistics), 64–71.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H. D., Aghazarian, H., Ganino, A. J., et al. (2003). CAMPOUT: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. *IEEE Trans. Syst. Man, Cybern. – Part A Syst. Humans* 33, 550–559. doi:10.1109/tsmca.2003.817398

Johnson, A. M., Burden, S. A., and Koditschek, D. E. (2016). A hybrid systems model for simple manipulation and self-manipulation systems. *Int. J. Robotics Res.* 35, 1354–1392. doi:10.1177/0278364916639380

Karydis, K., Zarouk, D., Poulakakis, I., Fearing, R. S., and Tanner, H. G. (2014). "Planning with the star(s)," in *Proceedings of the IEEE/RSJ international conference on* Chicago, IL: Intelligent Robots and Systems, 3033–3038.

Kiener, J., and von Stryk, O. (2007). "Cooperation of heterogeneous, autonomous robots: a case study of humanoid and wheeled robots," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, 959–564.

Kim, S., Bhattacharya, S., and Kumar, V. (2014). "Path planning for a tethered mobile robot," in *2014 IEEE international conference on robotics and automation*, 1132–1139.

Mathews, N., Christensen, A. L., O'Grady, R., and Dorigo, M. (2010). "Cooperation in a heterogeneous robot swarm through spatially targeted communication," in *Swarm intelligence*. Editors M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, et al. (Berlin, Heidelberg: Springer Berlin Heidelberg), 400–407.

Mellinger, D., Lindsey, Q., Shomin, M., and Kumar, V. (2011). "Design, modeling, estimation and control for aerial grasping and manipulation," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, 2668–2673.

Mellinger, D., Shomin, M., Michael, N., and Kumar, V. (2013). "Cooperative grasping and transport using multiple quadrotors," in *Distributed autonomous robotic systems*. Editors A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, et al. (Berlin, Heidelberg: Springer), 83, 545–558.

Messing, A., Neville, G., Chernova, S., Hutchinson, S., and Ravichandar, H. (2022). Grstaps: graphically recursive simultaneous task allocation, planning, and scheduling. *Int. J. Robotics Res.* 41, 232–256. doi:10.1177/02783649211052066

Miki, T., Khrapchenkov, P., and Hori, K. (2019). "Uav/ugv autonomous cooperation: uav assists ugv to climb a cliff by attaching a tether," in *2019 international conference on robotics and automation*, 8041–8047.

Ogusu, Y., Tomita, K., and Kamimura, A. (2020). "Microdrone-equipped mobile crawler robot system, dir-3, for high-step climbing and high-place inspection," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, 1261–1267.

Parker, L. E. (2001). *Heterogeneous multi-robot cooperation*. Cambridge, MA: Massachusetts Institute of Technology.

Pirjanian, P., Leger, C., Mumm, E., Kennedy, B., Garrett, M., Aghazarian, H., et al. (2002). "Distributed control for a modular, reconfigurable cliff robot," in *Proceedings of the IEEE international conference on robotics and automation*, 4083–4088.

Posa, M., Cantu, C., and Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *Int. J. Robotics Res.* 33, 69–81. doi:10.1177/0278364913506757

Rimon, E., and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Int. J. Robotics Automation* 8, 501–518. doi:10.1109/70.163777

Sebok, M. A., and Tanner, H. G. (2019). On the hybrid kinematics of tethered mobile robots. *Proceedings of the 2019 Am. Control Conf.*, 25–30. doi:10.23919/ACC.2019.8815354

Stager, A., and Tanner, H. (2019). Composition of local potential functions with reflection. *Proceedings of the IEEE Int. Conf. Robotics Automation*, 5558–5564. doi:10.1109/ICRA.2019.8793807

Stager, A., and Tanner, H. G. (2016). "Stochastic behavior of robots that navigate by interacting with their environment," in *Proceedings of the IEEE conference on decision and control*, 6871–6876.

Stager, A., and Tanner, H. G. (2020). "Mathematical models for physical interactions of robots with their environment," in *Proceedings of the 2018 international symposium on experimental robotics* (Springer Proceedings in Advanced Robotics), 549–558.

Sutera, G., Borgese, A., Guastella, D. C., Cantelli, L., and Muscato, G. (2020). "A multi-robot system for thermal vision inspection," in *Proceedings of the 23rd international symposium on measurement and control in robotics*, 1–6.

van der Schaft, A. J., and Schumacher, H. (2000). *An introduction to hybrid dynamical systems*. Springer.

Zhao, P., Mohan, S., and Vasudevan, R. (2020). Optimal control of polynomial hybrid systems via convex relaxations. *IEEE Trans. Automatic Control* 65, 2062–2077. doi:10.1109/tac.2019.2929110

Zhu, F., and Antsaklis, P. J. (2015). Optimal control of hybrid switched systems: a brief survey. *Discrete Event Dyn. Syst.* 35, 345–364. doi:10.1007/s10626-014-0187-5