Information Sciences 580 (2021) 419–435

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Modeling and control of heterogeneous field robots under partial observation

Chanyoung Ju [a,b], Hyoung Il Son [a,b,*]

[a] Department of Convergence Biosystems Engineering, Chonnam National University, Yongbong-ro 77, Gwangju 61186, Republic of Korea
[b] Interdisciplinary Program in IT-Bio Convergence System, Chonnam National University, Yongbong-ro 77, Gwangju 61186, Republic of Korea

ARTICLE INFO

Article history:
Received 25 March 2021
Received in revised form 18 August 2021
Accepted 20 August 2021
Available online 24 August 2021

Keywords:
Heterogeneous field robots
Supervisory control
Hybrid dynamical system
Discrete event system
Partial observation

ABSTRACT

The control of large-scale dynamic systems, such as field robots, requires a more modular and scalable approach than traditional control theory. Recently, we designed modular supervisory controllers for field robots in agricultural applications based on a hybrid system that combines continuous-time and discrete-event systems. We verified that the developed hybrid system could control field robots and observe all the events while meeting the behavioral specifications. In contrast, all possible events cannot be monitored by a supervisor in a real-field robot system because of the presence of sensors, noise, disturbances, and failure. Therefore, in this study, we expanded the modular supervisor presented in previous studies by considering partial observations. Specifically, we proposed a process for designing an appropriate modular supervisor by considering the observability. The experimental results demonstrated that only observable events could cause state changes, verified by dynamic simulations representing a natural field environment. The effects of the proposed hybrid system-based modeling and control methodology are discussed in light of our systematic results.

1. Introduction

Research on dynamic systems, such as field robot systems, has focused mainly on continuous-time systems modeled by differential equations [1]. However, dynamic system theory aims to control cyber-physical systems that contain large-scale dynamic models and discrete-event dynamics. Advanced systems, such as automobiles, airplanes, networks, and production systems, are combined into hybrid control architectures through a variety of complex connections [2]. However, conventional theories and approaches for continuous-time systems have limitations that primarily apply to the individual parts of hybrid systems that are required to pursue future work [3]. Therefore, advanced automation of field robot systems requires a hybrid control system that overcomes these limitations and is highly scalable and modular.

One solution that has drawn significant attention is the discrete-event system-based supervisory control, which can observe the events or states occurring in a system and systematically identify the controller behaviors (i.e., control actions) [4]. However, implementing such theories and incorporating them into dynamic systems is challenging [5]. For example, existing discrete-event systems and supervisory control frameworks are limited to theories, production processes, and network management. Further, there are leading problems in the implementation and integration of robotic applications [6]. To

* Corresponding author at: Department of Convergence Biosystems Engineering, Chonnam National University, Yongbong-ro 77, Gwangju 61186, Republic of Korea.
E-mail addresses: cksdud15@gmail.com (C. Ju), hison@jnu.ac.kr (H.I. Son).

https://doi.org/10.1016/j.ins.2021.08.071
0020-0255/

address these challenges, we developed a novel hybrid system [7] that combines a low-level continuous-time system [8,9] with a high-level discrete-event system [10] to apply supervisory control theory to a field robots. This methodology has a hierarchy of control architectures that meet the desired behavioral specifications while systematically recognizing the dynamic behavior of large-scale systems.

In a recent study [7,10], we addressed the design of a modular supervisory controller under full observation conditions for field robot systems. In other words, we proved the supervisory control problem (SCP) in the situations where all events occurring in the hybrid system were assumed to be observable. However, because of the lack of sensors for event observation, noise and disturbances, not all events can be observed in the real world. Therefore, such assumptions reduce the reliability of control systems for field robots [11]. This study addressed a supervisory control design problem to improve the feasibility and practicality of the developed hybrid system considering the observability. We define this problem as a supervisory control problem under a partial observation (SCPPO). The main contribution of this study is the proposition of SCPPO to model and control the hybrid system-based field robots by considering realistic environments with unobservable events. We seek to observe control actions and systematic results regarding the behavior of the proposed hybrid system changes in the SCPPO. Additionally, a design process for supervisory controllers was established for the SCPPO based on a hybrid system. This approach ensures the scalability and reliability of the control architectures for the robotic systems, thereby establishing the foundation for the advanced automation of large-scale dynamic systems. Finally, we discuss the direction of heterogeneous field robot modeling and controller design based on the SCPPO, define the limitations, and analyze systematic results based on dynamic simulations. In this study, the developed modular supervisory control system with partial observation is applied to agricultural applications, unlike the existing continuous-time-based heterogeneous field robot system that focuses only on task performance, without considering unobservable events [12].

The remainder of this paper is organized as follows. Section 2 introduces the proposed hybrid system approach and hybrid automata-based modeling method for the heterogeneous field robots. Section 3 defines the control theory for designing modular supervisory controllers, the process of specification design, and the solutions of SCP and SCPPO considering controllability and observability, respectively. Section 4 presents and discusses our experimental setup and results for evaluation of the proposed modular supervisors considering partial observations. Finally, Section 5 summarizes our conclusions and discusses possibilities for future research.

1.1. Related works

This section reviews related studies based on formal methods for multi-robots, particularly discrete-event systems, hybrid systems, and supervisory control systems. Supervisory control theory, a theory proposed by Ramadge and Wonham in 1987, introduced a methodology for controlling discrete-event systems [13]. It has been applied mainly to the field of manufacturing automation [14], however, has recently been introduced into a cyber-physical robotic system (i.e., a dynamic system). For task allocation of cooperative robot teams, a supervisory controller was designed to assign flexibility in task decomposition, coordination control, and failures diagnosis [15]. By constructing a control architecture for the application of heterogeneous multi-robot teams to urban search and rescue, Yugang et al. [16] found that this approach was practical for multi-robot control in field applications and was robust to increasing team size. Nevertheless, most research on the supervisory control of multiple robots aims at autonomous navigation [17] through behavior coordination [18] and formation control [19]. However, these works focus on modeling the discrete-event system and analyzing the supervisory controller rather than implementing or demonstrating their robot system.

Examples of the implementation and application of supervisory control frameworks include warehouse automation [20], agricultural UAVs [21], exploration [22], theme parking vehicles [23], swarm robotics [24], and attempts to integrate self-development tools and frameworks [25–27]. Most of the aforementioned studies employ a simple supervisory control through an automata model and do not handle the implementation problem by combining it with a continuous-time system. In recent research, for example, a learning-based synthesis approach [28], a probabilistic discrete-event system under partial observation [29], fuzzy discrete-event system-based shared control [30], robust supervisory-based control [31] and a multilevel discrete-event system for bus structure [32] have been studied for application to the robotics field based on discrete-event system and supervisory control theory, but these are still in a simple stage. Although hybrid dynamical systems or hybrid control architecture for controlling robots have been studied as an alternative approach, those research did not deal with the scalability and systematic approach for the complex dynamics system [33–36]. Recently, various studies such as warehouse automation by logistic robotic network [37], secure recovery procedure for manufacturing system [38], and automatic controller code generation for swarm robotics [39] are also conducted to expand the supervisory control theory and apply it to cyber-physical systems. These studies, in which supervisory control and a dynamic system based on a discrete-event system are proposed, do not guarantee solutions to the problems mentioned above. Other researchers also attempt to manage a multi-agent system based on hybrid control, but practicality and feasibility are not sufficiently guaranteed from the perspective of a field robot system [40–42]. Therefore, additional research is required to control, implement, and sustain large-scale dynamic systems using a discrete-event system and a supervisory control framework beyond the traditional control method in field robotics. No prior studies have examined the supervisory control of heterogeneous field robots to the best of our knowledge. We address this issue by proposing a hybrid system-based control architecture through a new perspective on supervisory control and classic control considering observability.

This study proposes a hybrid system-based control architecture for heterogeneous field robot cooperation considering partial observations. The hybrid system is a coupling scheme consisting of a continuous-time system and a discrete-event system. We designed the specification language to represent the goals and behavior policies, high-level plants, and high-level controllers based on the discrete-event system. However, when controlling the robot system with dynamics taken into consideration, a low-level controller is required. Therefore, we modeled the plant based on the continuous-time system and designed a low-level controller using ordinary differential equations. The command of a supervisor is transmitted to the low-level controller through the control logic channel, and the output of the field robot plant (measured values, state variables, events, etc.) is transmitted to the high-level stage through the information channel. Consequently, the supervisory controller can control the hybrid system by selectively enabling controllable events based on this information. In this study, we modeled a field robot system based on the dynamics of each mobile robot and previous extensive research [10] using hybrid automata and a hybrid system. Furthermore, the unobservable event-based modular supervisors were designed to be more scalable, maintainable, and superior to the centralized supervisory controller for managing large-scale dynamic systems. In summary, our research aims to develop a hybrid system-based control system that can systematically model, control, and analyze heterogeneous field robots based on partial observation. The proposed hybrid system is implemented, validated, and evaluated in a physics-based simulator similar to a field environment, and we present its systematic results and effectiveness.

2. Hybrid System based modeling

2.1. Hybrid system approach

In this study, a dynamic system is modeled using a hybrid system approach that combines a continuous-time system and a discrete-event system. Continuous-time systems use differential equations, which are typically used to model robotic systems. In contrast, formal methods are used in discrete-event systems, such as automata or Petri nets. We adopted the following deterministic finite-state automaton to construct a hybrid system:

Definition 1. A finite-state automaton G is defined as a 5-tuple consisting of the following elements [4,43]:

$$G = (Q, \Sigma, \delta, q_0, Q_m), \tag{1}$$

where Q is a finite set of states, Σ is a finite set of events, and δ is the partial transition function of Q ($\delta : Q \times \Sigma^* \mapsto Q$), q_0 is the initial state of G, and Q_m is the set of marker states that represent the goal state or final state ($Q_m \subset Q$). Σ^* denotes the set of all finite sequences of events in Σ, including the null sequence ε. In addition, the transition function δ was generalized using the following equations:

$$\delta(q_0, \varepsilon) = q_0 \tag{2}$$

$$\delta(q_0, s\sigma) = \delta(\delta(q_0, s), \sigma) \quad \forall s \in \Sigma^*, \forall \sigma \in \Sigma, \tag{3}$$

where $q_0 \in Q$ represents the initial state, and s is a possible sequence in G.

Definition 2. The languages of G are the *closed behavior* defined as follows:

$$L(G) = \{s \in \Sigma^* | \delta(q_0, s)!\}, \tag{4}$$

where $\delta(q_0, s)!$ indicates that the next state in which the sequence s occurs at q_0 is defined in G. The prefix closure and marked behavior of a language L(G) are defined as follows.

Definition 3. The prefix closure of language L, denoted as $\overline{L}$, is defined as follows:

$$\overline{L(G)} = \{t \in \Sigma^* | (\exists s \in L(G)) t \leqslant s\}. \tag{5}$$

Here, we can confirm that $L(G) \subseteq \overline{L(G)}$ and define the language L(G) as prefix-closed or closed when $L(G) = \overline{L(G)}$.

Definition 4. The definition of the marked behavior of a language L, denoted as L_m, is

$$L_m(G) = \{s \in L(G) | \delta(q_0, s) \in Q_m\} \subseteq L(G), \tag{6}$$

where G is nonblocking if $\overline{L_m(G)} = L(G)$. In other words, a marked state can be reached after a string occurs in any state of G [44]. Nonblocking is also defined as reachability, and it defines the proper automaton G because the language L(G) can fall into deadlocks or livelocks when L(G) is blocking.

For the multiple discrete-event model, let $G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i}), i = 1, 2$ are automata. Their parallel synchronous product, denoted as ||, is defined as follows [4].

Definition 5. Automatons are synthesized by the following operation: $Q = Q_1 \times Q_2, \Sigma = \Sigma_1 \cup \Sigma_2, q_0 = (q_{0,1}, q_{0,2})$, $Q_m = Q_{m,1} \times Q_{m,2}, (q_1, q_2) \in Q_1 \times Q_2, \forall s \in \Sigma_1 \cup \Sigma_2$

$$\delta((q_1, q_2), s) = \begin{cases} (\delta_1(q_1, s), q_2) & \text{if } \textit{condition } 1 \\ (q_1, \delta_2(q_2, s)) & \text{if } \textit{condition } 2 \\ (\delta_1(q_1, s), \delta_2(q_2, s)) & \text{if } \textit{condition } 3. \end{cases} \tag{7}$$

where *condition*1 is $(s \in \Sigma_1 \setminus \Sigma_2) \wedge (\delta_1(q_1, s)!)$, *condition*2 is $(s \in \Sigma_2 \setminus \Sigma_1) \wedge (\delta_2(q_2, s)!)$, and *condition*3 is $(s \in \Sigma_1 \cap \Sigma_2) \wedge (\delta_1(q_1, s)!) \wedge (\delta_2(q_2, s)!)$.

These equations are used to model the discrete part of the field robots and combine them with the dynamics of the continuous-time part. Therefore, the proposed hybrid system is a coupling scheme consisting of a continuous-time system and a discrete-event system, as shown in Fig. 1. We designed a specification language representing the goals and policies, a high-level plant, and a high-level controller based on a discrete-event system. However, for controlling a robot system considering its dynamics, implementing a low-level controller is imperative. Therefore, we modeled the systems based on a continuous-time system and designed a low-level controller using ordinary differential equations. Commands from the high-level controller were transmitted to the low-level controller through the control logic channel. The output of the field robots (measured values, state variables, events, etc.) were sent to the high-level stage through the information channel. Consequently, the high-level controller performed feedback control on the hybrid system by selectively enabling controllable events based on this information.

2.2. Hybrid automata

Hybrid systems are dynamic systems that combine continuous-time and discrete-event dynamics. For example, the actuator (a continuous part) of a field robot controlled by a microprocessor (a discrete part) is a hybrid system. Various hybrid systems have been developed for practical applications, such as automobiles, elevators, heating/cooling systems, and transportation systems. For this purpose, hybrid automata have been defined to model hybrid systems in this study [45].

Definition 6. The hybrid automaton $\mathcal{H}$ is a tuple containing the following elements:

$$\mathcal{H} = (D, C, E, U, F, \phi, I, B, \rho, D_0, C_0), \tag{8}$$

where D is the set of discrete states, C is the set of continuous states, E is the set of events, U is the set of admissible controls, F is the vector field of $\mathcal{H}$ ($F : D \times C \times U \to C$), ϕ is the discrete state transition function of $\mathcal{H}$ ($\phi : D \times C \times E \to D$), I is the set defining an invariant condition ($I \subseteq D \times C$), B is the set defining a guard condition ($B \subseteq D \times E \times C$), ρ is the reset function ($\rho : D \times E \times C \times U \to C$), D_0 is the initial discrete state, and C_0 is the initial continuous state. Moreover, the event set E can be divided into a controllable event set E_c and an uncontrollable event set E_{uc}, or the observable event set E_o and an unobservable event set E_{uo}. The next section introduces the modeling of field robots as hybrid systems through hybrid automata $\mathcal{H}$.

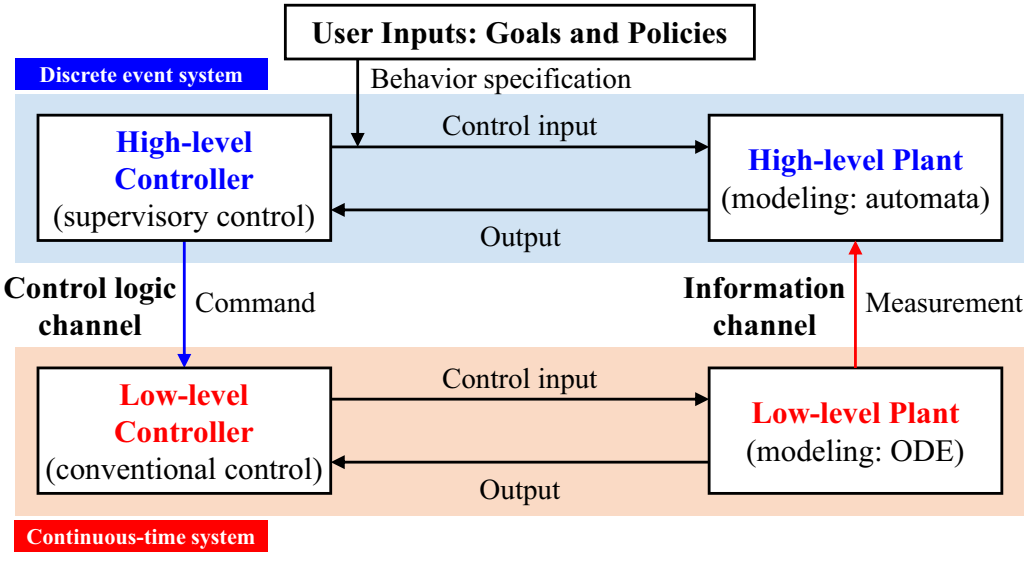


Fig. 1. Diagram of a hybrid system based hierarchical control structure (ODE stands for ordinary differential equation).

2.3. System modeling for heterogeneous field robots

One of our objectives was to enable collaboration among field robots so that hybrid automata models for field robots consisting of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) can be designed. The hybrid models for the UAVs and UGVs are shown in Figs. 2 and 3, respectively. In these models, the initial state is shown, double lines indicate marked states, discrete states are listed in the upper-left section of the shape, continuous states are described in the middle of the shape, and the events and state transition functions are specified by alphabetical designations and arrows, respectively. In contrast to previous studies [10], we modeled the states of hybrid models by focusing on the dynamics of robots using a continuous-time system that reflects continuous state variables such as the height $h_i \in \mathbb{R}$, the relative distance $d_i \in \mathbb{R}$, the position of UGV $p_i \in \mathbb{R}^2$, and the position of UAV $r_i \in \mathbb{R}^3$. In other words, hybrid modeling was considered for field robots by combining continuous-time system-based modeling (please refer to [8,9] for additional details) and discrete-event system-based modeling. As a result, each UAV (modeled as $\mathcal{H}_A$) consists of five states, 23 transitions, and 15 events, whereas each UGV (modeled as $\mathcal{H}_B$) consists of four states, 13 transitions, and 12 events. Therefore, the designed UAV model is assigned a mission in the hovering state and avoids obstacles while flying on an allocated path. In the case of other models, UGVs drive along a given path forming the desired formation. The events for each model are listed in Table 1. Odd-numbered events are controllable events, and even-numbered events are uncontrollable events. The observability of these events is discussed in Section 3.3. Details on how the high-level controller controls the hybrid plant via observable events are introduced in Section 3. Finally, we obtained a hybrid automata model for the entire plant system using the following parallel composition:

$$\mathcal{H}_{plant} = \mathcal{H}_A || \mathcal{H}_B^1 || \mathcal{H}_B^2 || \mathcal{H}_B^3 \tag{9}$$

Specifically, hybrid automata $\mathcal{H}_A$ models (shown in Fig. 2) represent the dynamic behavior of the UAVs. The states are defined as follows: $D_A = \{A_1, A_2, A_3, A_4, A_5\}$, where A_1: *Ideal*, A_2: *Arming*, A_3: *Hovering*, A_4: *Flying*, and A_5: *avoiding*. The eligible events E_A are listed in Table 1, and the discrete transition function is $\phi_A : f(D_A, E_A) = D_A$, as indicated by the arrow in Fig. 2. The initial state $D_{0,A}$ is A_1, and the desired state is $D_{m,A} = \{A_1, A_3, A_4, A_5\}$. From the perspective of the continuous-time system, h_i represents the height of the i-th robot, r_i represents its position, and d_i is the position of the i-th target point (i.e., the desired position). For example, if an event α_5 occurs in state A_2, the state transitions to A_3 occurs, where the continuous state C_A is a state in which the height h_i of the UAVs is greater than zero, the velocity $\dot{r}_i$ is close to zero, and a control command is sent with the target point ($||d_i|| > 0$). The initial continuous state of this model is defined as $h_i, \dot{r}_i, \dot{d}_i = 0$.

The hybrid automata, $\mathcal{H}_B$ (shown in Fig. 3), also models the dynamic properties of the UGVs with the states defined as follows: $D_B = \{B_1, B_2, B_3, B_4\}$, where B_1: *stationary*, B_2: *Navigation*, B_3: *Safety*, and B_4: *Formation*. The discrete transition function is $\phi_B : f(D_B, E_B) = D_B$, as indicated by the arrow in Fig. 2. The initial state is $D_{0,B}$ is B_1, and the desired state is $D_{m,B} = \{B_1, B_2, B_3, B_4\}$. For example, when UGVs attain the desired formation (i.e., the state is B_4) and an eligible event β_4 occurs that detects an obstacle, the state undergoes a transition to B_3. When the UGVs attain the required formation, the relative distance between the robots is controlled by a potential function to maintain the desired distance $\mathcal{D}_f$. In the B_3 state,

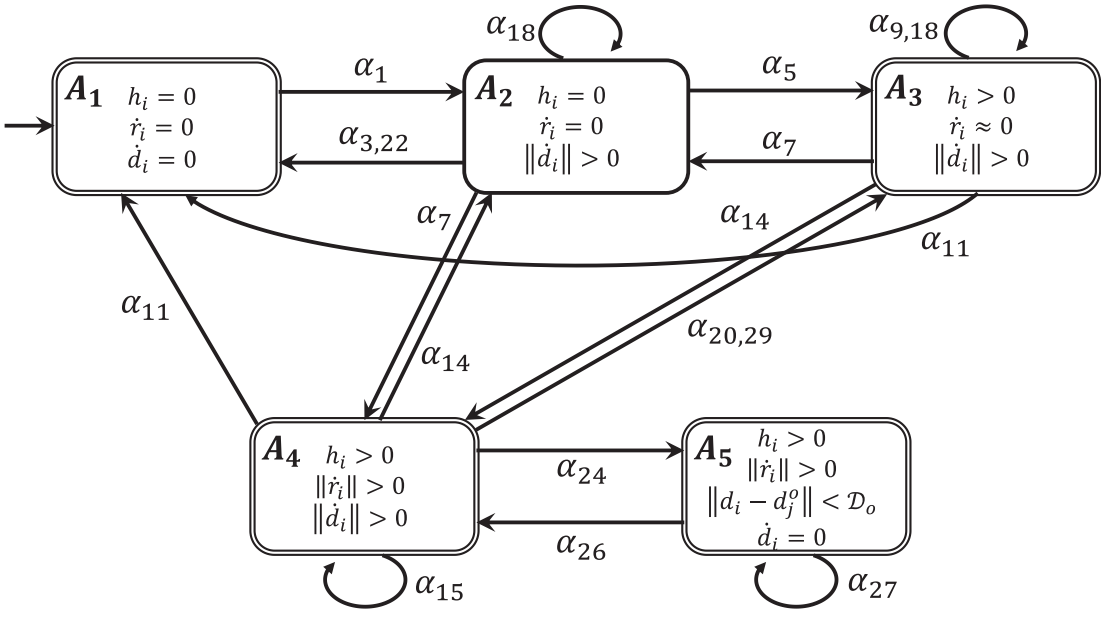


Fig. 2. Hybrid automata model of UAV $\mathcal{H}_A$.

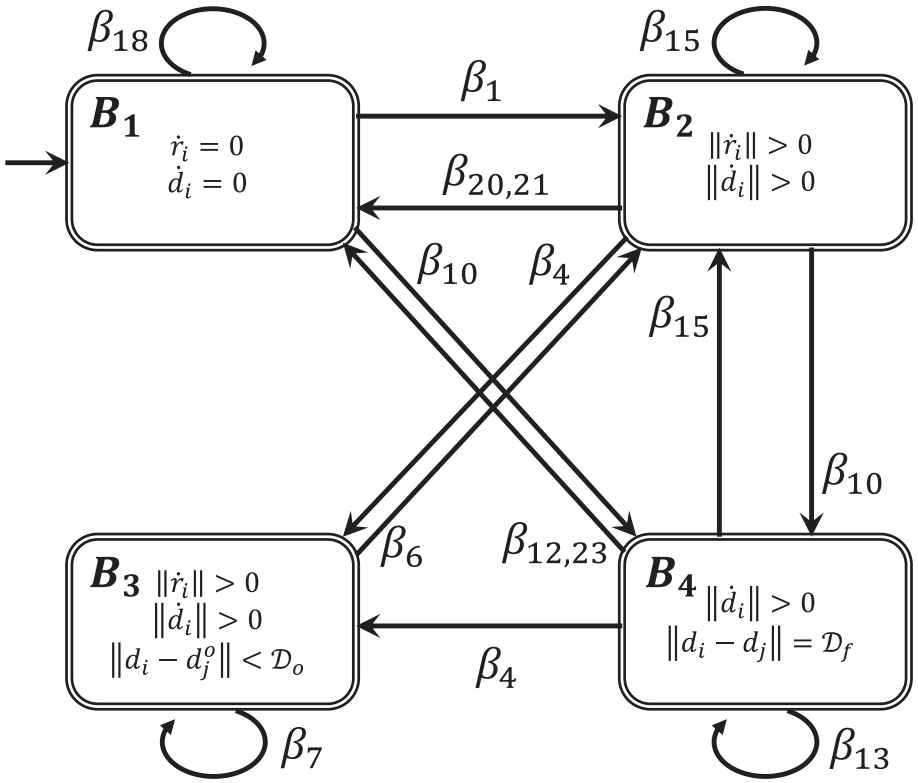


Fig. 3. Hybrid automata model for UGV $\mathcal{H}_B$.

Table 1
Description of eligible events for UAV model $\mathcal{H}_A$ and UGV model $\mathcal{H}_B$.

Plant	Event	Controllable	Observable	Description
UAV ($\mathcal{H}_A$)	α_1	O	O	Arm
	α_3	O	O	Disarm
	α_5	O	O	Take off
	α_7	O	O	Land
	α_9	O	O	Keep hovering
	α_{11}	O	O	Return to home
	α_{14}	X	O	Start mission
	α_{15}	O	O	Keep mission
	α_{18}	X	O	Receive mission
	α_{20}	X	X	Finish mission
	α_{22}	X	O	Time out
	α_{24}	X	O	Detect obstacles
	α_{26}	X	X	Detect free space
	α_{27}	O	O	Keep avoiding
	α_{29}	X	O	Clear mission
UGV ($\mathcal{H}_B$)	β_1	O	O	Start mission
	β_4	X	X	Detect obstacles
	β_6	X	O	Detect free space
	β_7	O	O	Keep avoiding
	β_{10}	X	X	Network connected
	β_{12}	X	O	Network disconnected
	β_{13}	O	O	Keep formation
	β_{15}	O	O	Keep mission
	β_{18}	X	O	Receive mission
	β_{20}	X	O	Finish mission
	β_{21}	O	O	Clear mission
	β_{23}	O	O	Break formation

the action of avoiding obstacles (β_7) is allowed by the high-level controller. In the continuous-time system, the velocity of the UGV ($||\dot{r}_i||$) and control input with the target point ($||\dot{d}_i||$) is greater than zero, and the distance between the UGV and the obstacle ($||d_i - d_j^o||$) is less than the desired safe distance ($\mathcal{D}_o$).

3. Modular supervisory control

3.1. Supervisory control theory

A supervisor is also defined as the automaton $\mathcal{S} = (X, \Sigma, \zeta, x_0, X_m)$, where X, Σ, ζ, x_0, and X_m indicate the state set, event set, state transition function, initial state, and marker states, respectively. Consider a plant defined as a discrete-event system G, where the behavior and the generated language of the plant G under supervision are defined as

$$\mathcal{S}/\mathrm{G} = \{X \times \mathrm{Q}, \Sigma, \zeta \times \delta, (x_0, \mathrm{q}_0), X_m \times \mathrm{Q}_m\} \tag{10}$$

$$\begin{aligned} &L(\mathcal{S}/\mathrm{G}): \ \varepsilon \in L(\mathcal{S}/\mathrm{G}), \forall \mathrm{s} \in \Sigma^*, \varepsilon \in \Sigma: \\ &\mathrm{s} \in L(\mathcal{S}/\mathrm{G}), \mathrm{s}\sigma \in L(\mathrm{G}), \sigma \notin \Theta \Rightarrow \mathrm{s}\sigma \in L(\mathcal{S}/\mathrm{G}) \end{aligned} \tag{11}$$

where σ is an eligible event and Θ is a control map defined as $\Theta : L(\mathrm{G}) \mapsto 2^{\Sigma_c}$. Additionally, the projection map $\mathcal{P} : \Sigma^* \mapsto \Sigma_o^*$ is defined as

$$\begin{aligned} &\mathcal{P}(\varepsilon) = \varepsilon \\ &\mathcal{P}(\sigma) = \begin{cases} \varepsilon & \text{if } \sigma \notin \Sigma_o \\ \sigma & \text{if } \sigma \in \Sigma_o \end{cases} \\ &\mathcal{P}(\mathrm{s}\sigma) = \mathcal{P}(\mathrm{s})\mathcal{P}(\sigma), \forall \mathrm{s} \in \Sigma^*, \forall \sigma \in \Sigma \end{aligned} \tag{12}$$

where the inverse-projection function of $\mathcal{P}$ is denoted by $\mathcal{P}^{-1} : \mathcal{P}(\Sigma_o^*) \to \mathcal{P}(\Sigma^*)$.

The controllability and observability of $L(\mathcal{S})$ with respect to G are described as follows.

Definition 7. $\mathcal{S}$ is defined as *controllable* w.r.t. $(\mathrm{G}, \Sigma_{uc})$ when the following condition is satisfied:

$$(\forall \mathrm{s}, \sigma)\mathrm{s} \in \overline{L(\mathcal{S})}, \ \sigma \in \Sigma_{uc}, \ \mathrm{s}\sigma \in L(\mathrm{G}) \Rightarrow \mathrm{s}\sigma \in \overline{L(\mathcal{S})}. \tag{13}$$

In other words, s, which is allowable by $\mathcal{S}$ and an uncontrollable event σ, is eligible in G if the sequence sσ is eligible in G. Furthermore, if $\mathcal{S}$ also allows sσ, then $\mathcal{S}$ is controllable with respect to G.

Definition 8. For $\mathcal{S} \subset \mathrm{G}, \mathcal{S}$ is said to be *observable w.r.t.* $(\mathrm{G}, \mathcal{P}, \Sigma_{uo})$ when the following condition is satisfied:

$$\begin{aligned} &\left(\forall \mathrm{s}, \mathrm{s}', \sigma \in \overline{L(\mathcal{S})}\right)\sigma \in \Sigma_{uo}, \mathcal{P}(\mathrm{s}) = \mathcal{P}(\mathrm{s}'), \\ &\mathrm{s}\sigma \in \overline{L(\mathcal{S})}, \mathrm{s}'\sigma \in L(\mathrm{G}) \Rightarrow \mathrm{s}'\sigma \in \overline{L(\mathcal{S})}. \end{aligned} \tag{14}$$

If the string sσ is permissible by the supervisor $\mathcal{S}$ and s$'\sigma$ is eligible in the plant G, then $\mathcal{S}$ must also permit s$'\sigma$, where two strings $\mathrm{s}, \mathrm{s}' \in \Sigma^*$ are recognized as the same string by the projection map $\mathcal{P}$ and are also permissible by $\mathcal{S}$, when σ is an unobservable event, then $\mathcal{S}$ is observable *w.r.t.* G.

Furthermore, the SCP used to design a supervisor is defined as follows:

Definition 9. For a given $K \subseteq \mathrm{G}$, we identify a supremal language $\mathcal{S}$ that is controllable with respect to $(\mathrm{G}, \Sigma_{uc})$ that satisfies $L(\mathcal{S}/\mathrm{G}) = K$ and $L(\mathcal{S}/\mathrm{G}) = \overline{L_m(\mathcal{S}/\mathrm{G})}$.

Therefore, if K is defined as the specification for G, then the SCP identifies a supervisory controller that satisfies $L(\mathcal{S}/\mathrm{G}) = K = \overline{L_m(\mathcal{S}/\mathrm{G})}$ and is nonblocking and controllable with respect to G. Within K, a supremal controllable sublanguage of K was identified as the solution of the SCP. Therefore, $\mathcal{S}$ can maximally allow the eligible language to occur in G. Please refer to [7,10] for more detailed procedures and algorithms for the SCP.

3.2. Design of behavior specifications

The supervisor was the result of establishing a legal language (i.e., behavior specification) and subsequently finding a supremal controllable sublanguage for the plant. Alternatively, we can find a controllable language that meets the conditions for becoming an appropriate supervisor $\mathcal{S}_i$ concerning the plant model $\mathcal{H}_i$. The behavioral specifications are designed to sat-

isfy the nonblocking, controllability, and non-conflict conditions for the cooperation of field robots. The policies $\mathbf{POL}_1, \mathbf{POL}_2, \ldots, \mathbf{POL}_{10}$ for designing the specifications are defined as follows:

- $\mathbf{POL}_1$: Arming must occur before the UAV takes off (i.e., hovering).
- $\mathbf{POL}_2$: A mission is assigned while the UAV is arming and is carried out while the UAV is hovering.
- $\mathbf{POL}_3$: When a UGV completes its mission, the corresponding UAV hovers and waits for the next mission.
- $\mathbf{POL}_4$: UGVs form the desired formation when the network is connected.
- $\mathbf{POL}_5$: UGVs can only receive missions and start missions when they are stopped.
- $\mathbf{POL}_6$: UGVs perform missions by creating formations.
- $\mathbf{POL}_7$: At the end of a mission, a UAV remains hovering, and the corresponding UGV remains stopped.
- $\mathbf{POL}_8$: If an obstacle is identified while a robot is moving, avoiding that obstacle becomes the top priority.

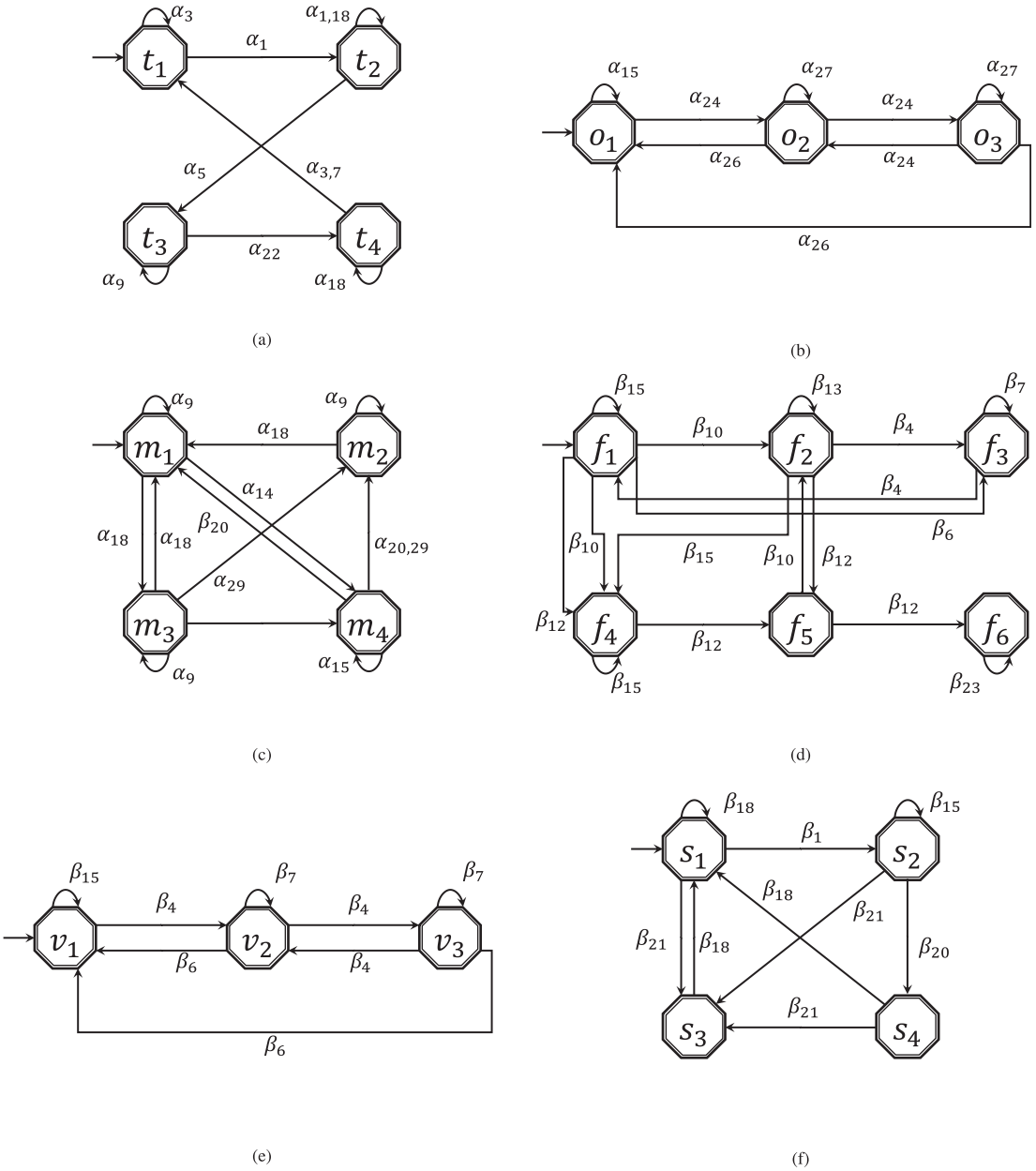


Fig. 4. Specifications for field robots: (a) flight for UAV K_A^1, (b) obstacle avoidance for UAV K_A^2, (c) mission management for UAV K_A^3, (d) navigation for UGV K_B^1, (e) obstacle avoidance for UGV K_B^2, and (f) mission management for UAV K_B^3.

- **POL**$_9$: Duplicate missions are not assigned to a given robot.
- **POL**$_{10}$: When a mission is over, robots must wait for the next mission.

Therefore, we modeled the specifications K_i that met these policies before synthesizing the modular supervisory controller $\mathcal{S}_i$, as shown in Fig. 4. Six specifications were designed to achieve the desired control objectives for each robot, as shown in Fig. 4. Figs. 4(a), 4(b), and 4(c) present the control specifications for the UAV models for arming and disarming, obstacle avoidance, and mission management, respectively. Additionally, Figs. 4(d), 4(e), and 4(f) present the control specifications for navigation, obstacle avoidance, and mission management for UGVs, respectively.

For example, we constructed the automaton K_A^1 presented in Fig. 4(a), which account for the policies of **POL**$_1$ and **POL**$_2$. The states of K_A^1 are defined as follows: (1) state t_1 indicates that the UAV is stopped; (2) state t_2 indicates the arming state in which the UAV operates and is assigned a desired goal; (3) the UAV is driven to perform a given mission in t_3, and (4) state t_4 represents the situation in which the UAV avoids obstacles. Specifically, the *take off* event α_5 is considered in the process of transition from state t_2 to t_3, and then **POL**$_1$ is satisfied. For **POL**$_2$, the control objective can be guaranteed by the synthesis of specifications (Figs. 4(a) and 4(c)). The mission is assigned when the status of the UAV is t_2 and m_1, and the assigned mission is performed in state m_4. It also satisfies **POL**$_7$ because β_{20}, the *finish mission* of the UGV is considered in Fig. 4(c). Please see [7] for detailed specifications.

We also constructed an automaton K_B^1, as shown in Fig. 4(d) for the specifications of the UGV (**POL**$_3$ to **POL**$_7$). Specifications $K_A^2$and K_B^2considering **POL**$_8$ are presented in Figs. 4(b) and 4(e), respectively. Furthermore, the specifications K_A^3 and K_B^3 reflecting **POL**$_9$and **POL**$_{10}$are presented in Figs. 4(c) and 4(f), respectively. In the case of UGVs, state B_1 receives mission transitions to state B_2 when a mission begins. In state B_2, a mission is performed depending on the controllable event β_{15}. Therefore, the control objectives model decides the events to be allowed and disallowed by the supervisor, such that robots do not collide with each other during a mission. Additionally, when an obstacle is found, it receives the highest priority. In UGVs, the system is designed to maintain the UGV formation while following the given path. The design specifications were formulated to reflect the desired policies and meet the conditions for the proper formal language K_i. In sum, the behavior of plant $\mathcal{H}_{plant}$ is selectively controlled by the modular supervisors $\mathcal{S}_i$ to satisfy the behavior specifications K_i. In the scenario of this study, the UAV performs mapping while flying in an unstructured environment. Based on the generated environment information, UGVs travel along a given route, avoiding obstacles.

3.3. Supervisory Controller for Heterogeneous Field Robots under Partial Observation

For modular supervisory control, all events Σ and specification K_i are synthesized to obtain the legal language. Note that not all events are represented in Fig. 4. If $\mathcal{S}_i, i = 1, 2, \ldots, m$, is verified for the legal specifications $K_i, i = 1, 2, \ldots, n$ $(m \leqslant n)$, then $\mathcal{S}_i$ is defined as a modular supervisory controller. The modular supervisor $\mathcal{S}_i$ satisfies the non-conflict condition, defined as follows:

$$\overline{\mathcal{S}} = \overline{\mathcal{S}_1} \wedge \overline{\mathcal{S}_2} \wedge \ldots \wedge \overline{\mathcal{S}_m},$$

where $\wedge$ is a meet product [4] (defined as $\mathcal{S}_1 \wedge \mathcal{S}_2 = L(\mathcal{S}_1) \bigcap L(\mathcal{S}_2)$). If $\mathcal{S}_i$ satisfies non-conflict condition, then all modular supervisors can control plant $\mathcal{H}_{plant}$ in the same manner as the centralized supervisor $\mathcal{S}$. If $\mathcal{S}_i$ is conflicting, then the modular supervisors do not satisfy the specifications $K_i, i = 1, 2, \ldots, n$ because they allow unnecessary events. Therefore, the modular supervisor $\mathcal{S}_i$ must satisfy the conditions of controllability, nonblocking, and nonconflicting.

For SCPPO, we specified that modular supervisors could not observe some events. In other words, events such as *finish mission* α_{20}, *detect free space* α_{26}, *detect obstacles* β_4, and *network connected* β_{10} are unobservable, as listed in Table 1. These conditions reflect observational inaccuracies due to communication errors, sensor failures, and noise in the real world. In the case of SCPPO, the solution is to find the supremal observable sublanguages of K_i for $(\mathcal{P}, \Sigma_{uo})$, and then solve the SCP (Algorithm 1). Therefore, the solution to the modular SCPPO is presented in Algorithm 2.

Algorithm 1: Solution of modular SCP

Input: Plant $\mathcal{H}_i$ and specification $K_i, i = 1, 2, \ldots, n$.
Output: Modular supervisors $\mathcal{S}_j$ or $\mathcal{S}\prime_j, j = 1, 2, \ldots, m$.

- Step 1: Define the subplant automaton $\mathcal{H}_{sub,i}$ and specification automaton K_i.
- Step 2: Determine $\mathcal{S}_j$ (the supremal controllable sublanguage) of K_i and $\mathcal{H}_{sub,i}$.
- Step 3: Verify the controllability of K_i w.r.t. $\mathcal{H}_{sub,i}$ using Σ_c in K_i. If K_i is controllable, then $\mathcal{S}\prime_j$ exists, where $\mathcal{S}\prime_j = K_i$. Goto Step 6.
- Step 4: If the specification K_i does not satisfy steps 3, go back to step 1 and redesign. Nevertheless, if K_i does not meet the proper conditions, then $\mathcal{S}\prime_j$ does not exist. Goto Step 5.
- Step 5: $\mathcal{S}_j$ is the solution of the SCP w.r.t. $(K_i, \mathcal{H}_{sub,i})$. **return** $\mathcal{S}_j$
- Step 6: If $L(\mathcal{S}_j) = L(\mathcal{S}\prime_j / \mathcal{S}_{sub,i})$, then $\mathcal{S}\prime_j$ is the solution of the SCP w.r.t. $(K_i, \mathcal{H}_{sub,i})$. **return** $\mathcal{S}\prime_j$, **else return** $\mathcal{S}_j$.

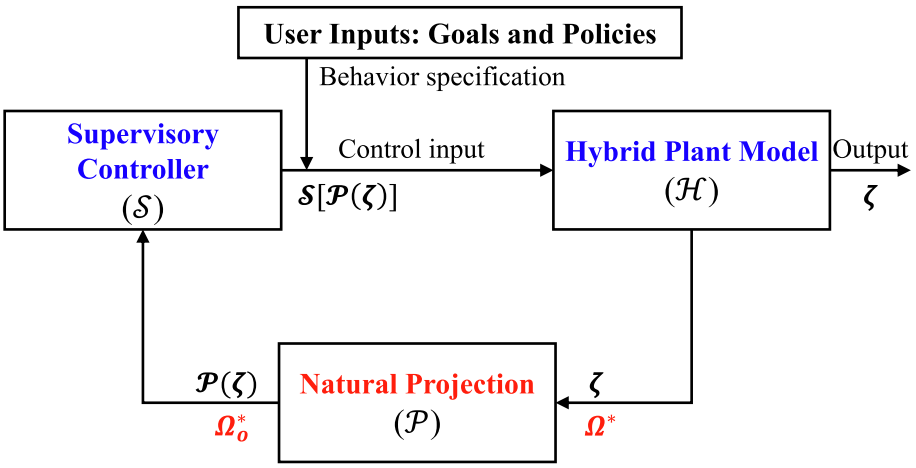


Fig. 5. Supervisory control architecture under partial observation for the hybrid plant.

Algorithm 2: Solution of modular SCPPO

Input: Plant $\mathcal{H}_i$, projection map $\mathcal{P}$, and specification $K_i, i = 1, 2, \ldots, n$. **Output:** Modular supervisors $\mathcal{S}_j, j = 1, 2, \ldots, m$.

- Step 1: Define the subplant automaton $\mathcal{H}_{sub,i}$ and specification automaton K_i. In addition, we define the projection map $\mathcal{P}$ and the unobservable event set Σ_{uo}.
- Step 2: Determine K_i^o (the supremal observable sublanguage) of K_i w.r.t. $(\mathcal{P}, \Sigma_{uo})$.
- Step 3: Calculate the projection map $\mathcal{P}(K_i^o)$ and $\mathcal{P}(\mathcal{H}_{sub,i})$
- Step 4: Follow the SCP indicated in Algorithm 1 w.r.t. $\mathcal{P}(K_i^o)$ and $\mathcal{P}(\mathcal{H}_{sub,i})$.
- Step 5: Calculate the inverse-projection map $\mathcal{P}^{-1}\{\mathcal{P}(\mathcal{S}_j)\}$
- Step 6: Verify the nonconflicting of $\mathcal{P}^{-1}\{\mathcal{P}(\mathcal{S}_i)\}$ w.r.t. $\mathcal{H}_{sub,i}$. If $\mathcal{P}^{-1}\{\mathcal{P}(\mathcal{S}_i)\}$ is nonconflicting, it returns to Step 1 and redesign.
- Step 7: Finally, $\mathcal{S}_j$ is the solution of the SCPPO w.r.t. $(K_i, \mathcal{P}, \mathcal{H}_{sub,i}, \Sigma_{uc}, \Sigma_{uo})$. **return** $\mathcal{S}_j$

A subcomponent $\mathcal{H}_{sub,i}$ of plant $\mathcal{H}_{plant}$ consists of a UAV $\mathcal{H}_A$ and UGV $\mathcal{H}_B$. First, we check whether each specification K_i satisfies the controllability, non-blocking, and non-conflict conditions for the plant. According to Algorithm 1, the specifications K_i that satisfy the requirements for the subplant UAV $\mathcal{H}_A$ and UGV $\mathcal{H}_B$, namely K_A^2, K_B^1, and K_B^2, are defined as modular supervisors $\mathcal{S}_A^2, \mathcal{S}_B^1$, and $\mathcal{S}_B^2$, respectively. In contrast, the supremal controllable sublanguage $\mathcal{S}_i$ is synthesized for the specifications K_i that do not satisfy controllability, which are K_A^1, K_A^3, and K_B^3. The obtained supremal controllable sublanguage $\mathcal{S}_i$ is the solution to the SCP for K_i and $\mathcal{H}_i$, as shown in Figs. 6 and 7, respectively. Then, the modular supervisors under partial observation, SCPPO of $\mathcal{H}_i$, are determined through Algorithm 2 w.r.t. $\mathcal{P}$ and $\Sigma_{uo} = \{\alpha_{20}, \alpha_{26}, \beta_4, \beta_{10}\}$.

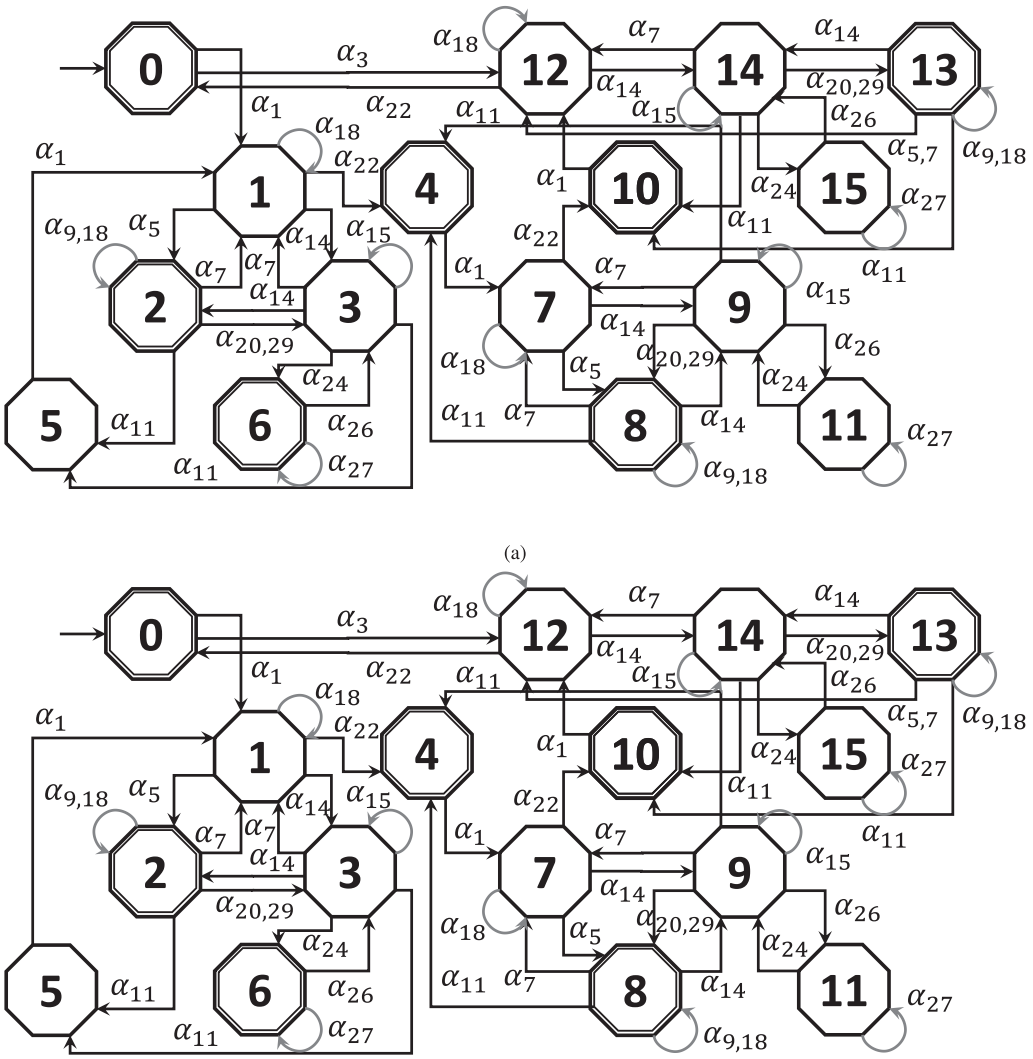
As a result, we found a modular supervisor $\mathcal{S}_i, i = 1, 2, \ldots, 6$ for the specification language K_i with respect to $\mathcal{H}_i$. Finally, we verified that the designed modular supervisory controllers use a supervisory control synthesis tool called TCT [6]. These modular supervisors $\mathcal{S}_i$ can control complex dynamic systems with the desired behavior by selectively allowing or disallowing controllable events Σ generated by the plant $\mathcal{H}_{plant}$ under partial observations (Fig. 5).

4. Experimental results and discussions

4.1. Experimental setup and implementation

To control a target plant $\mathcal{H}_{plant}$ modeled by hybrid automata, we designed a low-level controller and modular supervisory controller. The control function of the modular supervisor was transmitted through the control logic channel (i.e., the discrete-to-continuous interface). The generated events and measured state variables originating from the controlled field robot system were communicated to the high-level controller through the information channel (i.e., continuous-to-discrete interface). Based on this information, the modular supervisor provided a feedback control loop that satisfied the behavioral specifications reflecting the control objectives under partial observation.

A physics-based simulator was constructed to implement and verify the designed modular supervisory controller and hybrid control system for field robots. The robot system included one UAV and three UGVs for cooperative mapping. Each



(a)

(b)

Fig. 6. UAV modular supervisors ($\mathcal{S}_A^1$ and $\mathcal{S}_A^3$).

robot is equipped with LIDAR (light detection and ranging) and RGB (red, green, blue) camera sensors to create a three-dimensional map of the environments and recognize obstacles. The desired path from the starting point to the final point is computed by the A^* algorithm, which is a graph traverse-based path search algorithm to find the shortest path [46]. In the experimental procedure, the UAV performs global mapping by driving at altitudes higher than the fruit tree, and the UGV performs local mapping on the ground, suggesting that the cooperation of heterogeneous robots has a spatial advantage. The entire system's implementation uses CoppeliaSim, a robot simulator, and MATLAB for low-level and high-level controllers as shonwn in Fig. 8. A discrete-event system and supervisory control theory computation tool are combined with MATLAB, which sends real-time data to the simulator and receives responses for the hybrid system. To evaluate the performance of the proposed hybrid control system, the output was recorded at 50 Hz.

4.2. Results and discussions

Fig. 9 shows scenes of the dynamic progress of the heterogeneous field robot during the experiments. The heterogeneous field robots are assigned the desired path from the initial position to the target point. First, when an event that starts a mission occurs, the UAV maps the surrounding environment through flight. At the end of the UAV's flying, the environmental information on the constructed map is shared with the UGVs. UGVs drove the orchard environments based on shared infor-

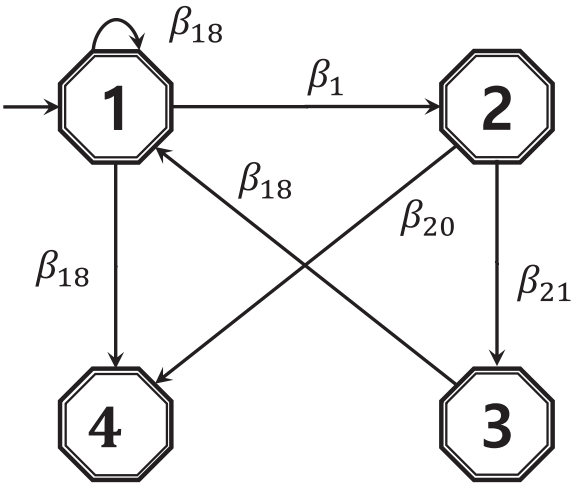


Fig. 7. UGV modular supervisor ($\mathcal{S}_B^3$).

mation while keeping the formation. When the UGV reaches the goal point under the supervision of the modular supervisors, the given cooperative mission ends.

Figs. 10 and 11 show the experimental results related to the hybrid control system. First, Fig. 10 shows the state transition, event occurrence, and trajectory of the field robot when all events are fully observable. The designed policies and behavior specifications to achieve control objectives can be classified into three types: 1) path-following control, 2) obstacle avoidance control, and 3) formation control. Fig. 10 shows that the field robot achieves three goals, and the supervisory controller also selectively allows controllable events while observing the eligible events. In other words, the supervisory controller manages the hybrid system by selectively suppressing controllable events to satisfy control specifications expressing policies.

In contrast, Fig. 11 shows the results of SCPPO when events α_{20} and β_{10} are unobservable. Compared with Figs. 10(b) and 11(b), we can see that α_{20}, which is a *finish mission* event, and β_{10}, which is a *network connection* event, were not observed under the same experimental conditions. The modular supervisor $\mathcal{S}_i$ cannot recognize these events because unobservable events E_{uo} are transformed into null events ε by the projection function $\mathcal{P}$. Therefore, Figs. 10(b) and 11(b) clearly indicate that the state of the field robot D_i is different because of an unobservable event. In the case of UAV $\mathcal{H}_A$, the path-following mission with obstacle avoidance was completed in the experimental environment, but the unobservable event α_{20} is not observed in the modular supervisors, as shown in Fig. 11(b). As a result, Fig. 11(a) shows that the state of the UAV D_A is maintained without transitioning from mission state A_4 to the hover state A_3.

UGV $\mathcal{H}_B$ also fails to generate the desired formation owing to the unobservable event β_{10}, and the state remains only *stationary* B_1. The reason for the non-transition of UGV states is that when designing the behavior specifications K_i and supervisory control $\mathcal{S}_i$, the UGV aims to perform the given mission in the *formation state* B_4 (**POL**$_6$). In addition, the stationary state

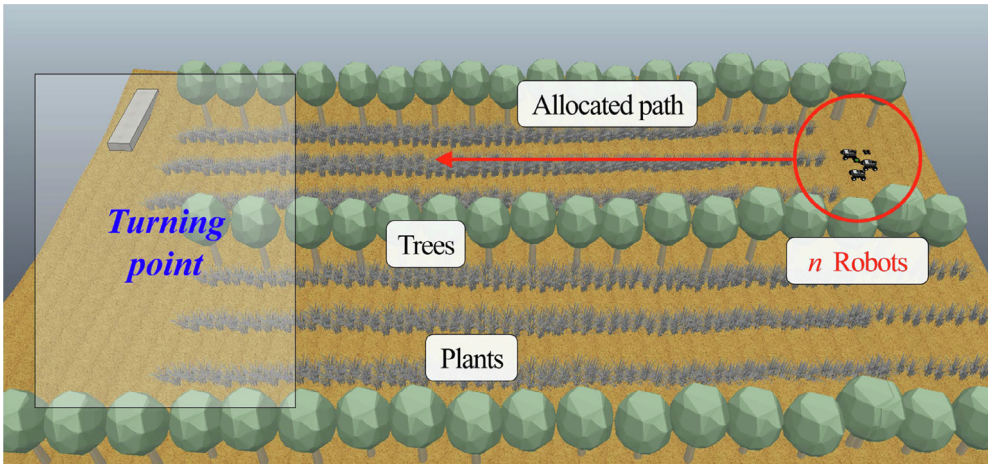


Fig. 8. Experimental environment for a supervisory control under partial observation.

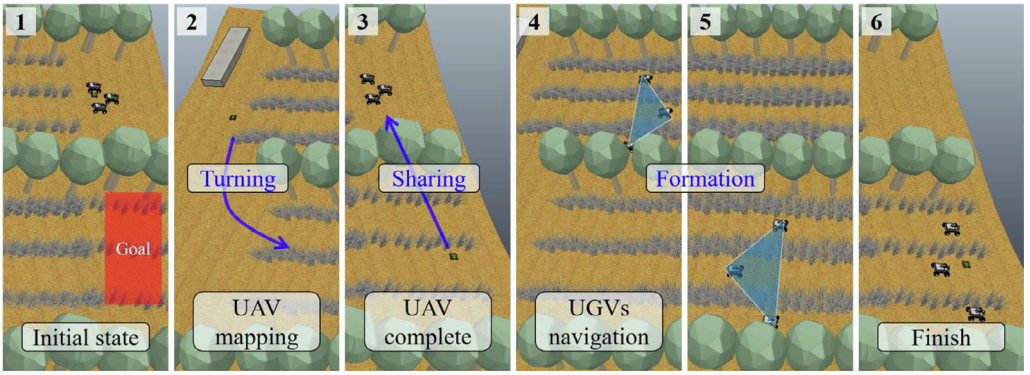


Fig. 9. Experimental scenes for cooperation of heterogeneous field robots.

of the UGV, B_1, was defined as the marked state D_m and was subsequently not changed in any other state under partial observation. In other words, the controllable events E_c that can be controlled by the modular supervisors $\mathcal{S}_i$, were disabled to transition from the current state D to the marked state D_m such that the event related to the formation control and mission control of UGVs was disallowed in *stationary* B_1.

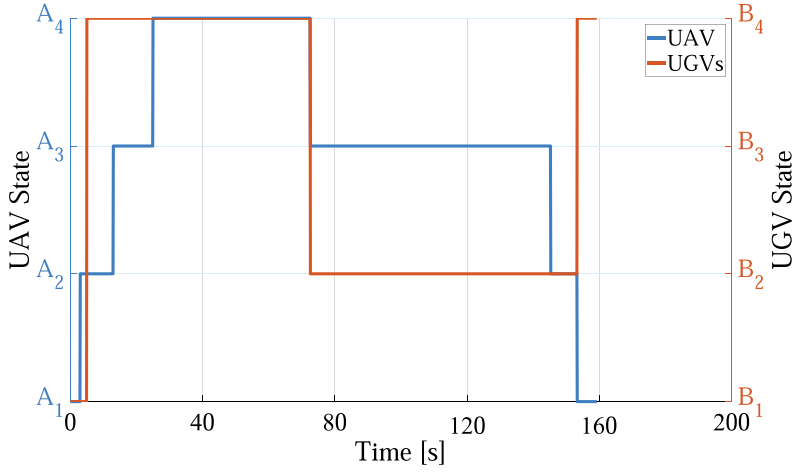
These results ensure that unobservable events do not cause a state transition and that supervisory controls meet behavioral specifications despite partial observation. In other words, when the behavior specifications K_i satisfy the proper conditions through Algorithm 2, the modular supervisors $\mathcal{S}_i$ can control the plant like a global supervisory controller even under partial observation. In designing these behavior specifications K_i, the occurrence of unobservable events means that the control trajectory of the plant system could not deviate from the path to which K_i belongs. Nevertheless, it can be found that α_9, a *hovering control* event, occurs when the *finish mission* event α_{20} of the UAV cannot be observed by supervisors. The permission of α_{20} implies that when the given mission is over, a *hovering control* event is allowed by the supervisory controller to ensure that the UAV is safe even if the desired events do not occur. As a result, in Figs. 10(c) and 11(c), it can be seen that the UAV was navigated with the desired trajectory but not the UGV in the partial observation environments (unobservable events α_{20} and β_{10}).

Although two events were assumed to be unobservable in these experiments, these systematic results vary depending on how we design the marked state or behavior specifications and the controllable or observable events. The reason that the supervisory control system satisfies the desired objective even when there is a non-change of state due to hidden observations is that modular supervisors under partial observation are designed as in Algorithm 2. The results of the SCP and SCPPO, which reflect the conditions of the ideal and actual environments, respectively, differ. The control system for field robots can be subject to many unexpected situations owing to disturbances and noise. Therefore, the proposed hybrid systems consisting of discrete-event systems and continuous-time systems should be designed to achieve the desired control objectives, even when unobservable events exist. Although previous studies have addressed SCP under ideal conditions [7,10], this study contributes to the area of supervisory control of field robots by covering a comprehensive study defined as SCPPO. Compared with our previous studies [7,10], these results indicate that even if a heterogeneous field robot is added to the entire system, the designed modular supervisors can control the hybrid systems without additional modeling under partial observations. Therefore, the proposed algorithm for SCPPO guarantees reliability and scalability for the practical application of field robot systems. This study, although the SCPPO solution is developed and the experimental results are presented through dynamic simulation, proves the effectiveness of the hybrid system-based supervisory control framework for the modeling and control of the field robot under partial observation. Moreover, the proposed hybrid system-based supervisory control system is superior for systematically addressing the large-scale dynamic system.

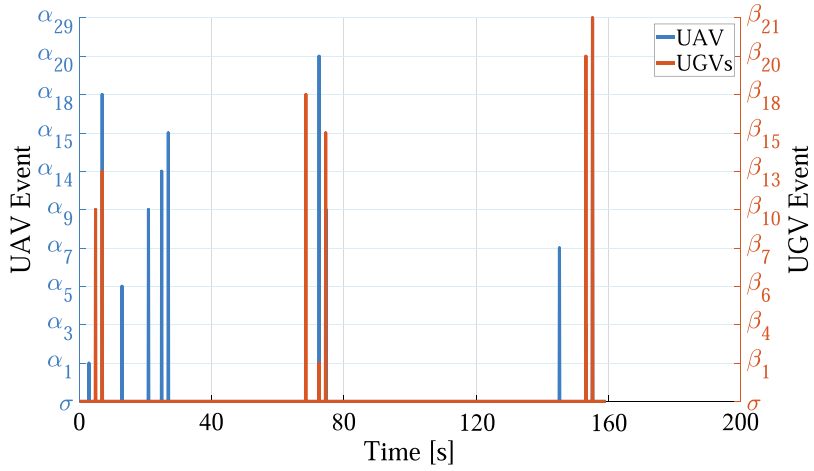
5. Conclusions

We used hybrid automata to model hybrid systems to model and control field robots and solved the SCP under partial observation conditions (i.e., SCPPO). The performance and efficiency of the proposed system were evaluated and verified based on a field robot system consisting of one UAV and three UGVs. For the dynamic large-scale plant that was modeled, we designed legal languages (i.e., behavior specifications) to achieve the control objectives. Based on these specifications, modular supervisory controllers were synthesized using the proposed Algorithms 1 and 2. Additionally, to establish partial observation conditions, we defined events that could not be observed to reflect natural environments.

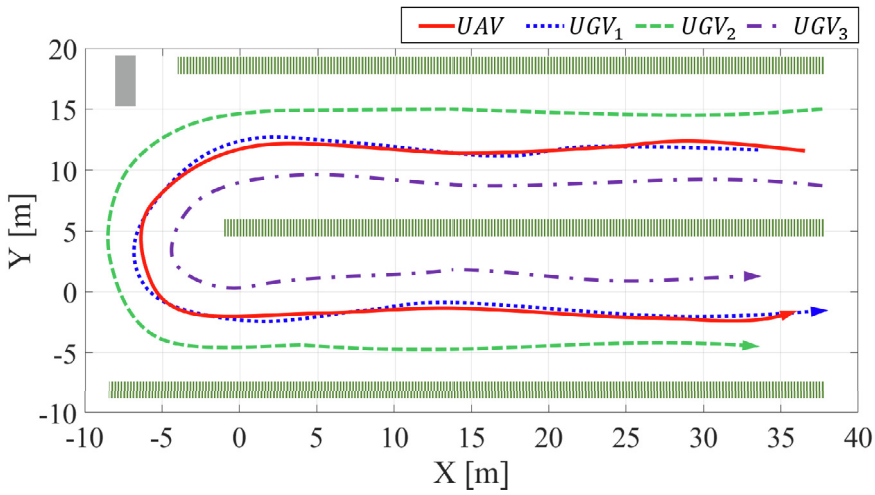
We implemented the proposed system and performed experiments using a physics-based simulator to evaluate our proposed hybrid system approach and develop a supervisory controller. The experimental results confirmed that the field robots achieved the given control objectives. However, no state transitions occurred in unobservable events; thus, the supervisory controller could not observe such transitions. These results mean that the modular supervisors selectively allow controllable



(a)

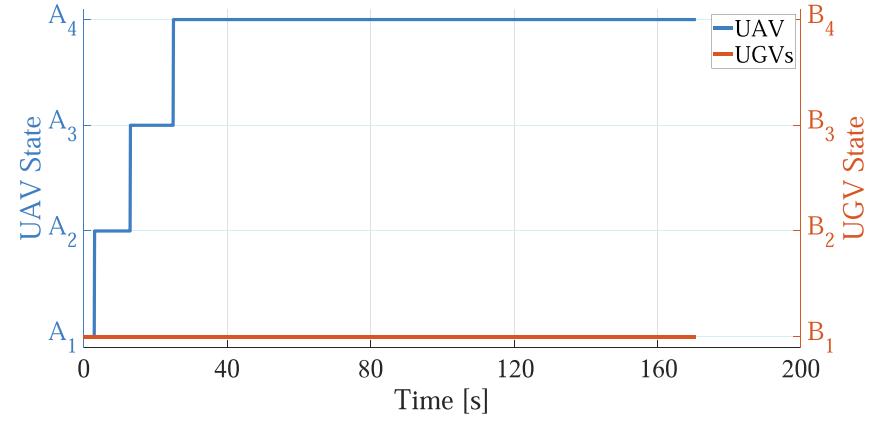


(b)

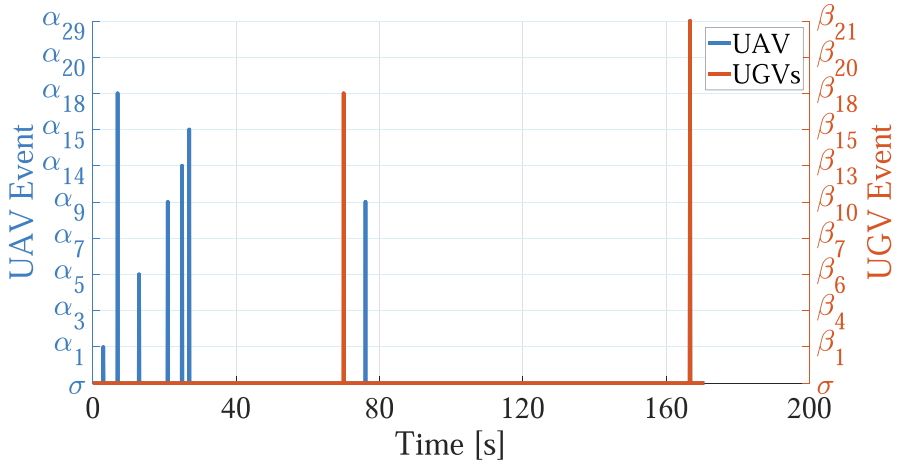


(c)

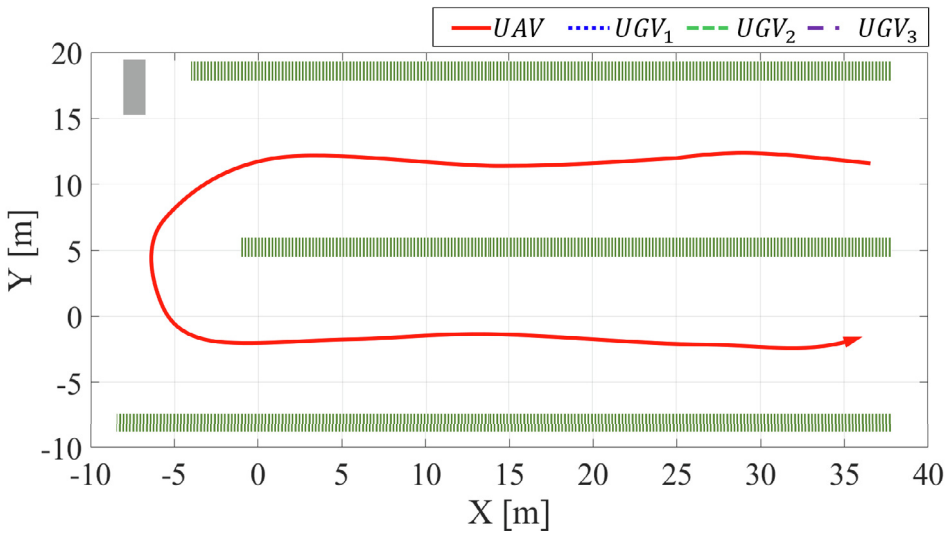
Fig. 10. Experimental results of full observation: (a) state transition, (b) event sequence, and (c) trajectory.



(a)



(b)



(c)

Fig. 11. Experimental results of partial observation: (a) state transition, (b) event sequence, and (c) trajectory.

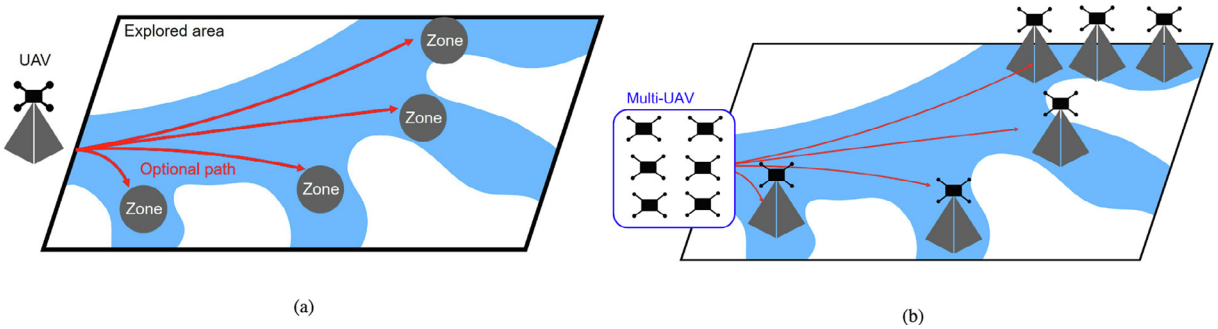


Fig. 12. Conceptual sketch of tributary mapping cooperation: (a) single UAV and (b) multi-UAV system.

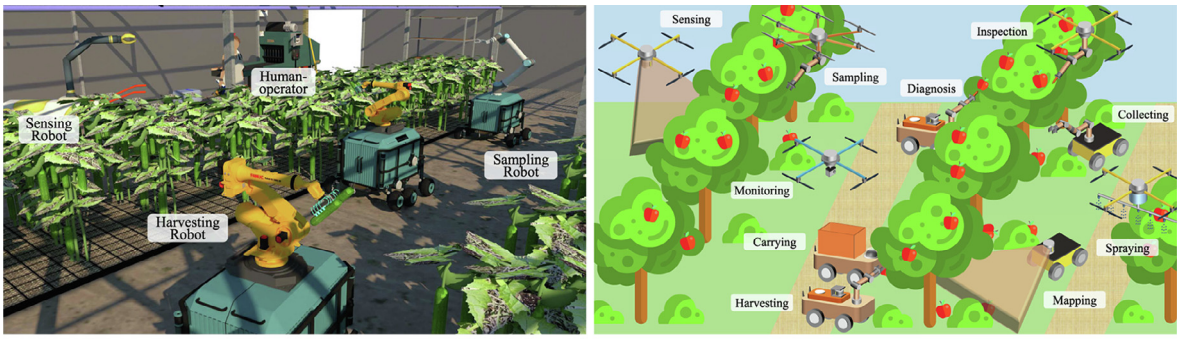


Fig. 13. Conceptual sketch of heterogeneous field robots based agricultural cooperation: (a) open field and (b) greenhouse.

events to follow the marked state that satisfies the designed behavioral specifications under partial observation. In the future, we considered the heterogeneous robot-based cooperative control system of tributary mapping and smart farming for practical applications as shown in Figs. 12 and 13. In these applications, we attempt to systematically solve how to split multiple robots to map the tributary branches and how to allocate heterogeneous tasks to heterogeneous field robots. Furthermore, a high-level supervisory controller will be developed to construct a hierarchical supervisory control architecture (i.e., three control layers) [47] and handle the failure diagnosis of decentralized supervisory control systems [14].

CRediT authorship contribution statement

Chanyoung Ju: Methodology, Software, Formal analysis, Writing - original draft, Writing - review & editing. **Hyoung Il Son:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF- 2018R1D1A1B07046948) and Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE)(P0008473, HRD Program for Industrial Innovation).

References

[1] J. Kim, S. Kim, C. Ju, H.I. Son, Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications, IEEE Access 7 (2019) 105100–105115, https://doi.org/10.1109/ACCESS.2019.2932119.
[2] R. Goebel, R.G. Sanfelice, A.R. Teel, Hybrid dynamical systems, IEEE Control Systems Magazine 29 (2) (2009) 28–93.
[3] K. Cai, W.M. Wonham, Supervisor localization: a top-down approach to distributed control of discrete-event systems, IEEE Trans. Autom. Control 55 (3) (2010) 605–618.

[4] W.M. Wonham, Supervisory control of discrete-event systems, Springer, 2015.
[5] S. Lafortune, Discrete event systems: Modeling, observation, and control, Ann. Rev. Control, Robotics, Autonomous Syst. 2 (2019) 141–159.
[6] W. Wonham, K. Cai, K. Rudie, Supervisory control of discrete-event systems: A brief history, Ann. Rev. Control 45 (2018) 250–256.
[7] C. Ju, H.I. Son, A hybrid systems-based hierarchical control architecture for heterogeneous field robot teams, [Online]. Available: arXiv:2002.08602 [cs. RO] (2020)..
[8] C. Ju, H.I. Son, Multiple uav systems for agricultural applications: control, implementation, and evaluation, Electronics 7 (9) (2018) 162.
[9] C. Ju, H.I. Son, A distributed swarm control for an agricultural multiple unmanned aerial vehicle system, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 233 (10) (2019) 1298–1308..
[10] C. Ju, H.I. Son, Modeling and control of heterogeneous agricultural field robots based on ramadge–wonham theory, IEEE Robotics Autom. Letters 5 (1) (2020) 48–55, https://doi.org/10.1109/LRA.2019.2941178.
[11] R. Zhang, K. Cai, W.M. Wonham, Supervisor localization of discrete-event systems under partial observation, Automatica 81 (2017) 142–147.
[12] J.J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, A. Barrientos, Robots in agriculture: State of art and practical experiences, Service Robots..
[13] P.J. Ramadge, W.M. Wonham, Supervisory control of a class of discrete event processes, SIAM J. Control Optim. 25 (1) (1987) 206–230.
[14] H.I. Son, S. Lee, Failure diagnosis and recovery based on des framework, J. Intell. Manuf. 18 (2) (2007) 249–260.
[15] A. Tsalatsanis, A. Yalcin, K.P. Valavanis, Dynamic task allocation in cooperative robot teams, Robotica 30 (5) (2012) 721–730.
[16] Y. Liu, M. Ficocelli, G. Nejat, A supervisory control method for multi-robot task allocation in urban search and rescue, in: 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, 2015, pp. 1–6.
[17] J. Dulce-Galindo, M.A. Santos, G.V. Raffo, P.N. Pena, Autonomous navigation of multiple robots using supervisory control theory, in: 2019 18th European Control Conference (ECC), IEEE, 2019, pp. 3198–3203.
[18] F.J. Mendiburu, M.R. Morais, A.M. Lima, Behavior coordination in multi-robot systems, in: 2016 IEEE International Conference on Automatica (ICA-ACCA), IEEE, 2016, pp. 1–7.
[19] G.W. Gamage, G.K.I. Mann, R.G. Gosine, Discrete event systems based formation control framework to coordinate multiple nonholonomic mobile robots, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4831–4836, https://doi.org/10.1109/IROS.2009.5354788.
[20] Y. Tatsumoto, M. Shiraishi, K. Cai, Z. Lin, Application of online supervisory control of discrete-event systems to multi-robot warehouse automation, Control Eng. Practice 81 (2018) 97–104.
[21] C. Ju, H.I. Son, Discrete event systems based modeling for agricultural multiple unmanned aerial vehicles: Automata theory approach, in: 2018 18th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2018, pp. 258–260.
[22] X. Dai, L. Jiang, Y. Zhao, Cooperative exploration based on supervisory control of multi-robot systems, Appl. Intell. 45 (1) (2016) 18–29.
[23] S.T. Forschelen, J.M. van de Mortel-Fronczak, R. Su, J.E. Rooda, Application of supervisory control theory to theme park vehicles, Discrete Event Dynamic Systems 22 (4) (2012) 511–540.
[24] Y.K. Lopes, S.M. Trenkwalder, A.B. Leal, T.J. Dodd, R. Groß, Supervisory control theory applied to swarm robotics, Swarm Intell. 10 (1) (2016) 65–97.
[25] A.M. Rahmani, B. Donyanavard, T. Mück, K. Moazzemi, A. Jantsch, O. Mutlu, N. Dutt, Spectr: Formal supervisory control and coordination for many-core systems resource management, ACM SIGPLAN Notices 53 (2) (2018) 169–183.
[26] R.C. Hill, S. Lafortune, Scaling the formal synthesis of supervisory control software for multiple robot systems, in: 2017 American Control Conference (ACC), IEEE, 2017, pp. 3840–3847.
[27] R. Malik, K. Åkesson, H. Flordal, M. Fabian, Supremica–an efficient tool for large-scale discrete event systems, IFAC-PapersOnLine 50 (1) (2017) 5794–5799.
[28] H. Zhang, L. Feng, Z. Li, A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems, IEEE Trans. Autom. Control 63 (10) (2018) 3345–3360, https://doi.org/10.1109/TAC.2018.2793662.
[29] W. Deng, J. Yang, D. Qiu, Supervisory control of probabilistic discrete event systems under partial observation, IEEE Trans. Autom. Control (2019), https://doi.org/10.1109/TAC.2019.2905305, 1–1.
[30] R. Liu, Y. Wang, L. Zhang, An fdes-based shared control method for asynchronous brain-actuated robot, IEEE Trans. Cybern. 46 (6) (2016) 1452–1462, https://doi.org/10.1109/TCYB.2015.2469278.
[31] M. Furci, R. Naldi, A. Paoli, L. Marconi, Robust supervisory-based control strategy for mobile robot navigation, in: E. Menegatti, N. Michael, K. Berns, H. Yamaguchi (Eds.), Intelligent Autonomous Systems 13, Springer International Publishing, Cham, 2016, pp. 121–133.
[32] M. Goorden, C. Dingemans, M. Reniers, J. van de Mortel-Fronczak, W. Fokkink, J. Rooda, Supervisory control of multilevel discrete-event systems with a bus structure, in: 2019 18th European Control Conference (ECC), 2019, pp. 3204–3211. doi:10.23919/ECC.2019.8795835..
[33] C. Hu, C. Hu, D. He, Q. Gu, A new ros-based hybrid architecture for heterogeneous multi-robot systems, in: The 27th Chinese Control and Decision Conference (2015 CCDC), IEEE, 2015, pp. 4721–4726.
[34] M. Milford, G. Wyeth, Hybrid robot control and slam for persistent navigation and mapping, Robotics Autonomous Syst. 58 (9) (2010) 1096–1104.
[35] M. Egerstedt, X. Hu, A hybrid control approach to action coordination for mobile robots, Automatica 38 (1) (2002) 125–130.
[36] Z. Huang, D. Chu, C. Wu, Y. He, Path planning and cooperative control for automated vehicle platoon using hybrid automata, IEEE Trans. Intell. Transp. Syst. 20 (3) (2018) 959–974.
[37] K. Cai, Warehouse automation by logistic robotic networks: a cyber-physical control approach, Front. Inform. Technol. Electron. Eng. 21 (2020) 693–704.
[38] L.V. Alves, P.N. Pena, Secure recovery procedure for manufacturing systems using synchronizing automata and supervisory control theory, IEEE Transactions on Automation Science and Engineering..
[39] F. Mirzaei, A.A. Pouyan, M. Biglari, Automatic controller code generation for swarm robotics using probabilistic timed supervisory control theory (ptsct), J. Intelligent Robotic Syst. 100 (2) (2020) 729–750.
[40] C. Dou, D. Yue, Q.-L. Han, J.M. Guerrero, Multi-agent system-based event-triggered hybrid control scheme for energy internet, IEEE Access 5 (2017) 3263–3272.
[41] C.-X. Dou, B. Liu, Multi-agent based hierarchical hybrid control for smart microgrid, IEEE Trans. Smart Grid 4 (2) (2013) 771–778.
[42] Y. Zheng, Q. Zhao, J. Ma, L. Wang, Second-order consensus of hybrid multi-agent systems, Systems Control Letters 125 (2019) 51–58.
[43] C.G. Cassandras, S. Lafortune, Introduction to discrete event systems, Springer Science & Business Media, 2009.
[44] X. Yin, S. Lafortune, Synthesis of maximally permissive nonblocking supervisors for the lower bound containment problem, IEEE Trans. Autom. Control 63 (12) (2018) 4435–4441.
[45] S. Bak, O.A. Beg, S. Bogomolov, T.T. Johnson, L.V. Nguyen, C. Schilling, Hybrid automata: from verification to implementation, Int. J. Softw. Tools Technol. Transfer 21 (1) (2019) 87–104.
[46] P. Hart, The condensed nearest neighbor rule (corresp.), IEEE Trans. Inform. Theory 14 (3) (1968) 515–516.
[47] H.I. Son, Design and implementation of decentralised supervisory control for manufacturing system automation, Int. J. Comput. Integr. Manuf. 24 (3) (2011) 242–256.