Article

The International Journal of Robotics Research
2024, Vol. 43(14) 2145–2182

DOI: 10.1177/02783649241242812
journals.sagepub.com/home/ijr
Sage

Distributed misbehavior monitors for socially organized autonomous systems

Adriano Fagiolini[1], Gianluca Dini[2], Federico Massa[2], Lucia Pallottino[2] and Antonio Bicchi[2,3]

Abstract

In systems in which many heterogeneous agents operate autonomously, with competing goals and without a centralized planner or global information repository, safety and performance can only be guaranteed by "social" rules imposed on the behavior of individual agents. Social laws are structured in a way that they can be verified just by using local information made available to an agent by a small number of its neighbors. Automobile mobility with traffic rules and logistics robots in warehouses are canonical examples of such "regulated autonomy", but many other fairly-competing autonomous systems are to be expected shortly. In these systems, detecting whether an agent is not abiding by the rules is crucial for raising an alert and taking appropriate countermeasures. However, the limited visibility due to the local nature of the information makes the problem of misbehavior detection hard for any single agent, and only an exchange of information between agents can provide sufficient clues to arrive at a decision. This paper attacks the misbehavior detection problem for a class of socially organized autonomous systems, where the behavior of agents depends on the presence or absence of other neighbors. We propose a solution involving a "local monitor", which runs on each agent and includes a hybrid observer and a set-valued consensus node. Based on whatever visibility is available, it reconstructs a set-valued occupancy estimate of nearby regions and combines it with communicating neighbors to reach a shared view and a mismatch discovery. We provide a formal framework for describing social rules that unify many different applications and a tool to generate code automatically for local monitors. The technique is demonstrated in various systems, including self-driving cars, autonomous forklifts, and distributed power plants.

Keywords

Misbehavior detection, security, robotics, set-valued consensus, self-driving cars, automatic generation

Received 16 January 2022; Revised 30 January 2024; Accepted 6 March 2024
Acting Editor In Chief: Tim Barfoot
Senior Editor: Dongjun Lee
Associate Editor: Roderich Gross

1. Introduction

Until a few decades ago, industrial manipulators operating in a closed, structured, completely known and human-free environment were the autonomous systems par excellence. More recently, multi-robot systems composed of identical copies of a given prototype robot have been developed to solve various service tasks, such as optimal spatial deployment, patrolling and surveillance, etc. In these systems, robots interact with one another by sharing local information between neighbors through distributed protocols, which allow for time synchronization (Sinopoli et al., 2003), global agreement (Olfati-Saber et al., 2007), space coverage (Cortes et al., 2004), and motion in formation (Fathian et al., 2018), to name a few. While being very efficient, these protocols often assume that interaction is encoded through signals with rigid formats and that robots are used within known environments.

Nowadays, several promising robotic platforms have become available, either with fixed or mobile bases, capable of collaborating with humans (Ajoudani et al., 2018) and each other (Kabir et al., 2021; Nguyen et al., 2019) in lesser-known and dynamically changing shared workspaces. Consequently, the challenge will soon be to safely and efficiently organize the cooperation and coexistence of many heterogeneous robots or agents, built by different manufacturers, with different intelligence skills and perhaps conflicting goals. These robots will be endowed with

[1]Mobile & Intelligent Robots @ Panormus Laboratory (MIRPALab), Department of Engineering, University of Palermo, Palermo, Italy
[2]Department of Information Engineering and the Research Center "E. Piaggio", Università di Pisa, Pisa, Italy
[3]Italian Institute of Technology, Genova, Italy

Corresponding author:
Adriano Fagiolini, Mobile & Intelligent Robots @ Panormous Laboratory (MIRPALab), Department of Engineering, University of Palermo, Viale delle Scienze, Building 10, Palermo 90128, Italy.
Email: fagiolini@unipa.it

diversity-enhanced autonomy (Ayanian, 2019) and empowered by algorithms for, e.g., online parallel planning under uncertainty (Cai et al., 2021), efficient multi-image feature matching (Serlin et al., 2020), decentralized navigation via Hamiltonian coordination primitives (Mavrogiannis and Knepper, 2021), target estimation and localization (Bourne et al., 2020), and collective swarm formation (Dong and Sitti, 2020). Their robustness will also be verifiable using real data sets now available (Agarwal et al., 2020; Behley et al., 2021; Pitropov et al., 2021). Near future scenarios will most likely involve open systems composed of many heterogeneous robots cooperating with each other and interacting with the environment. Robot coexistence will require a paradigm shift in organizing cooperation within a set of "social" rules, to be imposed on the behavior of each individual robot and structured in such a way that they can be verified simply by using local information (Shoham and Tennenholtz, 1995); these rules are to be designed off-line or learned online, but will lead to the autonomous robots involved being regarded as members of a "society" (Bicchi et al., 2010; Ko et al., 2021; Pallottino et al., 2007).

Probably the most widespread example of a *socially organized* system are the flocks of vehicles on the road: they include heterogeneous cars, trucks, and motorbikes that enter and exit the flock at any time, are driven by human drivers with different driving styles (Lefèvre et al., 2015a, 2015b; Kuderer et al., 2015) or are in autonomous mode (Caporale et al., 2019; Kabzan et al., 2020), anticipate or react quickly to adverse conditions (Pedone and Fagiolini, 2020; Radwan et al., 2020), locally self-organize in road platoons to optimize performance (cf. Morbidi et al. (2013); Besselink and Johansson (2015) and the pilot projects SARTRE (Solyom and Coelingh, 2013) and COMPANION (Eilers et al., 2015)), and, most crucially, they are supposed to obey a shared cooperation protocol entailed by the common traffic rules. Other examples are forklifts and similar robots used for logistics in automated warehouses, which obey a set of coordination rules to reach their destinations and avoid collisions. Yet, many more examples of open, heterogeneous, reconfigurable, and competing autonomous systems can be expected soon.

Unfortunately, greater flexibility of interactions allowed in a socially organized robotic system moving in a shared physical environment goes hand in hand with new, stealthier types of threats. In a renowned experiment conducted in 2016, the navigation subsystem of a car connected to the Internet was remotely hijacked to take full control of the vehicle (Greenberg, 2016); in a system as large as a social set of robots cooperating on the basis of rules and without a centralized controller, malfunctioning of a single individual, due to spontaneous failure or tampering, can produce effects that amplify in the society (Bossens et al., 2022). Relying on the assumption that all robots belonging to the society are systems robust to failures or attacks by design is unrealistic. In the absence of a central monitoring system, it is necessary to provide online misbehavior detection mechanisms to face these threats in a distributed way (Fagiolini and Bicchi, 2013; Gupta et al., 2006; Pasqualetti et al., 2013). To this end, it is advocated that robots monitor one another and try to determine whether or not their neighbors, with whom they interact, follow the social rules. The challenge is to cope with partial information about each robot, due to the limited visibility of on-board sensors, and misbehavior of any component of a neighboring robot, from its sensors to software and actuators. In addition, detecting misbehavior may be too difficult for a single robot, while sufficient clues for effective detection can only be found through information exchange.

In this context, traditional approaches for fault detection and diagnosis cannot be directly applied for many reasons. Firstly, most of the literature to date focuses on single robot scenarios (Pettersson, 2005; Zhuo-Hua et al., 2005). Only recently, some studies have examined fault detection in the more challenging case of multiple distributed robots (Khalastchi and Kalech, 2018, 2019). Secondly, robots have varying degrees of autonomy, operate in different and more dynamic physical environments, and thus may experience a broader spectrum of failures, including those induced by an arbitrarily intelligent or byzantine actor (Ashkenazi et al., 2023; Pasqualetti et al., 2012) that can tamper with the robot architecture at any level. The first relevant results, using the geometric approach, that addressed the problem for linear dynamical systems are the seminal works by Pasqualetti et al. (2012, 2013); the ideas have been partially extended for linear switching systems more recently (Duz et al., 2018; Zattoni et al., 2017). The resilience of multi-robot systems to communication failures has been addressed successively in deterministic contexts (Fagiolini et al., 2015; Kaur et al., 2021; Strobel et al., 2018) and, very recently, with probabilistic robot interaction, exploiting the control-theoretic notion of left-invertibility (Wehbe and Williams, 2021a, 2021b). A data-driven approach for detecting anomalies through statistical classifiers was proposed in the early work by Lau et al. (2011). Recently, an approach based on the adaptive immune system has been applied to robot swarms, which is capable of distinguishing abnormal behavior from the most frequent ones (Tarapore et al., 2019). Still, the literature lacks a general approach to codifying social rules in the components of a social robot's model and a systematic way to design, given the rules of a society, local processes for monitoring and collecting evidence of cooperativeness or non-cooperativeness from each robot and for social agreement; the latter is crucial to trigger, if necessary, any counteraction in response to the detection of a robot not following the rules, seen as an intruder of the society.

This paper addresses the problem of distributed misbehavior detection in systems where cooperation between robots is described by a set of social rules and robots themselves can be heterogeneous in their dynamics, actuators, and controllers. A new solution is proposed consisting of a distributed mechanism for monitoring and agreement, which applies to a class of socially organized autonomous systems

where the behavior of each robot depends, in a general way, on the presence or absence of other nearby robots. After introducing a formal hybrid modeling framework to describe these systems, the paper presents the theoretical results that enable the construction of a local monitor, an essential element of the proposed solution, which is a module running on each robot and trying to determine whether or not any neighboring robot follows the social rules. Once the behavior of a given robot is measured, the challenge of each monitor is to reconstruct, by whatever visibility is available, the possible inputs that may have generated this behavior in accordance with the social rules. Through this "inversion" of the hybrid model, the local monitor reconstructs a set-valued estimate of the occupancy of all regions near the robot. Then, each monitor becomes a node of a set-valued consensus algorithm and shares the estimate with other nearby monitors, in order to obtain a socially agreed view of the occupancy, discover any mismatched behavior of the robot, and activate any countermeasures if necessary.

An appealing feature of the solution is that the local monitor modules are obtained from software components already present in the architecture of each agent, and thus, its complexity can be determined. In addition, a software tool to automatically generate the code of local monitors and consensus nodes for all agents is provided. The approach validity is successfully demonstrated in various systems, including self-driving cars, autonomous forklifts, and distributed power generation plants. Simulations have been developed for all cases, and an experimental implementation has been done on real forklifts.

The document is organized as follows. Section 2 describes the state of the art and outlines the contributions of the present work. Section 3 formally presents the cooperation protocol model for socially organized robots, Section 4 explains the challenges and derives the local monitor components, and Section 5 focuses on the social agreement and describes the consensus node. The following two sections, 6 and 7, detail the application of the proposed method to three case studies. Section 8 summarizes the work achievements and contains a discussion on various technological aspects and observations. Five appendices complement the document as follows: Appendix A describes multimedia extensions. Appendix B demonstrates the event estimation map with incomplete and time-varying visibility. Appendix C illustrates the finite-time convergence of the set-valued consensus algorithm. Appendix D is a brief guide to the software tool. Finally, Appendix E summarized the key symbols used for the main objects involved in the work.

2. Related work and novelty

2.1. State of the art

Fault detection and diagnosis for robotics is a relatively new topic. Since robots are physical systems with varying degrees of autonomy operating in diverse and dynamic physical environments, the detection of anomalies or faults poses constraints that challenge traditional approaches. Most literature to date relies upon traditional approaches to fault detection and focuses on single robot scenarios (cf., e.g., Pettersson (2005); Zhuo-Hua et al. (2005)). In a recent survey, Khalastchi and Kalech (2019, 2018) looked at the distributed multi-robot domain and provided an overview of the types of faults and their impact on these systems. After outlining the requirements and challenges involved, the survey indicates qualitatively which current approaches can be used and lists some existing control schemes (Christensen et al., 2009; Li and Parker, 2007), all designed for specific contexts, that guarantee a certain level of robustness. Along the line, Parker (1998) has considered more general scenarios and rooted the main ideas towards the development of fully distributed, behavior-based architectures for the fault-tolerant cooperative control of robot teams. Stavrou et al. (2016) have described a learning-based technique that adapts a set of parameters after the robot is deployed in the target environment but still applies to a single-robot system.

General misbehavior detection, which includes simple failures, must address a broader spectrum of anomalies, including those induced by an arbitrarily intelligent or Byzantine actor that can tamper with the robot's architecture at both logical and physical levels. Specifically, Bossens et al. (2022) emphasized that robot teams used for long-term deployment must be resilient to disruptions, whether due to sensor and motor failures, weather conditions, or even adversary cyber attacks. Some disruptions can be anticipated during the design phase, which is the driving idea of the work by Zheng et al. (2016), which successfully proposed a multi-layered co-design approach for securing cyber-physical systems, thus enabling the implementation of resilient-by-design architectures; the framework is valid for linear systems and combines control-theoretic methods at the functional layer of the cyber-physical system with cybersecurity techniques at the embedded platform layer. In addition, Zhou and Tokekar (2021) outlined that current strategies for coordination and planning of multi-robots are vulnerable to adversarial attacks, revealing that the development of schemes resilient to such attacks and robust to uncertainty is still an open research area. It is highlighted again by Bossens et al. (2022) that when the system becomes large and complex interactions are present, anomalies can only be detected through local measurements of each team member but still need a team-wide view before any counteraction is taken.

A variety of works has focused on fault tolerance or, more generally, on the resilience of multiple robots to communication failures. In this regard, Pasqualetti et al. (2012, 2013) proposed in their seminal work a new technique based on the geometric approach that applies to linear dynamic systems. Later on, Duz et al. (2018) partially extended the results to detect stealthy attacks in switching linear systems, and Zattoni et al. (2017) proposed a technique with a prospective application to the field using the

notion of controlled invariance. Other works have tackled the problem in deterministic contexts (Ashkenazi et al., 2023; Fagiolini et al., 2015; Kaur et al., 2021), and only recently for systems with probabilistic robot interaction using the notion of left invertibility (Wehbe and Williams, 2021a, 2021b). In parallel, Strobel et al. (2018) proposed ensuring security through blockchain technology to tolerate the sharing of erroneous information by Byzantine robots. Lee and Hauert (2023) have used sparsity in robot swarms to establish trustworthiness in real implementations. Leaderless consensus in teams with linear dynamics has been achieved through the work by Zeng and Chow (2014), which proposed a resilient distributed control law using rollback and excitation recovery.

Few works have focused explicitly on motion misbehavior in multi-robot systems, or, if they do, they assume simple models and hypotheses. To begin with, Ashkenazi et al. (2023) examine motion misbehavior in robot swarms, where each robot is assumed to follow a motion policy that makes it move on a grid; it can cope with benign and Byzantine malfunctions, the former of which are tolerated through a forgive-and-forget strategy. Trinh and Nguyen (2023) have proposed a fault-tolerant model predictive controller to achieve leader-following formation with unicycle robots. In contrast, Guo et al. (2018) have presented a method in which robot models can be nonlinear and consist of various components that can be fault, but interaction with neighboring robots is not represented. A promising solution has been developed by Ferraro and Scordamaglia (2023), in which a set-valued approach is used to determine faults in a remotely controlled vehicle; this involves generating a feasible trajectory for the robot subject to unknown but limited external disturbances. Yet, so far, the method can only recover a single optimal trajectory and applies to the context of a single robot without general interaction with the environment or other robots. Tarapore et al. (2015, 2019) have developed a noteworthy approach based on the adaptive immune system and applied it to robot swarms. The method does not require knowledge of normal behavior patterns in advance. Instead, it uses the ability to tolerate certain behavior deviations to recognize frequent behaviors as normal and rare ones as abnormal and, thus, possibly faulty. Although this feature gives the method considerable flexibility, it also makes it unsuitable, at least in the current form, in systems where social rules specify allowed behaviors, and it is not true that a dynamic pattern executed by many agents is correct. Furthermore, the method assumes a one-to-one mapping between the presence/absence of other neighboring robots and the type of behavior a robot performs, whereas, in a socially organized system, robot interaction may involve general binary conditions, including combinations of logical negation, product, and sum, altogether.

Furthermore, an important property often exploited to detect faults or spurious inputs is the invertibility of a model with respect to a set of inputs (Millérioux and Daafouz, 2007; Sain and Massey, 2002; Vu and Liberzon, 2008). This property ensures that such inputs can be reconstructed by using only output measurement. Methods for explicit construction of input-state observers have been proposed for linear models (cf., e.g., Sundaram and Hadjicostis (2008)) and even used in robotics and automotive for various purposes (Pedone and Fagiolini, 2020; Trumić et al., 2022). In the present context, a robot monitoring another member of the society generally has to estimate signals generated by the presence/absence of other neighboring robots that are outside its sensing range. Due to the nature of social rules, however, scenarios in which two neighboring robots have different positions but are in the same area may not be distinguishable. Therefore, it is more plausible that the inversion maps single-point outputs to continuous sets representing areas where the existence or absence of a neighboring robot is required. In this regard, solutions for input and state reconstruction via set-valued observers (Doyen and Rapaport, 2001; Lin et al., 2003; Monteriu et al., 2007; Shamma and Tu, 1999; Xu et al., 2017) do not apply because they assume linear or input-affine dynamics. In contrast, the aim here is to address a class of hybrid systems, with nonlinear jump conditions and dynamics, that is sufficiently large to accommodate the implementation of social rules. The first two methods return the most probable value of a single state at any given time, whereas dealing with general misbehavior, including attacks, requires the estimation of complete occupancy maps of an observed robot's neighborhood. Also, they require persistent excitation and active disturbance rejection, which implies a modification of the given social rules. Yet, they are very effective in specific cases involving linear models so that they could complement the tools of the architecture presented here.

As will be shown later, the model of each social robot comprises a finite state machine, or automaton, whose temporal evolution is event-driven. Events are related to the occurrence of appropriate general conditions about the presence/absence of neighboring robots. Reconstruction of events from partial knowledge of a monitoring robot is, hence, an essential feature to be achieved. In this regard, observability and diagnosability for event-driven systems have been studied by, e.g., Cassandras and Lafortune (2006), Yoo and Lafortune (2002), Özveren and Willsky (1992), and Ramadge and Wonham (1989, 1987). Overall, the social robot model is essentially hybrid (Goebel et al., 2009), in that it also includes a time-driven dynamic that describes the physical behavior of the robot. Balluchi et al. (2002), Zad et al. (2003), and Fourlas et al. (2002) have studied the two properties mentioned above for hybrid models in the linear or input-affine settings. Yet, it should be noted that the number of inputs to each robot's hybrid model changes over time, as does the number of neighboring robots, and that the capacity of a monitoring robot to detect the occurrence of specific events changes also with its instantaneous visibility. The time-varying topology of a robot's interaction and the partial event visibility of a monitoring robot, together with the hybrid and possibly

nonlinear nature of the model, prevent the use of standard tools for observability, invertibility, and diagnosability; these tools assume instead constant visibility and interaction and that this information is fully known a priori.

A final point concerns the establishment of a socially accepted global view on the cooperativeness of a given robot. In distributed systems, consensus algorithms are often used (Jadbabaie et al., 2003; Olfati-Saber et al., 2007). Such algorithms have linear dynamics and allow participating agents to agree on the value of a physical quantity described by a real number or vector. As noted earlier, when attempting to invert the model of a social robot and reconstruct its inputs, one obtains continuous sets representing areas where neighboring robots should or should not be. Specifically, local monitors and consensus nodes, described below, must attempt to reconstruct and ultimately refine estimates of unknown inputs that explain the behavior of a commonly observed agent; the results of this process are continuous points and sets that describe regions where the presence/absence of a neighbor is necessary to prove the cooperativeness of the observed robot, which makes linear consensus strategies inapplicable here.

2.2. Contribution

As pointed out above, the scientific literature lacks some essential elements that would enable the practical realization of a robot society. In this context, this paper contributes to the state of the art in a number of ways, as described below.

Firstly, a general approach to codifying social rules and incorporating them into the components of a social robot model is currently missing. In this regard, the paper provides a formal framework for describing a class of socially organized autonomous systems in which the behavior of each robot is governed by a set of rules that depend on the presence or absence of other neighboring robots; the notion of neighborhood is defined through proximity maps that are functions of the states and intentions of any interacting robots. Accordingly, a cooperative robot is described by a model that includes its physical time-driven dynamics, low-level controllers that allow the execution of specific maneuvers, the event-driven dynamics of social interaction, the perception of the environment and nearby robots, the encoding of inputs into events, etc.

Secondly, while many approaches for distributed misbehavior detection in discrete-event systems, hybrid linear systems, and some classes of switching linear systems are available, a general methodology applying to a class of nonlinear multi-robot systems with social interaction is absent. In this respect, the paper proposes a systematic way to design, given the rules of the society, local processes for monitoring and collecting evidence of cooperativeness or non-cooperativeness from each robot and for later reaching social agreement. Specifically, social consensus is enabled through a distributed algorithm run by cooperating monitors, whose construction exploits the results recently developed in the field of set-valued Boolean iterative maps (Fagiolini et al., 2015).

An appealing feature of local monitors is that their construction is based on models and software components already present in the architecture of each robot. More precisely, the equations that describe the robot's physical dynamics, as well as the low-level control software that sends signals to the actuators, the one that updates the automaton and the one that encodes the binary conditions that indicate the occurrence of events, are appropriately exploited as black boxes, to generate predictor/corrector components automatically. This feature implies that their complexity can be determined, and their execution is typically already feasible on the robots.

A noteworthy theoretical result regards the construction of a nondeterministic observer of the discrete state of the system. Such observer is obtained through a suitable factorization of the event detection conditions, which takes into account the current partial visibility of the monitoring robot. It extends existing solutions (Cassandras and Lafortune, 2006) insofar as it shows: (1) how, given a discrete-event model, a nondeterministic observer can be derived even with partial and time-varying visibility of events; (2) the absence of the need to explicitly construct the observer, as calculations can be performed through a black box execution of the robot's automaton.

Lastly, the paper provides a software tool to automatically generate the code for local monitors and consensus nodes for all robots. The validity of the approach is successfully demonstrated in various systems, including self-driving cars, autonomous forklifts, and distributed power generation plants. The paper illustrates the proposed methodology through simulations for all case studies and an experimental implementation on industrial forklifts.

2.3. Nomenclature

The section ends by defining the nomenclature used in the text. Firstly, given a set A, the following notation is used: $\mathrm{Pow}(A)$ denotes the *power set* of A, i.e., the set of all possible subsets (including the empty set and the set A itself) that can be constructed from elements of A; $\mathbf{1}_A(x)$ denotes the *Indicator function* of A returning 1 if, and only if, $x \in A$; if the set $A \subseteq X \times Y$, where X and Y are two sets, $\mathrm{Proj}_X(A)$ is the *projection or restriction* of A to the set X, i.e., $\mathrm{Proj}_X(A) = \{x \mid (x, y) \in A, \text{with } x \in X \text{ and } y \in Y\}$; if $a \subseteq A$, $\overline{a} = A \setminus a$ is the *set complement* of a for A.

Moreover, given a vector space $\mathcal{Q}$, $\mathrm{Tan}(\mathcal{Q})$ is the *space tangent* to $\mathcal{Q}$ and $F_{[t_1,t_2]}(\mathcal{Q})$ is the *space of functions* defined from the time interval $[t_1, t_2]$ to $\mathcal{Q}$.

Furthermore, $\mathbb{B} \stackrel{\text{def}}{=} \{0, 1\}$ is the binary domain, $\oplus$, $\otimes$, and $\neg$ are the logical sum (*or*), product (*and*) and negation (*not*), respectively; also, given a suitable domain set X, a logical-valued function $b : X \to \mathbb{B}$, associated with a specific

condition on X, is said to be *active* if b returns 1 if, and only if, such condition is met.

In addition, given a signal $a(\tau)$, where τ is a continuous time, $\tilde{a}(t_0, t)$ denotes the *history* of $a(\tau)$ during the time interval $[t_0, t)$, i.e., a buffer memorizing the evolution of the signal during such interval; when the first argument is understood, the history is shortened as $\tilde{a}(t)$. Also, $a[k]$ denotes the signal $a(\tau)$ sampled at discrete times $t = t_k$, i.e., $a[k] = a(t_k)$.

Finally, given a constant $\epsilon > 0$, two signals, $a(t)$ and $b(t)$, are ϵ-*similar* in a time period $\mathcal{T} = [t_1, t_2)$, if $\|a(t) - b(t)\| \leq \epsilon$ for all $t \in \mathcal{T}$, where $\|\cdot\|$ is a norm over the signal domain.

3. Protocols for socially organized agents

Consider n robotic agents, $\mathcal{A}_1, \ldots, \mathcal{A}_n$, sharing a state-space or *environment* $\mathcal{Q}$. A distributed cooperation protocol $\mathcal{P}$ is a formal description of the constitutive elements of each agent, specifying their perception, dynamics and control, actuation, and the rules by which these elements are interconnected so as to determine their behavior and interaction with neighbors. Socially organized agents are agents sharing a cooperation protocol $\mathcal{P}$. The key symbols of the protocol are summarized in Table 7 in Appendix E.

3.1. Explanation and intuitive introduction

Intuitively, one can introduce $\mathcal{P}$ by referring, e.g., to a system of self-driving or human-driven vehicles traveling on a motorway. In this context, the goal of each motorcycle, car, or truck $\mathcal{A}_i$ on the road is to reach its destination while avoiding collisions, which is supposed to be achieved cooperatively by following a set of traffic rules. By looking from a top-down view, rules are based on safety- and performance-critical *events* and specify a number of *maneuvers*. Events affecting a vehicle $\mathcal{A}_i$ are, e.g., the presence of a slow car in the front lane or a fast motorcycle approaching from the next lane; they are triggered by binary conditions depending on the current *state* of $\mathcal{A}_i$ and those of the vehicles nearby. To perceive the presence or absence of such other vehicles and map out the road ahead, each $\mathcal{A}_i$ has cameras, lidars, ultrasonic detectors, and other sensors allowing it to see or measure the states of other neighboring vehicles within a *visibility* region; this is part of the environment that, at any current time, is not hidden by any vehicles and that is within a maximum distance from $\mathcal{A}_i$. A subset of the visibility region contains the information that directly affects the behavior of $\mathcal{A}_i$ and represents the instantaneous proximity space or *neighborhood* of the vehicle; correspondingly, all nearby vehicles affecting the behavior of $\mathcal{A}_i$ are its current *neighbors*. In addition, the maneuvers that are accepted by a traffic protocol are, e.g., overtaking a preceding vehicle, slowing down, moving to the next free lane, or simply continuing fast along the current lane, are characterized by a specific type of trajectory that $\mathcal{A}_i$ is required to track. Each time an event is detected, vehicle $\mathcal{A}_i$ must assess whether the current maneuver should be updated and, consequently, the type of trajectory being tracked to be changed. The correct execution of each maneuver is ensured by an onboard *controller* (either automatic or human) and an underlying actuation system, which implements a feedback (and possibly also feedforward) stabilizing control action through the steering and pedal/brake systems. The control action applied at any time depends on the current maneuver, as well as the specific mechanical structure, size, and inertia of the ith vehicle, in a word, its physical *dynamics*. All these elements ultimately determine the temporal evolution of the ith vehicle state or *behavior*. Such a state is formed of the linear and angular positions and the corresponding speeds, of $\mathcal{A}_i$ and is defined over the domain $\mathcal{Q}$ of all possible vehicle states on the road.

More in detail, from a bottom-up view, the continuous state of each $\mathcal{A}_i$ is a vector $q_i \in \mathcal{Q}$, describing the current configuration of the vehicle along the motorway and evolving based on a time-driven *dynamics* map f_i; its discrete state is a scalar $\sigma_i \in \Sigma_i$, describing the vehicle's current maneuver and being updated through an event-driven *automaton* map δ_i; the set of all maneuvers that are admitted by the traffic protocol are collected in a finite *discrete state set* Σ_i. The region of $\mathcal{Q}$ that is reachable by the lidars and cameras of $\mathcal{A}_i$ is represented by a *visibility* map V_i; the subset of V_i representing the proximity space or *neighborhood* $\mathcal{N}_i$ of the ith vehicle is formed of the union of distinct subregions or *topologies*, among which: κ_i are described by maps $\eta_{i,j}(q_i)$ denoting regions close to and moving with the vehicle; other h_i are described by maps $\eta^*_{i,j}$ denoting constant regions used to delimit the physical environment $\mathcal{Q}$ or describe boundary values for each q_i. Examples of topologies moving with the ith vehicle are the front, rear, and the two lateral areas with respect to the vehicle configuration q_i; those that are constant include the left and right roadside curbs. Moving topologies concur to defining the ith current neighborhood as $\mathcal{N}_i = \cup_{j=1}^{\kappa_i} \eta_{i,j}(q_i)$, which let us also define the ith *neighbor set* $N_i = \{\ell \mid q_\ell \in \mathcal{N}_i\}$ and, then, *input set* $I_i = \{q_\ell | \ell \in N_i\}$ as the set of indices and set of configurations, respectively, of all neighboring vehicles affecting $\mathcal{A}_i$. All such vehicles also represent the *neighbors* of $\mathcal{A}_i$.

The presence of a nearby vehicle $\mathcal{A}_\ell$ in each (i,j)th topology as well as the proximity of $\mathcal{A}_i$ to any constant region, are mapped into binary values $s_{i,j}$ via an *encoder* map s_i; the activation or non-activation of each $s_{i,j}$ are used to *encode* the detection conditions of *events* $e^{i,j} \in E_i$, where E_i is an *alphabet of events*, that are relevant for the traffic protocol $\mathcal{P}$. Specifically, the calculation of every event $e^{i,j}$ is codified as a binary product of a subset of the encoder map components $s_{i,j}$, either taken with their positive or negative values; this is specified by two index sets, $\gamma_{i,j}, \gamma^*_{i,j} \subseteq \{1, \ldots, \kappa_i + h_i\}$ indicating which components are to be multiplied with each other and whether their positive

or negative values are relevant. As an example, the event that triggers a change of maneuver/trajectory that forces the ith vehicle to overtake a slower one $\mathcal{A}_\ell$ in the front is detected by the simultaneous presence of the slower vehicle in the front lane area and the absence of other vehicles in the next lane area. All such index sets are collected in an *encoder index set* Γ_i. Finally, depending on the maneuver/trajectory to be executed, each vehicle has an onboard controller that continuously generates appropriate steering and pedal/brake commands; precisely, this is obtained by the ith *decoder* map $u_i \in \mathcal{U}_i$, with $\mathcal{U}_i$ the set of permissible control actions, which directs the evolution of the vehicle's dynamics f_i. The overall behavior of the ith vehicle is mathematically described by the hybrid model $\mathcal{H}_i$ that will be described formally below, the components of which are illustrated in Figure 1 and can be implemented in the simulator described in Appendix D.

3.2. Formal definition

One can now turn the attention to the case of n generic agents sharing an environment $\mathcal{Q}$. Letting the continuous state of each $\mathcal{A}_i$ be $q_i \in \mathcal{Q}$ and its discrete state be σ_i, $\mathcal{P}$ is formally defined as follows:

Definition 1. A distributed cooperation protocol $\mathcal{P}$ *consists in the specification, for each* $\mathcal{A}_i$, *of an octuple*

$$\mathcal{P}_i := \{T_i, V_i, E_i, \Gamma_i, \Sigma_i, \delta_i, u_i, f_i\},$$

whose first element is:

- $T_i = \{\eta_{i,1}, \ldots, \eta_{i,\kappa_i}, \eta^*_{i,1}, \ldots, \eta^*_{i,h_i}\}$ *is a* topology *set, with each* $\eta_{i,j} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q})$ *describing a nearby region of* $\mathcal{Q}$ *where the presence or absence of other agents affects* $\mathcal{A}_i$, *and with each* $\eta^*_{i,j} \in \mathrm{Pow}(\mathcal{Q})$ *describing a region independent of* q_i *and that represents a boundary of* $\mathcal{Q}$.

This first object is essential to define a few further concepts: the ith proximity space or neighborhood $\mathcal{N}_i = \cup_{j=1}^{\kappa_i} \eta_{i,j}(q_i)$, neighbor set $N_i = \{\ell | q_\ell \in \mathcal{N}_i\}$, *and* input set $I_i = \{q_\ell \,|\, \ell \in N_i\}$. *Accordingly, an agent* $\mathcal{A}_\updownarrow$ *is a neighbor of* $\mathcal{A}_i$ *(or* $\mathcal{A}_i$*-neighbor for short) if* $\ell \in N_i$; $n_i = \mathrm{card}\,(I_i)$ *is the number of neighbors of* $\mathcal{A}_i$. *Also, one can introduce the ith encoder map as* $s_i : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}^{\kappa_i + h_i}$, *whose jth component* $s_{i,j}$ *is active, for* $j \le \kappa_i$, *when any neighbor* $\mathcal{A}_\updownarrow$ *is in the jth topology* $\eta_{i,j}$, *and, for* $j > \kappa_i$, *when the ith state is in the jth topology* $\eta^*_{i,j-\kappa_i}$; *in formula, this is obtained as*

$$s_{i,j} : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}$$
$$(q_i, I_i) \mapsto \begin{cases} \oplus_{q_\ell \in I_i} \mathbf{1}_{\eta_{i,j}(q_i)}(q_\ell) & j \le \kappa_i. \\ \mathbf{1}_{\eta^*_{i,j-\kappa_i}} & j > \kappa_i \end{cases} \tag{1}$$

The remaining seven elements of $\mathcal{P}_i$ *represent the following further objects:*

- $V_i : \mathcal{Q}^n \to Pow(\mathcal{Q})$ *is a* visibility *map, describing the region of* $\mathcal{Q}$ *that can be seen by the sensors onboard* $\mathcal{A}_i$ *and where the state of any present agent can be measured;* V_i *is assumed to be a superset of* $\mathcal{N}_i$, *i.e.,* $\mathcal{N}_i \subseteq V_i(q_1, \ldots, q_n)$, *which implies that the ith neighborhood is entirely known to* $\mathcal{A}_i$ *through its sensors;*
- $E_i = \{e^{i,1}, \ldots, e^{i,v_i}\}$ *is a finite* alphabet of events;
- $\Gamma_i = \{\gamma_{i,1}, \gamma^*_{i,1}, \ldots, \gamma_{i,v_i}, \gamma^*_{i,v_i}\}$ *is an* encoder index *set, with each pair* $(\gamma_{i,j}, \gamma^*_{i,j}) \subseteq \{1, \ldots, \kappa_i + h_i\}^2$ *describing how the* (i,j)*th event* $e^{i,j}$ *is recognized, by using the ith encoder map* s_i, *via the* detector *map*

$$e_i : \mathbb{B}^{\kappa_i + h_i} \to Pow(E_i)$$
$$s_i \mapsto \left\{ e^{i,j} \in E_i \middle| \left(\otimes_{\ell \in \gamma_{i,j}} s_{i,\ell} \right) \otimes \otimes \left(\otimes_{\ell \in \gamma^*_{i,j}} \neg s_{i,\ell} \right) \right\}; \tag{2}$$

- $\Sigma_i = \{\sigma^{i,1}, \ldots, \sigma^{i,r_i}\}$ *is a finite* discrete state set;
- δ_i: $\Sigma_i \times Pow(E_i) \to \Sigma_i$ *is a deterministic* automaton *describing, for each ith pair of discrete state* σ_i *and detected event set* e_i, *the next discrete state* $\sigma_i' \in \Sigma_i$;
- $u_i : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to \mathcal{U}_i$ *is a* decoder *map, describing, for each triple of state* q_i, *discrete state* σ_i, *and input set* I_i, *the control action to be applied;*
- $f_i : \mathcal{Q} \times \mathcal{U}_i \to Tan(\mathcal{Q})$ *is a* dynamics *map, where* $\mathcal{U}_i$ *is the set of admissible inputs and* $Tan(\mathcal{Q})$ *is the space tangent to* $\mathcal{Q}$, *describing, for each ith pair of state* q_i *and input* u_i, *the instantaneous motion direction* $\dot{q}_i \in Tan(\mathcal{Q})$. ◆

Note that index i in $\eta_{i,j}$ indicates that each $\mathcal{A}_i$ may have a different jth topology. The argument of $\eta_{i,j}(q_i)$ is redundant, yet it is kept to recall its dependency on the ith state. Note also that the co-domain of e_i is $\mathrm{Pow}(E_i)$ to account for the general case when events $e^{i,j}$ can simultaneously occur.

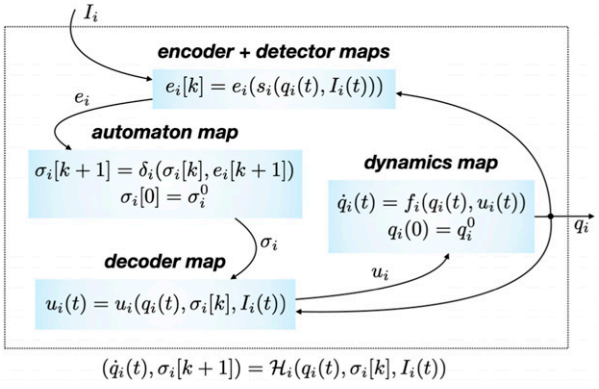

Figure 1. Illustration of the model of a cooperative robot $\mathcal{A}_i$.

3.3 Temporal behavior of a cooperative $\mathcal{A}_i$

By using the elements of each $\mathcal{P}_i$, the temporal behavior of $\mathcal{A}_i$ can be described as follows. Let $t \ge 0$ be the continuous

time and let $t = t_k$, for $k = 0, 1, \cdots$, be the kth instant at which the ith detector map e_i recognizes a new event. The ith continuous state q_i evolves, from an initial value $q_i^0 \in \mathcal{Q}$, according to the ODE

$$\dot{q}_i(t) = f_i(q_i(t), u_i(t)), \text{with } q_i(0) = q_i^0, \tag{3}$$

while the ith discrete state σ_i is updated, from an initial value $\sigma_i^0 \in \Sigma_i$, according to the iterative rule

$$\sigma_i[k+1] = \delta_i(\sigma_i[k], e_i[k+1]), \text{with } \sigma_i[0] = \sigma_i^0. \tag{4}$$

Based on the latest updated discrete state $\sigma[k]$ and the current input set $I_i(t)$, the decoder map applies the control action $u_i(t) = u_i\,(q_i(t), \sigma_i\,[k], I_i(t))$, which, plugged into (3), yields the *controlled dynamics* map

$$\begin{aligned}\dot{q}_i(t) &= f_i(q_i(t), u_i(q_i(t), \sigma_i[k], I_i(t))) = \\ &= f_i^*(q_i(t), \sigma_i[k], I_i(t));\end{aligned} \tag{5}$$

Similarly, based on the current ith state $q_i(t)$ and input set $I_i(t)$, the encoder map returns the vector $s_i(t) = s_i\,(q_i(t), I_i(t))$, which, inserted in the detector map, yields $e_i(t) = e_i\,(s_i(t))$, which in turn is plugged into (4), thus giving the *closed-loop automaton* map

$$\begin{aligned}\sigma_i[k+1] &= \delta_i(\sigma_i[k], e_i(s_i(q_i(t), I_i(t)))) = \\ &= \delta_i{}^*(q_i(t), \sigma_i[k], I_i(t)).\end{aligned}$$

Introducing the ith *hybrid dynamic* map

$$\begin{aligned}\mathcal{H}_i : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} &\to \text{Pow}(\mathcal{Q}) \times \Sigma_i \\ (q_i, \sigma_i, I_i) &\mapsto \begin{pmatrix} f_i{}^*(q_i, \sigma_i, I_i) \\ \delta_i^*(q_i, \sigma_i, I_i) \end{pmatrix},\end{aligned}$$

the ith complete state represented by the pair $(q_i, \sigma_i) \in \mathcal{Q} \times \Sigma_i$ evolves according to the dynamic model

$$\begin{pmatrix} \dot{q}_i(t) \\ \sigma_i[k+1] \end{pmatrix} = \mathcal{H}_i(q_i(t), \sigma_i[k], I_i(t)), \tag{6}$$

with $q_i(0) = q_i^0$ and $\sigma_i[0] = \sigma_i^0$.

Having derived the above model, for a given history $\tilde{I}_i(0,t)$ of the input set I_i, one can now define the ith *behavior* as the solution $\tilde{q}_i(0,t) = \phi_{\mathcal{H}_i}(q_i^0, \sigma_i^0, \tilde{I}_i(0,t))$, for $t \geq 0$, of (6). Note that $\tilde{I}_i(0,t)$ represents an input signal for the above solution. This finally lets us distinguish cooperative from uncooperative agents as follows:

Definition 2. Exactly cooperative and uncooperative agents. *$\mathcal{A}_i$ is exactly cooperative under $\mathcal{P}_i$ if the ith history $\tilde{q}_i(0,t)$ solves (6) for some initial state values, q_i^0 and σ_i^0, and some input set history $\tilde{I}_i(0,t)$. Otherwise, $\mathcal{A}_i$ is uncooperative.*

In the above sense, an agent $\mathcal{A}_i$ that is not following $\mathcal{P}_i$ is considered an *intruder* of the robot society.

4. Protocols for the design of local monitors

This section focuses on how an agent $\mathcal{A}_h$ is enabled to understand whether another $\mathcal{A}_i$ that is within its visibility scope, i.e., $q_i(t) \in V_h$ for some interval of t, is following the associated cooperation model $\mathcal{P}_i$. Referring to the motorway example, this amounts to understanding how to enable a vehicle $\mathcal{A}_i$ to determine whether one of the vehicles that is within visibility range is following the driving rules. For the sake of implementability, to achieve its goal, suppose that $\mathcal{A}_h$ measures and collects the history of the actual behavior of $\mathcal{A}_i$ over consecutive time slots, at the end of which a local *monitor* process $\mathcal{M}_h$ is executed to decide whether $\mathcal{A}_i$ is cooperative or not. The challenges faced by each $\mathcal{M}_h$ and the proposed solution are described below. The key symbols used are summarized in Table 8 in Appendix E.

4.1. Brief introduction of the local monitor

The implementation of a local monitor $\mathcal{M}_h$ must cope with having partial knowledge of the input set I_i of the agent being evaluated. Indeed, while one can assume that $\mathcal{A}_i$ has full access to its own neighborhood $\mathcal{N}_i$ through its sensors, the same hypothesis cannot be made regarding the monitor's sensors since, in general, one has $\mathcal{N}_i \subseteq V_h$. In the motorway example, some of the vehicles that are nearby $\mathcal{A}_i$ may be covered by other vehicles from the position of $\mathcal{A}_h$ or may be out of reach of its sensors. The information available for $\mathcal{M}_h$ is, in fact, the ith *partial input set* $I_i^h = I_i \cap V_h$ only.

Technically speaking, whenever the whole input set I_i is available during any time slot $[0, \Delta]$, a monitor $\mathcal{M}_h$ can compute the *unique* cooperative behavior of $\mathcal{A}_i$ via a forward simulation of model $\mathcal{H}_i$, initialized with the previously reconstructed state, and compare such a behavior with the history of the actually measured one, $\tilde{q}_i^{\text{meas}}(0,\Delta)$; if the two signals match exactly, $\mathcal{M}_h$ is able to conclude that $\mathcal{M}_h$ is executing a trajectory conforming to $\mathcal{P}_i$ and can classify the agent as certainly cooperative, or otherwise as uncooperative. This direct solution only applies if $I_i^h = I_i$, yet its strategy is basic to obtain a generalized solution that is valid also in the more challenging general case. Towards this goal, the following definition can be given, introducing a strict and relaxed notion of behavior explanation:

Definition 3. Behavior explanations and ϵ-similar behavior explanations. *A set $I^* \in \text{Pow}(\mathcal{Q}^{n_i})$ is said an explanation of a behavior $q_i^{\text{meas}}(t)$ for $t \in [0, \Delta]$ if the history $\tilde{I}^*(0,\Delta)$ solves (6), for some $\overline{\sigma} \in \Sigma_i$,* i.e., *$\tilde{q}_i^{\text{meas}}(0,\Delta) = \phi_{\mathcal{H}_i}(q_i^{\text{meas}}(0), \overline{\sigma}, \tilde{I}^*(0,\Delta))$. Given a tolerance $\epsilon > 0$, an explanation is ϵ-similar to $\tilde{q}_i^{\text{meas}}(0,\Delta)$ if the signals $\phi_{\mathcal{H}_i}(q_i^{\text{meas}}(0), \overline{\sigma}, \tilde{I}^*(0,\Delta))$ and $\tilde{q}_i^{\text{meas}}(0,\Delta)$ are ϵ-similar. Explanations with initial conditions $\overline{q} \in \mathcal{Q}$*

and $\overline{\Sigma} \subseteq \Sigma_i$ *are explanations where* $q_i^{\text{meas}}(0) = \overline{q}$ *and for some* $\overline{\sigma} \in \overline{\Sigma}$.

Intuitively speaking, in the case of a partial view, a monitor $\mathcal{M}_h$ can try to find the set $\widehat{I}_i^h$ of all possible input sets that are supersets of I_i^h and *explain* the ith measured behavior. Precisely, one has to find the set $\widehat{I}_i^h = \{\widehat{I}_{i,1}^h, \widehat{I}_{i,2}^h, \cdots\}$ such that each $\widehat{I}_{i,\ell}^h \in \text{Pow}(\mathcal{Q}^{n_i})$ satisfies $I_i^h \subseteq \widehat{I}_{i,\ell}^h$ and explains $\tilde{q}_i^{\text{meas}}(0, \Delta)$. Note that $\widehat{I}_i^h$ should be exhaustive, i.e., include all possible explanations, and its elements be the smallest ones, that is, each $\widehat{I}_{i,\ell}^h$ be of minimal cardinality and measure. In the motorway example, a monitor $\mathcal{M}_h$ onboard the hth vehicle must try to reconstruct the smallest yet exhaustive set $\widehat{I}_i^h$ of all possible input sets $\widehat{I}_{i,\ell}^h$, comprising the configurations of $\mathcal{A}_i$-neighbors that are visible from $\mathcal{M}_h$ or that lay in hidden areas out of V_h but that are required to explain the measured behavior of $\mathcal{A}_i$. As it will be clarified later, each $\widehat{I}_{i,\ell}^h$ represents an estimated *occupancy* map of the ith neighborhood, i.e., a map describing where all $\mathcal{A}_i$-neighbors possibly lie within $\mathcal{N}_i$ and where they must not; this map includes single-point continuous states describing measured locations of neighbors as well as continuous sets describing regions where the presence/absence of one or more neighbors is inferred from the behavior of $\mathcal{A}_i$, yet their correctness needs to be confirmed.

In this regard, by anticipating, for the sake of clarity, the consensus protocol of the next section, more monitors can attempt to classify $\mathcal{A}_i$ cooperatively by sharing locally estimated occupancy maps. A region in one of these maps is said to be *confirmed* if there exists at least one local monitor that can directly verify the presence/absence of an $\mathcal{A}_i$-neighbor with its own sensors; accordingly, an occupancy map $\widehat{I}_i^h$ is *fully confirmed* if all its regions are confirmed. As a special case, an occupancy map $\widehat{I}_i^h$ is fully confirmed by a single monitor $\mathcal{M}_h$, if $\mathcal{M}_h$ has full visibility on $\mathcal{N}_i$, since, if so, $I_i^h = I_i$. On the whole, as long as the set $\widehat{I}_i^h$ is not empty, $\mathcal{M}_h$ is able to conclude that $\mathcal{A}_i$ is *possibly cooperative*. Whenever there is one confirmed estimate $\widehat{I}_{i,\ell}^h$, the local monitor $\mathcal{M}_h$ can conclude that $\mathcal{A}_i$ is *certainly cooperative*. If no behavior explanation is found, i.e., $\widehat{I}_i^h = \emptyset$, $\mathcal{A}_i$ is classified as *uncooperative*. Finally, it should be noted that, if $\mathcal{A}_i$ is conforming to $\mathcal{P}_i$, the real ith input set I_i, which is *unknown* to monitor $\mathcal{M}_h$, is included by definition in each estimated occupancy map $\widehat{I}_i^h$, i.e., there always exists $\widehat{I}_{i,\ell}^h \in \widehat{I}_i^h$ satisfying $I_i \subseteq \widehat{I}_{i,\ell}^h$. In addition, if $\mathcal{M}_h$ is able to perform observations during consecutive time slots, the reasoning described above can capitalize on the last estimates of the continuous state and the set of discrete states, say $\overline{q} \in \mathcal{Q}$ and $\overline{\sigma} \in \Sigma_i$, respectively. Then, the search for explanations can be narrowed to those that start with initial conditions $\overline{q}$ and any discrete state in $\overline{\Sigma}$.

A final aspect to take into account is the uncertainty caused by the measurement noise of the monitor's sensors, whereby the measured continuous state, including $\mathcal{A}_i$'s behavior, may be slightly different from the actual states. An indirect yet effective way to deal with it is to introduce a *mismatch tolerance* $\epsilon > 0$ and loosen the matching test between the measured and any predicted behavior with a check on whether the difference between their signals is, in norm, within a tube of radius ϵ; this is accounted for by searching for ϵ-similar explanations (Bressan and Piccoli, 2007). Another way of looking at it is that any accepted behavior explanation is such that the history of the corresponding predicted behavior lies within a tube built around the solution of (6) with a radius of ϵ. In a sentence, the classification strategy of a monitor $\mathcal{M}_h$ to dealing with visibility uncertainty assumes that $\mathcal{A}_i$ is cooperative *until proven otherwise*, and thus involves trying to find all possible ϵ-similar explanations to the measured behavior.

4.2. Technical explanation of the local monitor

The ability of a monitor $\mathcal{M}_h$ to classify the behavior of $\mathcal{A}_i$, in a certain or uncertain way, depends on several factors. The first factor is the full or partial visibility of $\mathcal{M}_h$ over the region of the ith neighborhood. If $\mathcal{A}_h$ has full visibility on $\mathcal{N}_i$, $\mathcal{M}_h$ can directly assess, for each topology $\eta_{i,j}$, the presence/absence of any $\mathcal{A}_i$-neighbor, $\mathcal{A}_m$, and measure their continuous state, q_m; if this is true for all topologies, the ith input set seen by $\mathcal{M}_h$ equals the whole ith input set ($I_i^h = I_i$). Conversely, when $\mathcal{A}_h$ has partial visibility on any topology $\eta_{i,j}$, it can both happen that one or more $\mathcal{A}_i$-neighbors are hidden by other agents/obstacles or out of reach of the hth sensors, or that no neighbor at all is, in fact, present in the non-visible part, $\eta_{i,j}(q_i) \setminus V_h$, of the ($i$, j)th topology. In the latter case, part of the ith input set may not be known to $\mathcal{M}_h$ and I_i^h be a strict subset I_i; in general, then it will hold $I_i^h \subseteq I_i$. The information connected to this first reasoning is represented in the ith *topology check* map v_i that will be defined later. In addition, based on the available information, $\mathcal{M}_h$ can assess the ith encoder map s_i restricted to the visible part of the ith neighborhood, $\mathcal{N}_i \cap V_h$; in this way, it will obtain a lower estimate of it. This second information is stored in a *known encoder* map $\widehat{s}_i$, also defined later, whose jth component $\widehat{s}_{i,j}$ is such that $\widehat{s}_{i,j}(q_i, I_i^h) \leq s_{i,j}(q_i, I_i)$ because $I_i^h \subseteq I_i$.

A second fundamental ability for the monitor is to predict all possible cooperative behaviors of $\mathcal{A}_i$ that are compliant with the *partial* input set I_i^h by using the values of the known encoder map $\widehat{s}_i$. This feature requires first evaluating the superset of all events, $e^{i,j}$, possibly affecting the behavior of $\mathcal{A}_i$ and being recognized by the ith decoder map e_i. An *event estimator* map $\widehat{e}_i$ that always satisfies the condition

$e_i(s_i) \subseteq \widehat{e}_i(\widehat{s}_i, v_i)$ is introduced for this purpose. The explicit formula of $\widehat{e}_i$ is described later in Th. 1 and derived in Appendix B. The logic behind its derivation is to try to factorize the ith encoder map s_i in terms of the values of the known encoder $\widehat{s}_i$, and the topology visibility check v_i, thus obtaining a *visibility factorization* of the encoder map components $s_{i,j}$. Intuitively, this is based on the reasoning related to the two following cases:

- *An event occurring upon activation of the single jth component of* s_i. If monitor $\mathcal{M}_h$ has full visibility on $\eta_{i,j}$ (and hence $v_{i,j} = 1$), the encoder and known encoder maps have the same definitions, by which their values are equal ($\widehat{s}_{i,j} = s_{i,j}$) and, hence, the occurrence of $e^{i,j}$ coincides with the activation of $\widehat{s}_{i,j}$; otherwise, if $\mathcal{M}_h$ has partial view ($v_{i,j} = 0$), two cases can happen: if the known encoder $\widehat{s}_{i,j}$ is already active, also $s_{i,j}$ is active (since $I_i^h \subseteq I_i$) and event $e^{i,j}$ has necessarily occurred; if not, nothing can be said on the value of $s_{i,j}$ as there may exist one or more $\mathcal{A}_i$-neighbors not visible from $\mathcal{M}_h$. In the latter, $s_{i,j}$ is assumed conservatively as active and the event $e^{i,j}$ as possibly occurred. Writing the truth table of this logic explicitly yields the visibility factorization: $s_{i,j} = \widehat{s}_{i,j} v_{i,j} \oplus \neg v_{i,k}$;
- *An event occurring upon non-activation of the single jth component of* s_i. In case of full visibility ($v_{i,j} = 1$), it holds $\neg s_{i,j} = \neg \widehat{s}_{i,j}$. In case of partial visibility ($v_{i,j} = 0$), the presence of one or more $\mathcal{A}_i$-neighbors is sufficient to exclude the occurrence of event $e^{i,j}$; in contrast, the absence of $\mathcal{A}_i$-neighbors detected by $\mathcal{M}_h$, does not let to exclude that $s_{i,j}$ is active. The factorization here simplifies to: $s_{i,j} = \neg \widehat{s}_{i,j}$.

The general case is wholly derived in Appendix and is obtained as a recursive iteration of the reasoning above. The result is described in the following:

Theorem 1. *Consider a detector map e_i with associated encoder map s_i as in (1). The smallest, i.e., minimum cardinality, event estimator compatible with a known encoder map $\widehat{s}_i$ and a topology check map v_i is*

$$\begin{aligned} \widehat{e}_i : \mathbb{B}^{\kappa_i + h_i} \times \mathbb{B}^{\kappa_i + h_i} &\to Pow(E_i) \\ (\widehat{s}_i, v_i) &\mapsto \left\{ e^{i,j} \in E_i \middle| \left(\otimes_{\ell \in \gamma_{i,j}} * \neg \widehat{s}_{i,\ell} \right) \otimes \right. \\ &\left. \otimes \left(\otimes_{\ell \in \gamma_{i,j}} (\widehat{s}_{i,\ell} v_{i,\ell} \oplus \neg v_{i,\ell}) \right) \right\}. \end{aligned} \tag{7}$$

In a nutshell, the event estimator map $\widehat{e}_i$ *embeds* the effect of visibility uncertainty in the event detection process, so that a *conservative* calculation of all possible outcomes of the detector map e_i is obtained. It is worth noting that, to get the best performance in the event estimation, the set returned by $\widehat{e}_i$ should have the smallest cardinality, which in turn requires that the known encoder map $\widehat{s}_i$ is the largest lower estimate (so the closest from below) of s_i.

Still in relation to the second skill, $\mathcal{M}_h$ must be able to translate, a priori, the estimated events into expected cooperative behaviors $\tilde{q}_i^{\text{pre}}(0, \Delta)$. To this end, $\mathcal{M}_h$ can first compute all possible successive discrete states, considering all transitions of the automaton map δ_i that may be triggered; this information is stored in a *nondeterministic automaton* map Δ_i that is automatically synthesized from δ_i. These discrete states are then used to compute the set $\tilde{q}_i^{\text{pre}}(0, \Delta)$ of all predicted behaviors through the controlled dynamics map f_i^*; during this task, $\mathcal{M}_h$ may need to represent possible hidden $\mathcal{A}_i$-neighbors that lie in the region $\eta_{i,j}(q_i) \setminus V_h$; this is done by introducing an *extended input set* map $\widehat{I}_i^h = \{I_{i,\ell}^h\}_\ell$ that extends the known input set I_i^h with a set of discretized points in the hidden parts of each topology; based on each extended input $I_{i,\ell}^h$, the corresponding behavior $\tilde{q}_{i,\ell}(0, \Delta)$ is computed. Finally, as soon as the measured behavior $\tilde{q}_i^{\text{meas}}(0, \Delta)$ is available, $\mathcal{M}_h$ can, *a posteriori*, try to match any predicted behavior in $\tilde{q}_i^{\text{pre}}(0, \Delta)$ with the real one. It is crucial, here, to introduce a matching tolerance that takes into account the inaccuracy of the hth sensors, discretization in representing hidden agents in the ith neighborhood, and "small" yet acceptable mismatch between the ith nominal model and the real model of $\mathcal{A}_i$. As mentioned above, a practical way to do so is to use a *mismatch tolerance* parameter ϵ that is incorporated into the definition of a *similarity check* map $\bowtie_\epsilon$, which is used when comparing any pair of actual and expected behavior.

A third final skill is the ability of $\mathcal{M}_h$ to *invert* the cooperative model $\mathcal{H}_i$ in order to retrieve, a posteriori, information about the actual input set $I_i \subseteq \widehat{I}_i^h$, the actual events, and the occupancy of each (i, j)th topology. An explicit inversion is computationally impractical given the hybrid and nonlinear nature of $\mathcal{H}_i$. Therefore, $\mathcal{M}_h$ needs to build a bidirectional *link* between any predicted behavior $\tilde{q}_{i,\ell}(0, \Delta)$ and the corresponding discrete state and *extended known input* $\widehat{I}_{i,\ell}^h$. This information is stored in a *forward link* table $\mathcal{L}_i$, which will be defined later and which allows $\mathcal{M}_h$ to map any matched behavior back to the triggering events. By doing so, $\mathcal{M}_h$ can obtain an improved a-posteriori evaluation of the encoder map s_i, through a *refined encoder map* s_i^h, which uses the *inverse* map Δ_i^{-1} of the nondeterministic automaton map Δ_i to extract the set of events that may have caused the transition from two consecutive discrete states; this, in turn, leads to estimating the occupancy of the first κ_i topologies; this is described in an *occupancy estimator* map η_i^h. Also, this leads to enlarging the part of the ith visibility region V_i known to $\mathcal{M}_h$, which is obtained by a *refined visibility* map V_i^h also defined later.

4.3. Formal derivation of the local monitor

Denoting by $I_i^h = I_i \cap V_h$ the subset of the ith input set I_i that is known to $\mathcal{M}_h$, the construction of the local monitor

requires the introduction of the following few new objects, most of which derive directly from $\mathcal{P}_i$:

- A *topology check* map $v_i : \mathcal{Q} \times \mathrm{Pow}(\mathcal{Q}) \to \mathbb{B}^{\kappa_i}$, whose jth component is active if the jth topology of $\mathcal{A}_i$ is entirely visible from $\mathcal{A}_h$, i.e.,

$$v_{i,j} : \mathcal{Q} \times \mathrm{Pow}(\mathcal{Q}) \to \mathbb{B}$$
$$(q_i, V_h) \mapsto \begin{cases} 1 & \text{if } \eta_{i,j}(q_i) \subseteq V_h \\ 0 & \text{otherwise} \end{cases};$$

- A *known encoder* map $\widehat{s}_i$, whose jth entry is a lower approximation of $s_{i,j}$ based on I_i^h, with $\widehat{n}_i = \mathrm{card}(I_i^h)$:

$$\widehat{s}_{i,j} : \mathcal{Q} \times \widehat{\mathcal{Q}^{n_i}} \to \mathbb{B}$$
$$\left(q_i, I_i^h\right) \mapsto \oplus_{q_k \in I_i^h} \mathbf{1}_{\eta_{i,j}(q_i)}(q_k);$$

- A *nondeterministic automaton* map Δ_i describing, for each estimated pair of discrete state set $\widehat{\sigma}_i$ and events $\widehat{e}_i$, the set $\sigma_i^{'}$ of possible next discrete states; this is obtained as:

$$\Delta_i : \mathrm{Pow}(\Sigma_i) \times \mathrm{Pow}(E_i) \to \mathrm{Pow}(\Sigma_i)$$
$$(\widehat{\sigma}_i, \widehat{e}_i) \mapsto \{\overline{\sigma} \in \Sigma_i | \exists \sigma \in \Sigma_i, \sigma \subseteq \widehat{\sigma}_i | \delta_i(\sigma, \widehat{e}_i) = \overline{\sigma}\};$$

Correspondingly, its *inverse* map, Δ_i^{-1}, describing, for each pair (σ, σ') of discrete states, the set of events $e^{i,j}$ that may cause the transition from σ to σ' is also introduced; this is obtained as:

$$\Delta_i^{-1} : \mathrm{Pow}(\Sigma_i) \times \mathrm{Pow}(\Sigma_i) \to \mathrm{Pow}(E_i)$$
$$(\sigma, \sigma') \mapsto \{e \in E_i | \exists \overline{\sigma}, \overline{\sigma}' \in \Sigma_i, \overline{\sigma} \subseteq \sigma, \overline{\sigma}' \subseteq \sigma', \overline{\sigma}' \subseteq \Delta_i(\sigma, e)\};$$

Note that $\sigma_i \in \Sigma_i$ is the (unique) discrete state executed by $\mathcal{A}_i$ at a given time, while $\widehat{\sigma}_i \in \mathrm{Pow}(\Sigma_i)$ is used to build the conservative estimate set $\sigma_i^{'}$ including all possible discrete states complying with the knowledge of the local monitor;

- An *extended input set* map $\widehat{I}_i^h$ generating, from the ith known input set I_i^h, a set of *extended known inputs*, $\{\widehat{I}_{i,\ell}^h\}_\ell$, each including I_i^h and other points in the unknown neighborhood $\mathcal{N}_i \setminus V_h$ obtained via a discretization scheme.

Lastly, a few further operators need to be introduced:

- A *forward link* table $\mathcal{L}_i$ that is a list of triples, each of which includes a behavior defined over the interval [0, Δ], a discrete state, and the related input set; this is defined over the domain $D_i = \mathcal{F}_{[0,\Delta]}(\mathcal{Q}) \times \Sigma_i \times \mathrm{Pow}(\mathcal{Q})$;

- A *similarity check* map $\bowtie_\epsilon$ that, given a forward link table $\mathcal{L}_i$ and a measured behavior $\tilde{q}(0, \Delta)$, returns the subset of the tuples with behaviors that are similar to the measured one; this is obtained as

$$\bowtie_\epsilon : \mathrm{Pow}(D_i) \times \mathcal{F}_{[0,\Delta]}(\mathcal{Q}) \to \mathrm{Pow}(D_i)$$
$$(\mathcal{L}_i, \tilde{q}(0, \Delta)) \mapsto \{L^* = \left(\tilde{q}^*(0, \Delta), \sigma^*, I^*\right) \subseteq \mathcal{L}_i |$$
$$\tilde{q}^*(0, \Delta) \text{ is } \epsilon - \text{similar to } \tilde{q}(0, \Delta)\};$$

- A *refined encoder* map s_i^h returning the largest encoder map estimate complying with a known encoder map s and a pair (σ, σ') of successive discrete state sets; this is obtained as:

$$s_i^h : \mathbb{B}^{\kappa_i + h_i} \times \mathrm{Pow}(E_i) \times \mathrm{Pow}(E_i) \to \mathrm{Pow}\left(\mathbb{B}^{\kappa_i + h_i}\right)$$
$$(\widehat{s}_i, \sigma, \sigma') \mapsto \{s \in \mathbb{B}^{\kappa_i + h_i} | s \geq \widehat{s}_i,$$
$$e_i(s) \subseteq \Delta_i^{-1}(\sigma, \sigma')\}; \tag{8}$$

- An *occupancy estimator* map $\eta_i^h = (\eta_{i,1}^h, \ldots, \eta_{i,\kappa_i}^h)$, where each map $\eta_{i,j}^h$ is a function that, given the jth components, s^j and $\overline{s}^j$, of known and refined encoder maps, respectively, and the known input set I_i^h, returns points and continuous sets estimating the occupancy of the (i, j)th topology; this is obtained as:

$$\eta_{i,j}^h : \mathbb{B} \times \mathbb{B} \times \mathrm{Pow}\left(\widehat{\mathcal{Q}^{n_i}}\right) \to \mathrm{Pow}(\mathcal{Q})$$
$$\left(s^j, \overline{s}^j, I_i^h\right) \mapsto \begin{cases} \left\{I_{i,j}^h, W_{i,j}^h\right\} & \text{if } s^j, \\ \left\{I_{i,j}^h, W_{i,j}^h, \overline{W}_{i,j}^h, \overline{s}^j\right\} & \text{if } \neg s^j, \end{cases} \tag{9}$$

where $I_{i,j}^h = I_i^h \cap \eta_{i,j}(q_i)$, $W_{i,j}^h = \eta_{i,j}(q_i) \cap V_h$, and $\overline{W}_{i,j}^h = \eta_{i,j}(q_i) \setminus V_h$ are the portions of the (i, j)th topology that are visible and non-visible to $\mathcal{M}_h$, respectively; note that $W_{i,j}^h$ is a region where $\mathcal{M}_h$ has direct visibility and thus can *confirm or exclude* any assumptions about the presence or absence of additional neighbors of $\mathcal{A}_i$, information that will be used in the consensus algorithm described in the next section;

- A *refined visibility* map $V_i^h = (V_{i,1}^h, \ldots, V_{h,\kappa_i}^h)$, where each map $V_{i,j}^h$ describes the portion of the (i,j)th topology which is eventually known to $\mathcal{M}_h$ either through direct measures or through inference from the behavior of $\mathcal{A}_i$; given the jth components, s^j and $\overline{s}^j$, of known and refined encoder maps, respectively, these are obtained as:

$$V_{i,j}^h : \mathbb{B} \times \mathbb{B} \to \mathrm{Pow}(\mathcal{Q})$$
$$(s^j, \overline{s}^j) \mapsto \begin{cases} W_{i,j} & \text{if } s^j, \\ W_{i,j} \cup \eta_{i,j}(q_i) & \text{if } \neg s^j; \end{cases} \tag{10}$$

- A *classifier* map $\mathcal{C}_i$ that, given a forward link table $\mathcal{L}_i$, a measured behavior $\tilde{q}$, and the list $\{V_{i,j}\}$ of visible portions of each (i,j)th topology, returns the corresponding classification decision; this is given by:

$$\mathcal{C}_i : D_i \times \mathcal{F}_{[0,\Delta]}(\mathcal{Q}) \times \mathrm{Pow}(\mathcal{Q}) \to \{S_1, S_2, S_3\}$$
$$\left(\mathcal{L}_i, \tilde{q}, \{W_{i,j}\}_j\right) \mapsto \begin{cases} S_1 & \text{if } \overline{L} \neq \emptyset, \quad \exists\, \overline{I}_{i,\ell} \subseteq W_i, \\ S_2 & \text{if } \overline{L} \neq \emptyset, \quad \forall\, \overline{I}_{i,\ell} \not\subseteq W_i, \\ S_3 & \text{if } \overline{L} = \emptyset, \end{cases} \tag{11}$$

with S_1 = 'certainly cooperative', S_2 = 'possibly cooperative', and S_3 = 'uncooperative', and with $\overline{L} = \mathcal{L}_i \bowtie_\epsilon \tilde{q} = \{\tilde{q}_{i,\ell}, \sigma_{i,\ell}, \overline{I}_{i,\ell}\}_\ell$, and $W_i = \cup_j W_{i,j}$.

Having introduced all required components, one can now describe the operation of the hth monitor. For practical implementability, the monitoring of $\mathcal{A}_i$ is organized as a process that is run every Δ seconds. Let $k \in \mathbb{N}_0^+$ be the iteration *step* of the current execution, and let $t \in [0, \Delta]$ be the elapsed time since the start of that iteration. Suppose that $\mathcal{M}_h$ measures, at every t, the actual state of $\mathcal{A}_i$, say $q_i^{\mathrm{meas}}(t)$, so that, at the end of every step, the history of the *measured behavior*, $\tilde{q}_i^{\mathrm{meas}}(0,\Delta)$, is available. Moreover, each kth step comprises a prediction and an update/matching phase. During the *prediction* phase, $\mathcal{M}_h$ computes the set of all possible behaviors conforming to $\mathcal{P}_i$ based on the available partial input set I_i^h; this is given by a set

$$\tilde{q}_i^{\mathrm{pre}}(0,\Delta) := \left\{\tilde{q}_{i,1}^{\mathrm{pre}}(0,\Delta), \tilde{q}_{i,2}^{\mathrm{pre}}(0,\Delta), \cdots\right\}, \tag{12}$$

where each $\tilde{q}_{i,\ell}^{\mathrm{pre}}(0,\Delta) \in \mathcal{Q}$ for all times. This is obtained by evaluating the known encoder map $\widehat{s}_i$, which then allows estimating all possible events through the event estimator $\widehat{e}_i$; then, the history set $\tilde{q}_i^{\mathrm{pre}}(0,\Delta)$ of all predicted behaviors are computed via a forward simulation of the controlled dynamics set map F_i. The forward link table $\mathcal{L}_i$ is also updated during this stage. During the *update/matching* phase, $\mathcal{M}_h$ compares the *unique* measured behavior $\tilde{q}_i^{\mathrm{meas}}(0,\Delta)$ with all predicted behaviors and drops out the ones that are not ϵ-similar; this is obtained through the operation $\mathcal{L}_i[k] \bowtie_\epsilon \tilde{q}_i^{\mathrm{meas}}(0,\Delta)$ that returns the only links with matching behaviors. Restricting the result to the set Σ_i also allows updating the discrete state set.

Finally, $\mathcal{M}_h$ can determine an estimate of the *occupancy* of each (i,j)th topology, for $j \leq \kappa_i$, possibly including data on the presence/absence of $\mathcal{A}_i$-neighbor in each unknown part $\eta_{i,j} \setminus V_h$; it can also extend its own visibility map V_h based on information inferred via the behavior matching process. The strategy behind these two operations is as follows. First, each (i,j)th component of the encoder map s_i, for $j \leq \kappa_i$, is expanded as the binary sum of the known encoder map $\widehat{s}_{i,j}$ and the remaining unknown term, say $p_{i,j}$; in formula, this is

$$s_{i,j}(q_i, I_i) = \widehat{s}_{i,j}\left(q_i, I_i^h\right) \oplus p_{i,j}(q_i, I_i), \tag{13}$$

where $p_{i,j}(q_i, I_i) = \oplus_{q_\ell \in I_i \setminus V_h} \mathbf{1}_{\eta_{i,j}(q_i)}(q_\ell)$ conveys information on $I_i \setminus V_h$, whose value is to be estimated. To achieve so, assume that $\widehat{s}_{i,j}$ is a priori predicted, while the set $s_{i,j}^h$ is computed a posteriori during the update phase. Now recall that, by construction, $\widehat{s}_i \leq s_i$ and, for every $s \in s_i^h$, $s \geq \widehat{s}_i$ (see the definition of s_i^h in (8)), and also note that, if $\mathcal{A}_i$ is cooperative, $s \geq s_i$; this implies that $\widehat{s}_i \leq s_i \leq s$, with $s \in s_i^h$, meaning that the available a priori and a posteriori estimates bound the real yet unknown value of the encoder map s_i. Then, using (13), one can write $\widehat{s}_i \leq \widehat{s}_i \oplus p_i \leq s$, with $s \in s_i^h$ and, directly inspecting all cases, the following is obtained: for each $s = (s^1, \ldots, s^{\kappa_i + h_i}) \in s_i^h$, if $s^j = \widehat{s}_{i,j} = 0$, then it must be $p_{i,j} = 0$ and the absence of any $\mathcal{A}_i$-neighbor in the non-visible part of the (i,j)th topology is inferred; if $s^j = 1$ but $\widehat{s}_{i,j} = 0$, then it must be $p_{i,j} = 1$ and the presence of at least $\mathcal{A}_i$-neighbor in the non-visible part of the (i,j)th topology is inferred; the case where $s^j = \widehat{s}_{i,j} = 1$ is not informative, as $p_{i,j}$ can be either 0 or 1, and adds no information, since $\mathcal{M}_h$ already knows about the existence of an $\mathcal{A}_i$-neighbor in the (i,j)th topology justifying the behavior of $\mathcal{A}_i$; finally the case $s^j = 0$ and $\widehat{s}_{i,j} = 1$ is not possible by construction.

From the two informative cases described above, it is possible to obtain the occupancy estimates $\eta_{i,j}^h$ and the components of the refined visibility map for each (i,j)th topology. Let $W_{i,j} = \eta_{i,j}(q_i) \cap V_h$ be the portion of the (i,j)th topology that is visible by $\mathcal{M}_h$. To begin with, the occupancy estimate must contain the list $I_{i,j}^h = I_i^h \cap \eta_{i,j}(q_i)$ made of the continuous states of all $\mathcal{A}_i$-neighbors that are visible by $\mathcal{M}_h$. The case with $I_{i,j}^h$ not empty (and hence $\widehat{s}_{i,j} = 1$), does not allow, as said above, inferring information on the non-visible part of the (i,j)th topology; accordingly, the jth component of the refined visibility map is set to be the visible portion of the (i,j)th topology, i.e., $V_{i,j}^h = W_{i,j}$. Instead, with $I_{i,j}^h$ being empty (and hence $\widehat{s}_{i,j} = 0$), two sub-cases occur: first, if no further neighbor is inferred a posteriori ($s_{i,j}^h = 0$), the occupancy estimate must describe the fact the non-visible part of the (i,j)th topology needs to empty if $\mathcal{A}_i$ is cooperative; second, if at least an $\mathcal{A}_i$-neighbor is inferred a posteriori ($s_{i,j}^h = 1$), the occupancy estimate must describe the fact the non-visible part of the (i,j)th topology needs to be occupied if $\mathcal{A}_i$ is cooperative. In both cases, the occupancy estimate must contain the region $\eta_{i,j}(q_i) \setminus V_h$, *annotated* by the binary number $\overline{s}^j$. Also, the monitor's visibility is extended by setting the jth component of the refined visibility component to $V_{i,j}^h = \eta_{i,j}(q_i)$. All this reasoning is encoded into the ith occupancy estimator map η_i^h and refined visibility map V_i^h.

Finally, the logic by which $\mathcal{M}_h$ can determine the cooperativeness of $\mathcal{A}_i$, intuitively described in the previous section, is made formal as follows. First, $\mathcal{M}_h$ can extract, from the forward link table $\mathcal{L}_i$, the elements that contain a behavior that is ϵ-similar to the measured behavior $\tilde{q}$, which

are obtained as $\overline{L} = \mathcal{L}_i \bowtie_\epsilon \tilde{q}$. Then, if $\overline{L} = \phi$, no explanation exists and the ith agent is considered uncooperative; otherwise, given $\overline{L} = \{\tilde{q}^{\text{pre}}_{i,\ell}, \sigma^{\text{pre}}_{i,\ell}, \overline{I}_{i,\ell}\}_\ell$, $\mathcal{M}_h$ needs to check whether or not any estimated input set $\overline{I}_{i,\ell}$ can be directly confirmed by its own sensors; in the first case, $\mathcal{A}_i$ is certainly cooperative, while in the second case, it is assumed to be possibly-cooperative. These two last cases are distinguished by verified if $\overline{I}_{i,\ell} \subseteq \cup_j W_{i,j}$. This reasoning is encoded into the *classifier* map $\mathcal{C}_i$ described above.

4.4. Algorithmic description of the local monitor

From a procedural viewpoint, $\mathcal{M}_h$ is organized as follows:

- *Initialization Phase.* The iteration step is set to $k = 0$. $\mathcal{M}_h$ measures the instantaneous value of the continuous state of $\mathcal{A}_i$ and uses it as the first update estimate $q^{\text{post}}_i(0) := q^{\text{meas}}_i(0)$; moreover, having no prior knowledge of the ith behavior, the discrete state set is set conservatively to any possible value, i.e., $\sigma^{\text{post}}_i[0] = \Sigma_i$.
- *Prediction Phase.* At the kth step, $\mathcal{M}_h$ evaluates the topology visibility check v_i and the known encoder map $\widehat{s}_i$ using the latest value of the continuous state $q^{\text{meas}}_i(0)$ and the partial input set I^h_i; this is obtained as follows:

$$v_i[k] = v_i\big(q^{\text{meas}}_i[k], V_h[k]\big),$$
$$\widehat{s}_i[k] = \widehat{s}_i\big(q^{\text{meas}}_i[k], I^h_i[k]\big);$$

then, the set of predicted events is computed by using the known encoder map as $e^{\text{pre}}_i[k] = \widehat{e}_i(\widehat{s}_i[k], v_i[k])$, which allows computing the set of predicted discrete states

$$\sigma^{\text{pre}}_i[k] := \Delta_i\big(\sigma^{\text{post}}_i[k-1], e^{\text{pre}}_i[k]\big); \tag{14}$$

and, thereafter, the extended known input set $\widehat{I}^h_i = \{I_{i,\ell}\}_\ell$ and the set of predicted cooperative behaviors (12); these behaviors are obtained as the solutions, for all $(\sigma_\ell, I_{i,\ell}) \in \sigma^{\text{pre}}_i[k] \times \widehat{I}^h_i$, of the following ODE with initial condition $q_{i,\ell}(0) = q^{\text{post}}_i[k]$:

$$\dot{q}_{i,\ell}(t) = f^*_i\big(q_{i,\ell}(t), \sigma_\ell, I_{i,\ell}\big); \tag{15}$$

Finally, the forward link table $\mathcal{L}_i$ is updated as the set of all triples of predicted behavior $\tilde{q}_{i,l}(0,\Delta)$, and related discrete state σ_ℓ and extended known input set $I_{i,\ell}$, i.e.,

$$\mathcal{L}_i[k] := \big\{\big(\tilde{q}_{i,l}(0,\Delta), \sigma_\ell, I_{i,\ell}\big)\big\}_\ell \ .$$

- *Update/Matching Phase.* During this phase, $\mathcal{M}_h$ tries to match the measured behavior with any predicted one and, if successful, updates the values of the ith continuous state $q^{\text{post}}_i[k]$, the discrete state set $\sigma^{\text{post}}_i[k]$, and known encoder map $s^{\text{post}}_i[k]$; this is obtained through the operations:

$$q^{\text{post}}_i(t_{k+1}) = q^{\text{meas}}_i(t_{k+1}),$$
$$\sigma^{\text{post}}_i[k] = \text{Proj}_{\Sigma_i}\big(\mathcal{L}_i[k] \bowtie_\epsilon \tilde{q}^{\text{meas}}_i(0,\Delta)\big),$$
$$s^{\text{post}}_i[k] = s^h_i\big(s^{\text{pre}}_i[k], \sigma^{\text{post}}_i[k-1], \sigma^{\text{post}}_i[k]\big);$$

In addition, $\mathcal{M}_h$ computes the occupancy estimator map η^h_i and enlarges, if possible, the known visibility map V^h_i as follows:

$$\eta^h_i[k] = \eta^h_i\big(s^{\text{pre}}_i[k], s^{\text{post}}_i[k], I^h_i[k]\big),$$
$$V^h_i[k] = V^h_i\big(s^{\text{pre}}_i[k], s^{\text{post}}_i[k]\big);$$

Finally, the behavior of $\mathcal{A}_i$ is classified as follows:

$$C^h_i[k] = \mathcal{C}_i\big(\mathcal{L}_i[k], \tilde{q}^{\text{meas}}_i(0,\Delta), \{V_{i,j}\}\big).$$

- *Moving to the next iteration.* The iteration step is incremented ($k = k+1$), and the execution loops back to the prediction phase.

5. Set-valued consensus protocols for socially-agreed classification

This section moves on to describe how m local monitors, $\mathcal{M}_1, \ldots, \mathcal{M}_m$, operating on board different agents, $\mathcal{A}_1, \ldots, \mathcal{A}_m$, can exploit intercommunication to improve their classification and reach a *socially agreed* decision on the cooperativeness of $\mathcal{A}_i$. This is especially important when, as is usually the case, the individual monitor $\mathcal{M}_h$ does not have enough information to reach a certain conclusion. In the motorway example, this represents the situation in which m local monitors in several vehicles try to work out whether the driver (automatic or human) of a commonly seen or nearby vehicle is following the driving rules or not. In most cases, no vehicle is able to see the entire neighborhood of $\mathcal{A}_i$ and may have enough information to make a decision; however, by combining the estimates of several vehicles, detection may become possible, and a trajectory of $\mathcal{A}_i$ that does not comply with the driving rules may be discovered.

Here, monitors become *nodes* of a *consensus protocol* and exchange information by transmitting messages according to a communication graph $\mathcal{G}_i$. The graph depends on the distance between the vehicles and not on whether the vehicles can see each other or not, thereby the hth monitor can obtain information even from a monitor it cannot see. Exchanged messages are integral and authentic, i.e., their content is correct, and the identity of the sending monitor is guaranteed (cf. the discussion in the conclusion). Assuming that each monitor/node $\mathcal{M}_h$ shares with its communication neighbors, or *c-neighbors*, the output of the locally assessed occupancy estimator map, η^h_i, the remainder of this section discusses what operations and under what conditions enable the design of each consensus node. The key symbols used are summarized in Table 9 in Appendix E.

5.1. Design requirements for the consensus

A first requirement in developing the protocol is to conform to the distributed nature of $\mathcal{P}$. Consequently, the consensus protocol needs to be distributed, i.e., not involve a centralized process, and be based on consensus nodes that use only the information reconstructed by each monitor $\mathcal{M}_h$ or received from their c-neighbors. A common approach in these contexts (Olfati-Saber et al., 2007) is to design consensus nodes as *iterative* processes that elaborate, at every given consensus step κ, a limited amount of information. In particular, here, each hth monitor/node is assumed to have a *consensus state* $X_h \in \mathrm{Pow}(\mathcal{Q})$ that is initialized with the latest estimate of the occupancy map, i.e., $X_h[0] = \eta_i^h[k]$; at every κ, the node extracts the information received in the messages of its c-neighbors, suitably updates its state $X_h[\kappa]$, and shares it back. This is performed up to a maximum step $\overline{\kappa}$, to be quantified, and is intended to allow all participating monitors to improve their classification of $\mathcal{A}_i$ and to agree on a socially accepted decision. Moreover, this decision should be the same as that of a hypothetical, but not present, central process that uses all estimates, $\eta_i^1[k], \ldots, \eta_i^m[k]$, together. Note that κ is a step-index that is independent of the temporal index k used in the monitor.

A second point to consider is related to the *existence* of a centralized decision and the ability of the decentralized consensus nodes to reach it through iteration. In this regard, an estimate $\overline{\eta}_i$ of the occupancy of all (i, j)th topologies is *global* if it is simultaneously based on the data, $\eta_i^1[k], \ldots, \eta_i^m[k]$, of all local monitors. A well-defined centralized decision on the cooperativeness of $\mathcal{A}_i$ is obtained only if a global estimate $\overline{\eta}_i$ is *uniquely defined*. This requirement imposes conditions on the type of data to be handled and the operation used to combine it, as clarified qualitatively below. Regarding first the type of data, it is worth noting that the outputs of the local monitors are continuous sets in $\mathrm{Pow}(\mathcal{Q})$, and not of real scalars or vectors as is typical in traditional consensus algorithms (Olfati-Saber et al., 2007); therefore, a *set-valued* consensus framework as in Fagiolini et al. (2015) must be adopted. In the motorway example, the outputs of the local monitors are sets comprising points in $\mathcal{Q}$, continuous regions that are subsets of $\mathcal{Q}$ where the presence or absence of other vehicles is required, or continuous regions indicating possible configurations of nearby vehicles engaged in platooning.

Moreover, with regard to the type of operation, the following should be considered. Suppose that $\bowtie^2$ is a binary operation used to combine the estimates, η_i^1 and η_i^2, obtained by two monitors that are observing the behavior of $\mathcal{A}_i$, i.e., suppose that $\eta_i^{'} = \eta_i^1 \bowtie^2 \eta_i^2$ is a new estimate that is hopefully more accurate than each η_i^1 and η_i^2 taken individually. Now, if a third estimate η_i^3 is available from another monitor, there are at least six ways to define a global $\overline{\eta}_i$ using the binary operator $\bowtie^2$; these estimates are: $(\eta_i^1 \bowtie^2 \eta_i^2) \bowtie^2 \eta_i^3$, $(\eta_i^1 \bowtie^2 \eta_i^3) \bowtie^2 \eta_i^2$, $(\eta_i^2 \bowtie^2 \eta_i^1) \bowtie^2 \eta_i^3$, $(\eta_i^2 \bowtie^2 \eta_i^3) \bowtie^2 \eta_i^1$, $(\eta_i^3 \bowtie^2 \eta_i^1) \bowtie^2 \eta_i^2$, $(\eta_i^3 \bowtie^2 \eta_i^2) \bowtie^2 \eta_i^1$; other global estimates can be found by associating a different order between the parentheses, e.g., as $\eta_i^1 \bowtie^2 (\eta_i^2 \bowtie^2 \eta_i^3)$ and $\eta_i^1 \bowtie^2 (\eta_i^3 \bowtie^2 \eta_i^2)$. One way to overcome this ambiguity is to introduce a ternary operator $\bowtie^3$ which generates the output $\eta_i^{'} = \bowtie^3 (\eta_i^1, \eta_i^2, \eta_i^3)$. Although this reasoning can be extended in principle for any number of input arguments, and it may be assumed that each monitor $\mathcal{M}_h$ is provided with all the resulting operators, such a solution is not practical in the present context. Indeed, each monitor/node $\mathcal{M}_h$ has a number of c-neighbors that varies over time and depends on the communication range; if the operator to be used would depend on this number, every monitor $\mathcal{M}_h$ should *always* know the exact number of other nodes that are participating in the consensus, in order to select the proper operator. Unfortunately, this information is centralized and may be unavailable due to regulation-based privacy restrictions. Finally, this strategy would require all nodes to process the estimates in the very same order and would generally introduce more complexity in the node algorithm.

Another aspect also affecting the second point above is the heterogeneity between the *convergence times* of any two nodes/monitors $\mathcal{M}_1$ and $\mathcal{M}_2$. Qualitatively speaking, the time within which a node has enough information to build a centralized estimate $\overline{\eta}_i$ is proportional to the maximum number of hops required for all information to reach it. This number varies widely from monitor to monitor and strongly depends on its current visibility. Thus, it may be the case that the state X_1 of a monitor $\mathcal{M}_1$ and those of all its c-neighbors are converging to some value, while the consensus process on a monitor $\mathcal{M}_2$ is still in a transient phase. If no specific property is satisfied by the operator used by any monitor to update its states, a phenomenon that may occur is that, when $\mathcal{M}_2$ eventually converges, $\mathcal{M}_1$ may have reverted to another transient; this would lead to an undesirable, possibly endless oscillation. One solution is to avoid the transient and require all nodes to process all received estimates in the same order, which again is a centralized solution. Conversely, it is desirable that when a monitor/node and its c-neighbors have reached the same value, $\eta_i^1 = \cdots = \eta_i^m$, implying that a consensus state is already forming locally, this value is automatically kept indefinitely.

The above issues fall within the general requirements of a set-valued consensus algorithm Fagiolini and Bicchi (2013). They are solved simultaneously if the merging of any two occupancy estimates, η_i^1 and η_i^2, can be obtained by a binary operator $\bowtie$ satisfying the following properties:

- (*idempotency*) $\eta_{i,1} \bowtie \eta_{i,1} = \eta_{i,1}$, which ensures that a consensus value that is established between any monitor/node $\mathcal{M}_h$ and its c-neighbors is maintained indefinitely;
- (*commutativity*) $\eta_i^1 \bowtie \eta_i^2 = \eta_i^2 \bowtie \eta_i^1$, ensuring that the updated state of each hth monitor/node is independent of the order in which messages are received by it;

- (*associativity*) $\eta_i^1 \bowtie (\eta_i^2 \bowtie \eta_i^3) = (\eta_i^1 \bowtie \eta_i^2) \bowtie \eta_i^3$, where η_i^3 is a third occupancy estimate, guaranteeing that a ternary operator constructed on any composition of $\bowtie$ gives the same result.

If the three properties hold, one can define the unique and global centralized estimate

$$\overline{\eta}_i = \left(\cdots\left(\left(\eta_i^1 \bowtie \eta_i^2\right) \bowtie \eta_i^3\right) \bowtie \cdots\right) \bowtie \eta_i^m = \bowtie_{h=1}^{m} \eta_i^h, \quad (16)$$

where the compact form on the right-hand side is unambiguous since the order in which each pair of input arguments is combined does not change the output. Having listed the properties of $\bowtie$, its design will be discussed in the next subsection.

A final aspect to consider is the uncertainty due to imperfect synchronization between monitors and differences in their measurement noises. A practical way to deal with this uncertainty is to introduce a *consensus tolerance* parameter χ. This is used to conservatively expand the extension of all point measurements of the continuous states of $\mathcal{A}_i$ and detected $\mathcal{A}_i$-neighbors, as well as the occupied/free regions of each (i, j)th topology. Precisely, each measured point q_m is replaced by a sphere of radius χ centered at q_m, while the boundaries of each free/occupied space in the (i, j) th topology are lifted outward by a quantity χ. This is done by a *lifting* map Lift $(\cdot)$ that will defined later. Parameter χ is to be tuned based on simulation, as discussed for the mismatch tolerance ϵ used in the previous section.

5.2. Technical explanation of consensus nodes

The derivation of the hth consensus node requires defining the following few objects:

- A *lifting* map Lift $(\cdot)$ that, given an occupancy estimate $\eta_i = \{\eta_{i,\ell}\}_\ell$ and a consensus tolerance χ, returns the set of the elements $\eta_{i,\ell}^*$ enlarged by a quantity χ; this is obtained as:

$$\begin{aligned} \text{Lift} : \text{Pow}(\mathcal{Q}) \times \mathbb{R}^+ &\to \text{Pow}(\mathcal{Q}) \\ \left(\eta_i^h, \chi\right) &\mapsto \left\{\eta_{i,\ell}^* \in \text{Pow}(\mathcal{Q})\right\}, \end{aligned} \quad (17)$$

where $\eta_{i,\ell}^*$ is the sphere of radius χ centered at $\eta_{i,\ell}$, if $\eta_{i,\ell}$ is singleton, or it equals the set $\eta_{i,\ell}$ after being lifted outward by χ, otherwise;

- An *occupancy merging* map that, given two occupancy estimates η_i^1 and η_i^2, generates a new occupancy estimate η_i^+, by applying for each jth component the operator

$$\begin{aligned} \bowtie : Z_i \times Z_i &\to Z_i \\ \left(\eta_{i,j}^1, \eta_{i,j}^2\right) &\mapsto \eta_{i,j}^+, \end{aligned} \quad (18)$$

with $Z_i = \text{Pow}(\mathcal{Q}) \times \text{Pow}(\mathcal{Q}) \times \text{Pow}(\mathcal{Q}) \times \mathbb{B}$, whose description is detailed below.

Technically speaking, the reasoning behind the construction of a new and more accurate occupancy estimate $\eta_i^{'}$ is as follows. At any consensus step κ, any two consensus nodes have current estimates η_i^1 and η_i^2 of the occupancy map. Recall that each $\eta_i^h = (\eta_{i,j}^h, \ldots, \eta_{i,\kappa_i}^h)$ and each jth component is either of the form $\eta_{i,j}^h = \{I_{i,j}^h, W_{i,j}^h\}$, thus containing only known information, or $\eta_{i,j}^h = \{I_{i,j}^h, W_{i,j}^h, \overline{W}_{i,j}^h, \overline{s}^j\}$, and thus also contains information deducted. Here, *known* information refers to data that has been initially measured by a local monitor or confirmed in a previous merging of two estimates, while *deducted* information refers to data that has been inferred by a local monitor or through a previous merging of two consensus nodes' estimates but has not yet been confirmed by known data of any consensus node. Accordingly, for the hth consensus node, each $I_{i,j}^h$ is the known input set of all visible neighbors that lay in the (i, j)th topology, $W_{i,j}^h$ is the visible portion of the (i, j)th topology, $\overline{W}_{i,j}^h$ is a non-visible portion of the (i, j)th topology where the presence $(\overline{s}^j = 1)$ or absence $(\overline{s}^j = 0)$ of other robots is required. In addition, to account for the different measurement noise affecting each local monitor and the possible incomplete synchronization between consensus nodes, each continuous state $\overline{q}^h \in \mathcal{Q}$, initially estimated by a monitor $\mathcal{M}_h$, is mapped into a sphere $\overline{q}^{h,*} \in \text{Pow}(Q)$, of radius χ and center $\overline{q}^h$, which belongs to $I_{i,j}^h$. This is done using the lift map defined above.

Now, consider first the case where both estimates contain only known information. Adopting a simpler notation here for the ease of exposition, let the two estimates be $x_1 = (I_1, W_1)$ and $x_2 = (I_2, W_2)$, and let $x_+ = (I_+, W_+)$ be the new estimate. To begin with, in the region visible by both consensus nodes, i.e., $W_1 \cap W_2$, any sphere $\overline{q}^{1,*} \in I_1$ has, by construction, a nonempty intersection with one sphere $\overline{q}^{2,*} \in I_2$, and vice versa; within that region, each pair of such spheres can be replaced by their intersection $\overline{q}^{1,*} \cap \overline{q}^{2,*}$ and included as known information in I_+. Then, in the subregions $W_1 \setminus W_2$ and $W_2 \setminus W_1$, where only one consensus node has visibility, all spheres $\overline{q}^{h,*}$ can only be directly included in I_+ without any refinement. Furthermore, since both known regions of the two consensus nodes have been used, the overall known area can be updated to their union, i.e., $W_+ = W_1 \cup W_2$.

As a second case, suppose now that the second estimate also contains deducted information, i.e., $x_2 = (I_2, W_2, \overline{W}_2, s_2)$; consistently, suppose also that the new estimate has the form $x_+ = (I_+, W_+, \overline{W}_+, s_2)$. The known information is treated as above. It is now essential to understand what information can be extracted by combining the known data of the first node with the deducted one of the former. To this regard, such data may be *confirmed* by known data of the first consensus node, *refined* (thus leaving again a deducted yet more precise information), *disconfirmed* (hence also revealing an uncooperative behavior of $\mathcal{A}_i$), or left *invariant*. These cases happen as follows:

- ($s_2 = 1$) $\overline{W}_2$ *describes a region where the second node has deducted the presence of a robot.* First, one can try to see if there exists at least one sphere $\overline{q}^{1,*} \in I_1$ that belongs to $\overline{W}_2$; if so (case 1), the deduction is confirmed and the region $\overline{W}_2$ can be safely removed as it is already represented by the included sphere $\overline{q}^{1,*}$, also $\mathcal{A}_i$ can be classified as cooperative; if not, one can try to check if $\overline{W}_2$ is totally visible from the first node, which happens when $\overline{W}_2 \subseteq W_1$; in such a case (case 2), the deducted information is disconfirmed by the known data of the first node, which also means that $\mathcal{A}_i$ is uncooperative; if $\overline{W}_2$ is not totally visible from the first node (case 3), the presumed robot may lay in the remaining hidden portion, which is added to the new estimate as a deducted, yet more precise, information through the smaller set $\overline{W}_+ = \overline{W}_2 \backslash W_1$; finally, if $W_1 \cap \overline{W}_2 = \emptyset$ (case 4), nothing can be said and the deducted information is left invariant, i.e., $\overline{W}_+ = \overline{W}_2$. In these last two cases, $\mathcal{A}_i$ is classified as possibly-cooperative.
- ($s_2 = 0$) $\overline{W}_2$ *describes a region where the second node has deducted the absence of a robot.* First, one can try to see if there exists at least one sphere $\overline{q}^{1,*} \in I_1$ that belongs to $\overline{W}_2$; if so (case 1), the deduction is disconfirmed and $\mathcal{A}_i$ can be classified as uncooperative; if not (case 2), one can verify if $\overline{W}_2$ is totally visible from the first node, which happens when $\overline{W}_2 \subseteq W_1$, in which case the deducted information is confirmed and the region $\overline{W}_2$ can be safely removed as it will be included in the visible region $W_+ = W_1 \cup W_2$; also, $\mathcal{A}_i$ can be classified as cooperative; if $\overline{W}_2$ is not totally visible from the first node (case 3), the absence of robots must still be satisfied in the remaining hidden portion, which is added to the new estimate as a deducted, yet more precise, information through the smaller set $\overline{W}_+ = \overline{W}_2 \backslash W_1$; finally, if $W_1 \cap \overline{W}_2 = \emptyset$ (case 4), as above, the deducted information is left invariant, i.e., $\overline{W}_+ = \overline{W}_2$. In these last two cases, $\mathcal{A}_i$ is classified as possibly-cooperative.

In all these cases, the new visible part is updated by including the visible regions of both nodes, i.e., $W_+ = W_1 \cup W_2$. The other settings in which the first estimate also includes deduced information, or both estimates do it, follow the above discussion directly, noting that the same reasoning applies *mutatis mutandis*. This is encoded in the occupancy merging map $\bowtie$ defined above. It is finally worth noting that $\bowtie$ is idempotent, commutative, and associative.

Having said this, all monitors that are trying to determine whether or not $\mathcal{A}_i$ is cooperative can become nodes of a distributed consensus algorithm that allows them to share information and reach a social agreement. More precisely, given the ith communication graph $\mathcal{G}_i(\mathcal{W}_i, \mathcal{E}_i)$, where $\mathcal{W}_i = \{1, \ldots, m\}$ is a set of vertices representing the m monitors/nodes that are trying to classify the behavior of $\mathcal{A}_i$, and where $\mathcal{E}_i$ is a set of edges that describes the available communication links. Denote with $d_{h,j}$ the distance between two vertices $h, j \in \mathcal{W}_i$, i.e., the minimum number of vertices to traverse in order to go from h to j or vice versa; denote with $\mathrm{diam}(\mathcal{G}_i)$ the graph *diameter* being the maximum distance between any two vertices, i.e., $\mathrm{diam}(\mathcal{G}_i) = \max_{h,j \in \mathcal{W}_i} d_{h,j}$; in addition, given a vertex $h \in \mathcal{W}_i$, the vertices which are connected via any sequence of p edges are termed the p-hop c-neighbor of h and are described as $w_h(p) = \{j \in \mathcal{W}_i | d_{h,j} \leq p\}$.

Assuming that $\mathcal{G}_i$ is *undirected*, i.e., any present link is bidirectional and allows a monitor/node to send and receive messages from another connected monitor/node, and is *connected*, i.e., there exists a multi-hop link connecting any two vertices in $\mathcal{W}_i$, the following result can be shown, the proof of which can be found in Appendix C:

Theorem 2. Social agreement on the centralized occupancy estimate. *The collective consensus state* $X = (X_1, \ldots, X_m)$ *initialized with the value* $(\eta_i^1, \ldots, \eta_i^m)$ *and updated according to the iterative rule*

$$X_h[\kappa + 1] = \bowtie_{\ell \in w_h(1)} \; X_\ell[\kappa], \text{ for } \kappa > 0, \tag{19}$$

converges, in at most $\overline{\kappa} = \mathrm{diag}(\mathcal{G}_i)$ *steps, to the steady-state value* $\mathbf{1}_m \overline{\eta}_i$ *of social agreement where* $\overline{\eta}_i$ *is the centralized occupancy estimate obtained as in* (16).

5.3. Algorithmic description of consensus nodes

From a procedural viewpoint, the steps to be performed are the following:

▷ *Initialization Phase.* Initialize the consensus step ($\kappa = 0$). The hth monitor/node initializes its consensus state $X_h[0]$ with the occupancy estimate $\eta_i^h = \{\eta_{i,\ell}^h\}_\ell$ obtained via (9), once it has been lifted by the consensus tolerance χ: this is done via the formula:

$$X_h[0] = \mathrm{Lift}(\eta_i^h, \chi);$$

▷ *Consensus Update Step.* After incrementing the consensus step ($\kappa = \kappa + 1$), the hth nodes updated its current occupancy estimate $\eta_i^h[\kappa]$ using data received from its c-neighbors; this is obtained by applying the following formula:

$$\eta_{i,j}^h[\kappa + 1] = \bowtie_{\ell \in w_h(1)} \; \eta_{i,j}^\ell[\kappa], \quad \forall j = 1, \cdots \kappa_i;$$

▷ *Loop back.* The execution loops back to the update step.

6. Autonomous forklifts in a warehouse

In this section, the case study of an industrial warehouse is presented to show how the proposed methodology works. Figure 2 considers n autonomous forklifts moving products from conveyor belts (source points) to storage stacks

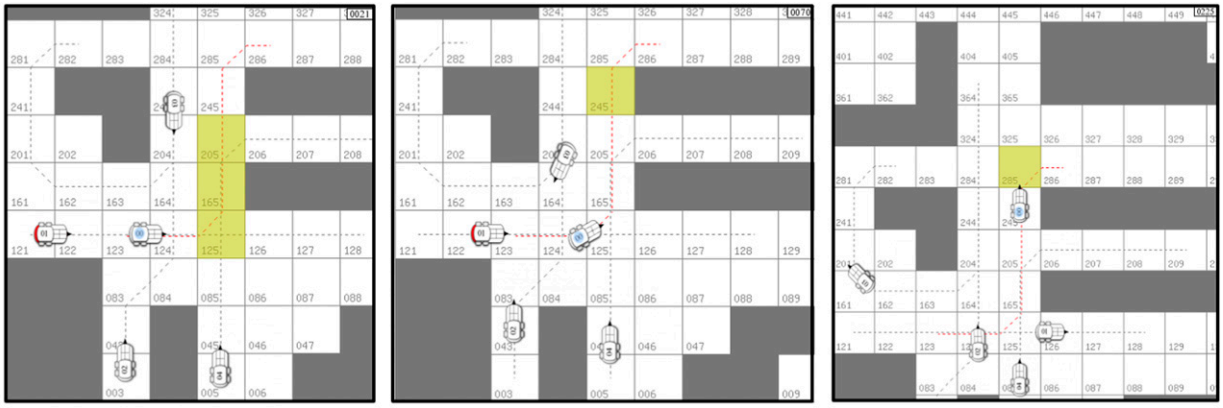

Figure 2. Warehouse case-study. Five forklifts are supposed to cooperate as follows: forklifts $\mathcal{A}_0$ and $\mathcal{A}_1$ need to negotiate entrance to the cells 123 and 125; forklifts $\mathcal{A}_1$ and $\mathcal{A}_2$ for the cell 124; forklifts $\mathcal{A}_1$ and $\mathcal{A}_4$ for the cell 125; forklifts $\mathcal{A}_0$ and $\mathcal{A}_4$ for the macro-cell {125, 165, 205}.

(storage points). The industrial environment consists of a matrix of *cells* and *macrocells*. Each cell is a square area accessible by a single forklift at any one time, while each macrocell is a corridor simultaneously accessible by several forklifts moving in the same direction. Each forklift $\mathcal{A}_i$ is assigned a *path*, which is a sequence of adjacent cells and macrocells and which connects the sources to storage points. When the paths of two forklifts intersect, each forklift must follow a protocol $\mathcal{P}_i$ to avoid collisions cooperatively, thereby priority to any forklift approaching from the right is given. Specifically, when a forklift $\mathcal{A}_j$ is approaching from the right, $\mathcal{A}_i$ must decelerate (D), or otherwise accelerate (A) to a maximum speed of v^m. Forklifts have omnidirectional sensors that measure the continuous states of neighboring forklifts within a distance R_i.

6.1. Cooperation protocol and local monitor explicit derivation

The protocol is based on the following assumptions: (1) the ith path consists of connected line segments, (2) the ith forklift is initialized and always remains on the path, (3) orientation changes of the path are executed instantaneously on the spot by the robot, (4) the ith decoder map u_i is independent of the continuous states of neighboring forklifts. The first three assumptions allow a general dynamics map to be considered for each forklift but reduce the required maneuvers in $\mathcal{P}_i$ only to two, which helps illustrating the local monitor derivation; the fourth assumption implies that no discretization scheme is needed.

The environment is $\mathcal{Q} = \mathbb{R}^2 \times SO(2) \times \mathbb{R}$. A generic point in the environment is denoted as $q = (x, y, \theta, v)$. Referring to the ith forklift, the components of $\mathcal{P}_i$ and their operation are as follows. Given the continuous state $q_i = (x_i, y_i, \theta_i, v_i)$ and a safety distance d_i, the topology set is $\eta_i = (\eta_{i,1})$ with

$$\eta_{i,1} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q})$$
$$q_i \mapsto \left\{ q \middle| (x - x_i)^2 + (y - y_i)^2 \le d_i, \; -\frac{\pi}{2} \le \arctan\left(\frac{y - y_i}{x - x_i}\right) - \theta_i \le \frac{\pi}{4} \right\},$$

and the neighborhood is $\mathcal{N}_i = \eta_{i,1}(q_i)$; the encoder map is $s_i = s_{i,1}$ with

$$s_{i,1} : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}$$
$$(q_i, I_i) \mapsto \oplus_{q_\ell \in I_i} \mathbf{1}_{\eta_{i,1}(q_i)}(q_\ell);$$

the visibility map is

$$\mathcal{V}_i : \mathcal{Q}^n \to \mathrm{Pow}(\mathcal{Q})$$
$$(q_1, \ldots, q_n) \mapsto \left\{ q \middle| (x - x_i)^2 + (y - y_i)^2 \le R_i \right\},$$

with $R_i > d_i$; given the alphabet of events $E_i = \{e^{i,1}, e^{i,2}\}$, the encoder index set is $\Gamma_i = \{\gamma_{i,1}, \gamma^*_{i,1}\}$, with $\gamma_{i,1} = \gamma^*_{i,1} = 1$, so that the detector map is

$$e_i : \mathbb{B} \to \mathrm{Pow}(E_i)$$
$$s_i \mapsto \begin{cases} e^{i,1} \text{ if } \neg s_{i,1}, \\ e^{i,2} \text{ if } s_{i,1}; \end{cases}$$

The discrete state set is $\Sigma_i = \{\mathrm{A}, \mathrm{D}\}$ and the automaton is

$$\delta_i : \Sigma_i \times \mathrm{Pow}(E_i) \to \Sigma_i$$
$$\begin{aligned} (\mathrm{A}, e^{i,1}) &\mapsto \mathrm{A}, \\ (\mathrm{A}, e^{i,2}) &\mapsto \mathrm{D}, \\ (\mathrm{D}, e^{i,1}) &\mapsto \mathrm{A}, \\ (\mathrm{D}, e^{i,2}) &\mapsto \mathrm{D}, \end{aligned}$$

with initial state $\sigma_i^0 = \mathrm{D}$; the decoder map is

$$u_i : \mathcal{Q} \times \Sigma_i \to \mathcal{U}_i$$
$$\begin{aligned} (q_i, \mathrm{A}) &\mapsto (-\mu(v_i - v^m), 0), \\ (q_i, \mathrm{D}) &\mapsto (-\mu v_i, 0), \end{aligned}$$

where μ is a positive constant, and the dynamics map is

$$f_i : \mathcal{Q} \times \mathcal{U}_i \to \mathrm{Pow}(\mathcal{Q})$$
$$(q_i, u_i) \mapsto (v_i \cos\theta_i, v_i \sin\theta_i, \omega_i, a_i).$$

The controlled dynamics map is then

$$f_i^* : \mathcal{Q} \times \Sigma_i \to \mathrm{Pow}(\mathcal{Q})$$
$$(q_i, \mathrm{A}) \mapsto (v_i \cos\theta_i, v_i \sin\theta_i, 0, -\mu(v_i - v^m)),$$
$$(q_i, \mathrm{D}) \mapsto (v_i \cos\theta_i, v_i \sin\theta_i, 0, -\mu v_i).$$

and its solution $q_i(t) = \phi_{f_i^*}(q_i[k], \sigma_i[k])$, for $t \in [0, \Delta]$, is

$$\begin{cases} x_i(t) = x_i[k] + F(\sigma_i[k], t)\cos(\theta_i(0)), \\ y_i(t) = y_i[k] + F(\sigma_i[k], t)\sin(\theta_i(0)), \\ \theta_i(t) = \theta_i(0), \\ v_i(t) = V(\sigma_i[k], t) + v_i[k]e^{-\mu t}, \end{cases} \tag{20}$$

with $F(\mathrm{A}, t) = v^m t + \frac{v_i[k] - v^m}{\mu}\alpha(t)$, $F(\mathrm{D}, t) = \frac{v_i[k]}{\mu}\alpha(t)$, $V(\mathrm{A}, t) = v^m\alpha(t)$, $V(\mathrm{D}, t) = 0$, and $\alpha(t) = 1 - e^{-\mu t}$.

Moreover, the construction of a monitor $\mathcal{M}_h$ is based on the following. The visibility check map is $v_i = (v_{i,1})$, with

$$v_{i,1} : \mathcal{Q} \times \mathrm{Pow}(\mathcal{Q}) \to \mathbb{B}$$
$$(q_i, V_h) \mapsto \begin{cases} 1 & \text{if } \eta_{i,1}(q_i) \subseteq V_h, \\ 0 & \text{otherwise,} \end{cases}$$

The known encoder map is $\widehat{s}_i = \widehat{s}_{i,1}$ with

$$\widehat{s}_{i,1} : \mathcal{Q} \times \widehat{\mathcal{Q}^{n_i}} \to \mathbb{B}$$
$$\left(q_i, I_i^h\right) \to \oplus_{q_\ell \in I_i^h} \mathbf{1}_{\eta_{i,1}(q_i)}(q_\ell),$$

The event estimator is

$$\widehat{e}_i : \mathbb{B} \times \mathbb{B} \to \mathbb{B}^2$$
$$(\widehat{s}_i, v_i) \mapsto \begin{pmatrix} \neg\widehat{s}_{i,1} \\ \widehat{s}_{i,1} v_{i,1} \oplus \neg v_{i,1} \end{pmatrix},$$

and finally the nondeterministic automaton is initialized with the discrete state value $\{\mathrm{A}, \mathrm{D}\}$ and its dynamics is

$$\Delta_i : \mathrm{Pow}(\Sigma_i) \times \mathrm{Pow}(E_i) \to \mathrm{Pow}(\Sigma_i)$$
$$\begin{array}{l} \left(\mathrm{A}, e^{i,1}\right), \left(\mathrm{D}, e^{i,1}\right), \left(\{\mathrm{A}, \mathrm{D}\}, e^{i,1}\right) \mapsto \mathrm{A}, \\ \left(\mathrm{A}, e^{i,2}\right), \left(\mathrm{D}, e^{i,2}\right), \left(\{\mathrm{A}, \mathrm{D}\}, e^{i,2}\right) \mapsto \mathrm{D}, \\ \left(\mathrm{A}, \left\{e^{i,1}, e^{i,2}\right\}\right), \left(\mathrm{D}, \left\{e^{i,1}, e^{i,2}\right\}\right), \\ \left(\{\mathrm{A}, \mathrm{D}\}, \left\{e^{i,1}, e^{i,2}\right\}\right) \end{array} \mapsto \{\mathrm{A}, \mathrm{D}\}.$$

6.2. Performance evaluation

The validity and performance of the proposed methodology have been evaluated in a representative industrial warehouse. Assuming the average size of 1.5 × 2 m for a forklift, which can be located in any orientation, it is reasonable to assume that the warehouse layout consists of cells that are twice the largest size of the forklift, i.e., 4 × 4 m. Then, let us suppose that the entire warehouse, or a part of it, consists of a grid of 40 × 20 cells, leading to an overall size of 160 × 80 m. Also, suppose that a varying number of fixed obstacles, such as storage points, let accessible cells be limited to a number ranging from 30 to 70% of the total, which roughly equates to 120 to 280 free cells. A variable number of forklifts is considered, out of which the first, $\mathcal{A}_0$, misbehaves, and the other ones (varying in number from two to 10) must try to find it out as they move towards their destination.

Precisely, each forklift is assigned with a path of length 10 cells and initialized at the center of the starting cell, and must follow $\mathcal{P}$ in order to avoid collisions when accessing shared cells or macrocells; conversely, $\mathcal{A}_0$ is programmed to stop upon accessing the first shared resource, thus possibly generating a (partial) deadlock. The other forklifts are assumed to run their local monitors and possibly their consensus nodes. In addition, given the large number of possible combinations, several initial configurations have been randomly generated, varying in density of free cells, number of forklifts/monitors, and the activation or non-activation of consensus nodes, all characterized by having at least one monitor in the vicinity of $\mathcal{A}_0$ and a forklift (possibly the same one) that must share a cell/macrocell with it during the simulation run.

The methodology has been statistically evaluated in terms of several metrics, by analyzing 50 datasets that are generated according to the following rationale: with regard to the density of the environment, five different values (30%, 40%, 50%, 60% and 70%) representing the percentage of free cells have been taken into account; with regard to the number of forklifts/monitors, values from 1 to 10 have been used. This leads to the 50 datasets indicated above. Moreover, each dataset comprises 1024 simulations obtained using an equal number of randomly generated initial configurations, characterized by the same value of environment density ρ and number of forklifts/monitors m. Finally, a communication graph $\mathcal{G}_0$ is randomly generated per dataset, with the only requirement to be connected.

As a first metric, the *detection success rate* S_1 has been analyzed with respect to the various datasets. It should be recalled here, as is implied by the theory, that in the proposed approach, each robot is considered to be possibly cooperative unless proven otherwise. Then, in this scenario, the result of the classification of $\mathcal{A}_0$ by any monitor can either be uncooperative, when the monitor has sufficient visibility to discover the absence of another forklift blocking $\mathcal{A}_0$, or possibly-cooperative otherwise. In order to measure the improvement achieved by the exchange of information among the monitors, the number of successes and failures has been assessed both before and after the start of the consensus algorithm: in the former case, detection is counted as successful if at least one local monitor is able to discover the misbehavior of $\mathcal{A}_0$, and unsuccessful otherwise; in the latter case, it is counted as successful if the

Table 1. Warehouse case-study. Performance analysis with respect to the number of monitors m, environment density ρ, and the activation of the consensus algorithm.

		W/out consensus		With consensus	
m	ρ (%)	$S_1^m(\%)$	$S_1^d(\%)$	$S_1^m(\%)$	$S_1^d(\%)$
1	30	9.7	3.2	-	-
1	40	10.3	3.9	-	-
1	50	9.1	2.9	-	-
1	60	8.5	4.2	-	-
1	70	8.2	3.6	-	-
2	30	15.6	1.6	35.5	2.1
2	40	16.0	1.7	36.6	1.4
2	50	14.7	3.2	38.4	0.9
2	60	15.0	1.7	37.5	1.1
2	70	12.8	4.1	37.3	2.7
3	30	19.3	2.8	48.1	1.6
3	40	19.9	3.6	47.9	1.3
3	50	19.8	2.0	47.8	0.9
3	60	18.4	1.9	46.5	2.0
3	70	16.7	3.7	46.1	2.2
4	30	23.9	2.0	58.8	0.9
4	40	23.9	2.0	57.2	1.7
4	50	23.1	1.8	57.3	1.4
4	60	22.9	1.9	55.0	1.8
4	70	20.0	2.7	54.7	0.9
5	30	27.3	1.5	67.4	1.0
5	40	27.3	1.7	66.2	1.0
5	50	27.7	1.1	64.9	1.4
5	60	25.3	2.5	63.4	1.0
5	70	23.3	3.1	61.0	1.4
6	30	30.2	2.3	76.7	1.0
6	40	30.9	2.5	74.1	0.5
6	50	29.2	2.0	71.5	0.8
6	60	28.8	3.2	68.6	1.3
6	70	26.5	1.6	65.3	1.2
7	30	33.6	1.6	84.5	0.4
7	40	32.7	1.1	81.8	0.9
7	50	32.3	1.8	77.8	0.5
7	60	32.0	1.9	74.8	0.6
7	70	30.5	3.1	69.4	0.6
8	30	35.9	1.0	90.9	0.7
8	40	35.8	1.9	87.3	0.7
8	50	35.1	1.7	82.9	0.9
8	60	33.5	1.7	78.6	0.6
8	70	32.0	2.2	73.9	0.5
9	30	38.3	2.6	95.5	0.3
9	40	38.0	1.8	91.5	0.4
9	50	38.0	1.8	85.8	0.6
9	60	35.1	1.0	80.8	1.0
9	70	33.8	1.4	74.7	0.8
10	30	39.5	1.8	99.1	0.6
10	40	40.0	1.0	93.7	0.5
10	50	39.3	1.6	89.0	0.9
10	60	37.0	2.4	82.1	0.9
10	70	35.1	2.4	76.3	1.2

centralized occupancy estimate $\overline{\eta}_0$, reached by all consensus nodes, leads to a misbehavior discovery, and unsuccessful otherwise.

The results of the analysis are shown in Table 1, which reports the average values S_1^m and standard deviations S_1^d of the success rate S_1 for each different dataset. A typical simulation run with $m = 4$ monitors, density $\rho = 30\%$, and activated consensus nodes is graphically depicted in Figure 3. At least four notable trends can be extrapolated, three of which concern S_1^m and one concerns S_1^d:

- Firstly, it can be seen that the average success rate S_1^m increases, for a constant value of the environment density ρ, as the number m of monitors increases. From an a posteriori inspection of the simulations, it can be stated that with a greater number of monitors moving in the environment shared with $\mathcal{A}_0$, it is more likely that at least one of them is in the right configuration to be able to detect the misbehavior; in a sentence, the method *works better with more monitors*;
- Secondly, it can be observed that the average success rate decreases, albeit slightly, with increasing density. It can be seen that the presence of more obstacles justifies this decrease and consequently also by the lower visibility of the monitor; shortly, in a more object-rich environment, the malfunctioning robot is *more likely to hide evidence* of its misbehavior;
- The third observed trend is related to the use of the consensus algorithm and, thus, the sharing of information among monitors. Besides confirming that, as expected, the average success rate S_1^m always increases when consensus is activated, it occurs also that the increase is stronger for larger values of m. Furthermore, for $m \leq 3$, a relative decrease in the success rate continues to occur when the density ρ increases, but this phenomenon is much less present, in proportion, for larger values m; in a nutshell, by social agreement in a more populated environment gives the methodology *robustness and efficiency*;
- A final aspect concerns the standard deviation σ_E, which, with or without the consensus activated, increases with the density of the environment, i.e., the success rate deviation of all simulations within each dataset from the mean value is larger; however, it is important to say that S_1^d decreases as the number of monitors increases, thus demonstrating a higher *reliability* of the methodology.

As a second step in the evaluation of the methodology, the robustness against uncertainty due to a small model mismatch, measurement noise, and partial desynchronization between the monitors is assessed. To this end, it is interesting to compare the two cases in which $\mathcal{A}_0$ is affected by process and measurement noise but is cooperative in the first case and uncooperative in the second; also, monitors are affected by a small offset disturbing their synchronization.

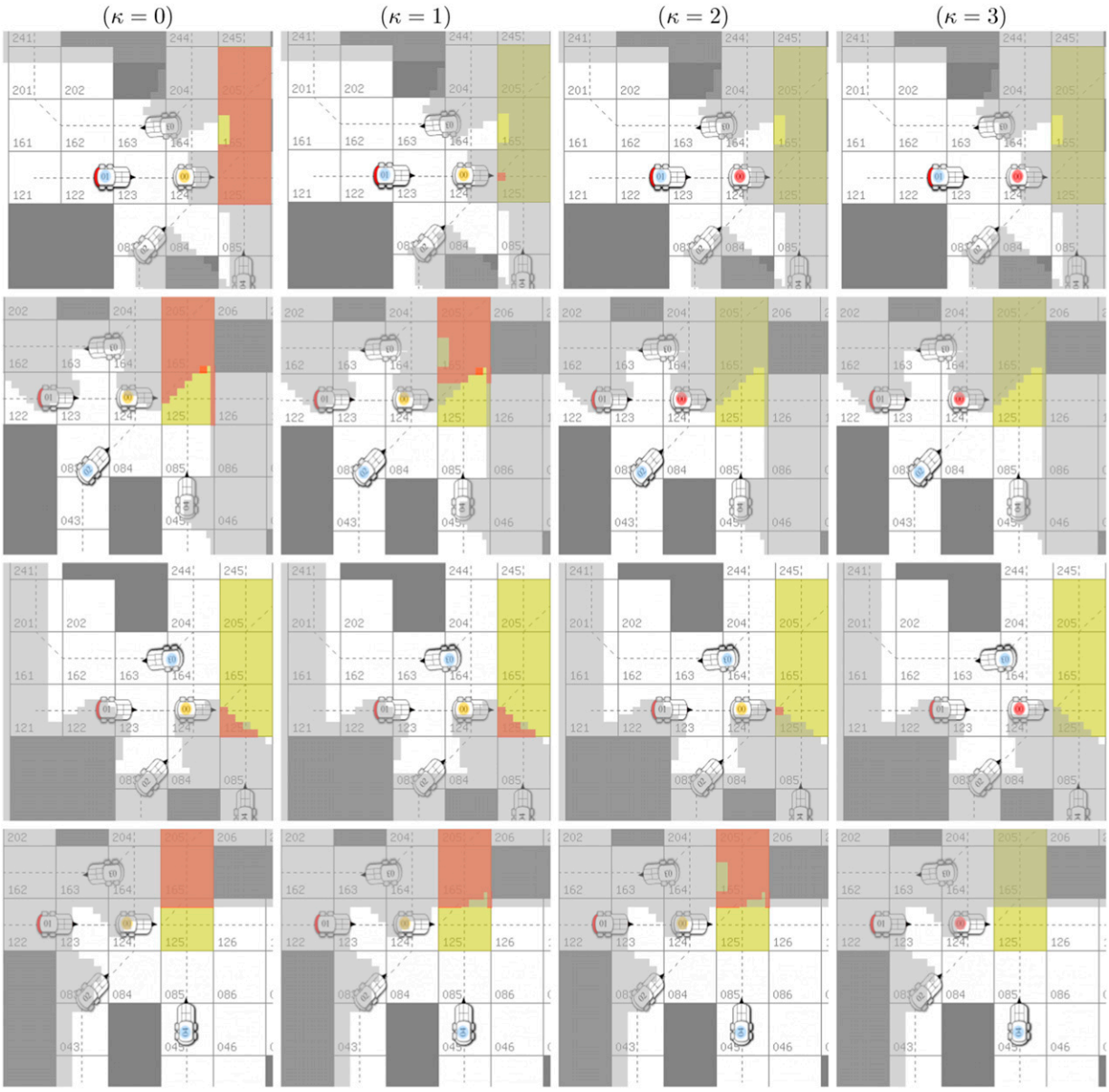

Figure 3. Warehouse case-study. Typical run of simulation with $m = 4$ monitors and environment density $\rho = 30\%$ in which a misbehaving forklift $\mathcal{A}_0$ stops and causes deadlock. The hth row, for $h = 1, \ldots, 4$, refers to the hth forklift with its onboard monitor and consensus node. The κth column illustrates the collective consensus state $X(\kappa)$, with $X(0) = \eta_0^h$ being the map initially estimated by every $\mathcal{M}_h$, which graphically shows how the occupancy map is estimated and improved as the consensus step κ progresses. A blue circle indicates the current monitor, while a yellow/red circle on $\mathcal{A}_0$ indicates if the classification returns possibly-cooperative or uncooperative, respectively. It can be observed that all forklifts reach a social agreement on the misbehavior of $\mathcal{A}_0$.

To better highlight the possible performance degradation, it is worth considering scenarios with a number of forklifts/monitors and an environment density as in the dataset best performing in the previous evaluation step, i.e., with $m = 10$ and $\rho = 30\%$. Then, denoting with $\mathbb{I}_4$ the identity matrix of order 4, model mismatch, measurement noise, and monitor desynchronization are, as common, assumed to be described by Gaussian random signals, white and zero-averaged, with covariance matrices given by $\alpha_1\mathbb{I}_4$, $\alpha_2\mathbb{I}_4$, and $\alpha_3\mathbb{I}_4$, respectively, such that each α_ℓ denotes the 2-norm of the ℓth matrix. That is, all datasets with uncertainty have been obtained in this way: the continuous state q_0 is updated according to the modified ODE $\dot{q}_0 = f_0(q_0, u_0) + \mu_{0,1}$, where $\mu_{0,1}(t)$ is the model mismatch signal, the measured behavior is $\tilde{q}_0^{\text{meas}}(0, \Delta) + \tilde{\mu}_{0,2}(0, \Delta)$, where $\mu_{0,2}(t)$ is the measurement noise signal, and the effect of desynchronization occurs as $\tilde{q}_0^{\text{meas}}(-|\mu_{0,3}|, \Delta - |\mu_{0,3}|)$, where $\mu_{0,3}$(t) is a time offset signal. Moreover, for the sake of tractability, it is assumed in this evaluation that all other forklifts move according to the nominal rules of the cooperation protocol so that the effects generated by the individual agent $\mathcal{A}_0$ can be shown and that all monitors measure the same noisy

Table 2. Warehouse case-study. Performance analysis with $m = 10$, $\rho = 30\%$, and activated consensus, and with respect to the model mismatch covariance norm α_1, the noise covariance norm α_2, and the monitor tolerance ϵ.

			False uncoop.		False coop.	
$10^3\alpha_1$	$10^2\alpha_2$	10ϵ	$S_2^m(\%)$	$S_2^d(\%)$	$S_3^m(\%)$	$S_3^d(\%)$
0.25	0.33	0.33	0.3	0.2	0.1	1.8
0.25	0.33	0.67	0.1	0.1	0.1	2.8
0.25	0.33	1.00	0.0	0.0	0.2	1.0
0.25	0.67	0.33	0.9	0.8	0.1	2.8
0.25	0.67	0.67	0.6	0.4	0.2	1.2
0.25	0.67	1.00	0.2	0.0	0.2	0.2
0.25	1.00	0.33	1.8	0.5	0.1	0.2
0.25	1.00	0.67	1.3	0.7	0.2	0.1
0.25	1.00	1.00	0.9	0.9	0.3	0.2
0.50	0.33	0.33	6.9	6.7	0.3	3.7
0.50	0.33	0.67	5.2	0.8	0.7	0.2
0.50	0.33	1.00	2.6	2.5	0.9	0.1
0.50	0.67	0.33	7.2	6.9	0.4	4.4
0.50	0.67	0.67	6.4	3.1	0.8	1.5
0.50	0.67	1.00	3.1	2.5	1.1	1.6
0.50	1.00	0.33	8.4	1.2	0.5	0.6
0.50	1.00	0.67	6.8	2.9	1.0	1.1
0.50	1.00	1.00	3.9	3.6	1.2	1.7
0.75	0.33	0.33	12.1	9.6	0.7	8.4
0.75	0.33	0.67	9.1	8.7	1.1	2.3
0.75	0.33	1.00	5.7	3.7	1.4	2.4
0.75	0.67	0.33	13.0	0.5	0.8	0.3
0.75	0.67	0.67	12.1	6.3	1.4	0.3
0.75	0.67	1.00	6.1	5.7	1.5	3.0
0.75	1.00	0.33	13.5	6.2	0.9	1.9
0.75	1.00	0.67	12.9	4.8	1.4	7.9
0.75	1.00	1.00	7.2	5.4	1.6	5.2
1.00	0.33	0.33	21.0	7.2	1.4	4.2
1.00	0.33	0.67	16.1	3.6	1.9	1.2
1.00	0.33	1.00	11.0	1.9	2.5	1.7
1.00	0.67	0.33	23.0	3.2	1.3	2.3
1.00	0.67	0.67	17.8	0.6	2.6	0.4
1.00	0.67	1.00	12.2	3.4	2.6	2.6
1.00	1.00	0.33	25.8	1.2	1.7	0.7
1.00	1.00	0.67	19.1	1.9	2.8	1.7
1.00	1.00	1.00	14.0	0.5	2.7	0.0

behavior of $\mathcal{A}_0$. As two further metrics, the rate S_2 of *false uncooperative classifications*, in the first case, and the rate S_3 of *false cooperative classifications*, in the second case, i.e., the number of simulations in which $\mathcal{A}_0$ is incorrectly classified out of the total of 1024 simulations in the dataset; the simulations in which it is classified as possibly-cooperative are not considered as correct classifications. Finally, the evaluation is carried out by varying $\alpha_1 \in [0.0025, 0.0100]$ with a step of 0.0025, $\alpha_2 \in [0.033, 0.100]$ with a step of 0.033, $\alpha_3 \in [0.33, 1.00]$, with a step of 0.33, and using the tolerance parameters ϵ and χ.

The results obtained are listed in Tables 2 and 3. The following interesting trends can be observed:

Table 3. Warehouse case-study. Performance analysis with $m = 10$, $\rho = 30\%$, and activated consensus, and with respect to the monitor desynchronization covariance norm α_3 and the consensus tolerance χ.

		False uncoop.		False coop.	
α_3	χ	$S_2^m(\%)$	$S_2^d(\%)$	$S_3^m(\%)$	$S_3^d(\%)$
0.33	0.33	1.2	0.2	2.0	0.6
0.33	0.67	0.1	0.1	1.6	0.8
0.33	1.00	0.0	0.0	2.1	2.0
0.67	0.33	4.8	2.1	2.6	0.5
0.67	0.67	3.6	0.1	2.7	2.3
0.67	1.00	1.2	0.3	3.2	0.9
1.00	0.33	8.6	0.5	4.1	2.7
1.00	0.67	6.2	2.2	5.2	1.9
1.00	1.00	2.5	0.1	6.1	2.4

- Table 2 shows first that both S_2 and S_3 increase with the model error and the measurement noise, but the latter has a much smaller negative impact on the performance. In addition, it can be seen that increasing the tolerance parameter ϵ reduces, in general, the rate of false uncooperative classifications S_2, but, at the same time, increases that of false cooperative ones S_3. Yet, the increasing trend of S_3 is almost one order smaller than the decreasing trend of S_2, which yields the nice property whereby it is possible to increase ϵ to the extent necessary to reduce S_2, but at the same time not have a large rate of false cooperative classifications S_3, which are the riskiest ones;
- Table 3 shows that both rates S_2 and S_3 grow with increasing desynchronisation between monitors, with S_3 growing slightly less than S_2, although being still of the same order. In addition, it should be noted that increasing the consensus tolerance χ reduces S_2, but at the same time slightly increases S_3. This reveals that synchronization among monitors plays an important role since the more consensus nodes are synchronized, the better the detection system works.

All in all, this second part of the evaluation has shown that proper tuning of the tolerance parameters ϵ and χ gives robustness to the realized detection system, and also that an end-user can focus only on determining these two constants, which are sufficient to collect the effects of uncertainty in a macroscopic way.

6.3. Experiments on a real industrial plant

As a final step, the effective implementability of the warehouse cooperation protocol $\mathcal{P}$ and the distributed misbehavior detection algorithm is demonstrated within the control system of commercial laser-guided vehicles (LGVs), produced by the Italian company Elettric80 S.p.A.; to this end, a representative industrial plant is recreated

where three LGVs that move from starting cells, where virtual stretch-wrappers are located, to final cells with storage points where pallets are collected. Each LGV is characterized by the dynamics map f_i of a unicycle robot and has a steering low-level controller that allows it to keep on the path and move at a desired speed, and that represents the decoder map u_i of the proposed approach; being a commercial vehicle, the decoder is a black box implemented on a Beckhoff TwinCAT3 platform. On top of it, a supervisory control framework is available and is open to be programmed in C# for high-level control of the vehicle. Using data from local proximity sensors, laser measurements for triangulation, and a WiFi connection to a base station, the supervisory control can implement the functions of the encoder map s_i, detector map e_i and automaton map δ_i, thus realizing the cooperative strategy seen above for collision avoidance. Specifically, each LGV has a path consisting of a number of adjacent cells to which it must negotiate access through communication with the other two LGVs. After implementing the last three maps and using the compiled object of the decoder map u_i, the local monitor $\mathcal{M}_h$ has been generated through the rules presented above and the tool in Appendix D, its code has been compiled for the available target processor and then connected to the Beckhoff TwinCAT3 system.

In the experiment reported in Multimedia Extension 1 (described in Table 6 in Appendix A), the three LGVs are assigned with paths whose description in terms of cells and macro-cells, some of which are shared, are transmitted to them by the base station. In their absence of misbehavior, each LGV cooperatively executes the social rules, thus avoiding collisions, and reaches its assigned storage point before returning to its starting cell. In the experiment, one LGV is affected by a control system failure that causes it to stop as soon as it reaches the first shared cell, thus causing the system to stall (this can be seen in the first part of the Extension). Then, the remaining two correct LGVs cooperatively detect the misbehavior through their local monitors and consensus nodes; after agreeing on the misbehavior, they temporarily exclude the fault LGV, renegotiate access to the shared cells, and solve the deadlock (this is visible in the second part of the Extension). Finally, assuming that the faulty LGV control system is restored, either through human intervention or a remote reset, the LGV asks to be readmitted to the cooperation protocol and, once access is granted, can start moving and complete its mission (last part of the

Figure 4. Warehouse case-study. Highlights from the experiment with three commercial LGV within a representative industrial plant. The onboard low-level control of each LGV is encapsulated via software as a black box to represent the decoder map of the system; the supervisory control system is programmed in C# for high-level control of the vehicle to implement the functions of the encoder map s_i, detector map e_i and automaton map δ_i. Once the source code of these three components is written, the local monitor and consensus nodes are generated via the tool described in Appendix D and then linked with the object code of the decoder map to the Beckhoff TwinCAT3 target machine. Experiments have been performed at the premises of the Italian company Elettric80 SpA.

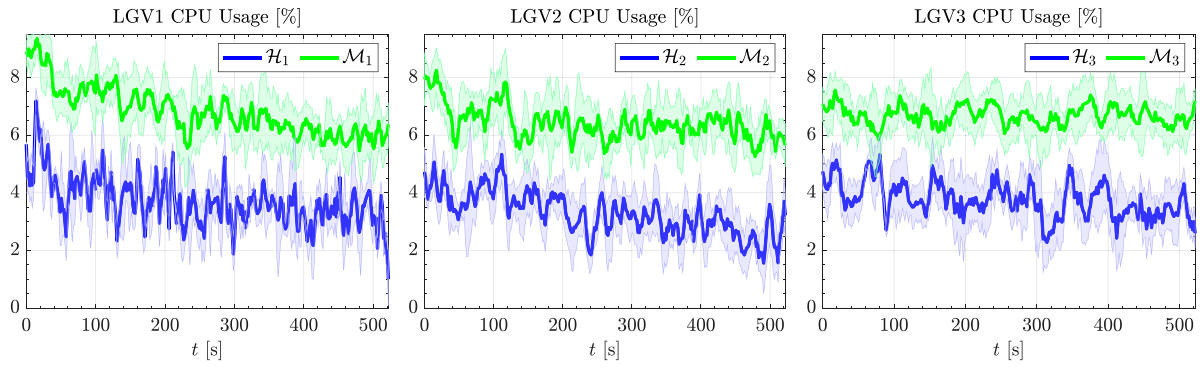

Figure 5. Warehouse case-study. Analysis of the CPU usage from the emulation of 27 settings with commercial LGVs within a representative industrial plant (in the legend, $\mathcal{H}_i$ is the ith process controlling the motion of the LGV, while $\mathcal{M}_i$ is the ith ensemble of the local monitor and the consensus node). The temporal behavior of the average and standard deviation values are graphically depicted. The CPU usage of the sensing process is about twice as high as that of the motion process. Yet, both processes have very low CPU occupancy, which demonstrates effective implementability on the platform.

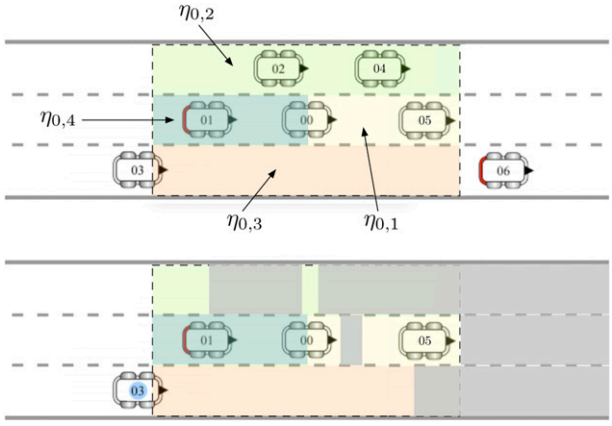

Figure 6. Motorway case-study. Illustration of the first four topologies of a car $\mathcal{A}_0$ (above) and the visibility of a monitor onboard on $\mathcal{A}_3$ (below).

Extension). Some highlights of the experiment are reproduced in Figure 4.

A final indicator to demonstrate the actual applicability of the proposed method in a real system is CPU utilization. In this regard, the Elettric80 company has proprietary software that allows *emulation* of the behavior of a team of LGVs, each controlled by the very same software that is present on the real hardware platforms. In practice, the software simulates the physical dynamics of the LGVs, including various forms of uncertainty, and it collects and sends the simulated signals to the actual control boards on which the coordination algorithm and the misbehavior detector generated by the presented tool are running. This enables the construction of a dataset of virtual evolutions of the robots, but also the access to data such as CPU utilization during all phases of operation.

For this purpose, a dataset of 27 emulations has been generated, with three LGVs moving within the virtual layout of the above-used industrial plant. Each emulation differs by a different assignment of paths to the LGVs. Each LGV has on board two processes: one for the control of its motion and one that monitors the other two LGVs. The CPU utilization of the control and misbehavior detection processes has been measured. To realign the time spans of all emulations, their durations have been truncated to the shortest one (after verifying that no significant information would be lost). Analysis of the resulting data is shown in Figure 5, from which it can be noted that the CPU usage of the detection process is approximately twice larger than that of the coordination process; however, both processes have very low CPU occupancy. These two aspects ultimately show the effective implementability on the platform, as well as the availability of room for implementing further functions.

7. Other applications of the method

7.1. The motorway case-study

In this section, a second case study presents an automotive scenario with n cars traveling along a motorway with m lanes. To avoid collisions and reach their destinations, cars must adhere to European traffic rules, as summarized here. Each car, represented as $\mathcal{A}_i$, possesses a desired travel speed denoted as v_i^d. The driver of each car must perform specific actions, including accelerating (F), decelerating (S), changing lanes to the left (L) or right (R), or entering a platoon (P). Intuitively, when $\mathcal{A}_i$ approaches a slower car $\mathcal{A}_j$ in front and the left lane is unoccupied, it should transition from acceleration (F) to the sequence L, F, R to overtake on the left. If the left lane is occupied by another car $\mathcal{A}_h$, $\mathcal{A}_i$ must switch to S. Furthermore, if $\mathcal{A}_j$ in front has a similar desired travel speed, v_j^d, $\mathcal{A}_i$ should switch to P and adjust its velocity to match that of the preceding car, v_j^d. The rest of this section formalizes the cooperation protocol and shows how the suggested method enables the discovery of vehicle

misbehavior. We assume that agents are subject to a maximum error of 10% and have tuned ϵ accordingly.

Formally, the cooperative protocol $\mathcal{P}$ can be found as follows, referring without loss of generality to Figure 6. The ith continuous state is $q_i = (x_i, y_i, \theta_i, v_i, v_i^d)$, with x_i and y_i its longitudinal and lateral positions, θ_i is its steering angle, v_i is its longitudinal speed. Denoting by d_f and d_b the forward and backward safety distances, the ith set of topologies is as follows: $\eta_{i,1}(q_i)$ is the front region, extending from x_i forward by d_f; $\eta_{i,2}(q_i)$ is the region on the next left lane, extending from x_i backward by d_b and forwards by d_f; $\eta_{i,3}(q_i)$ is the similar region on the right; $\eta_{i,4}(q_i)$ is the rear region extending from x_i downwards by d_b; $\eta_{i,5}(q_i)$ is a further rear region extending downwards by d_{int}, indicating the region of interaction with cars approaching from behind. Formally, denoting with $q = (x, y, \theta, v, v^d)$ a generic point in $\mathcal{Q}$, they are described as follows:

$$
\begin{aligned}
&\eta_{i,1} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i, x_i + d_f], \\
&\qquad y \in \Big[\Big\lfloor \frac{y_i}{w} \Big\rfloor w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w \Big] \Big\}, \\
&\eta_{i,2} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i - d_b, x_i + d_f], \\
&\qquad y \in \Big[\Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 2 \Big) w \Big] \Big\}, \\
&\eta_{i,3} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i - d_b, x_i + d_f], \\
&\qquad y \in \Big[\Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor - 1 \Big) w, \Big\lfloor \frac{y_i}{w} \Big\rfloor w \Big] \Big\}, \\
&\eta_{i,4} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i, x_i + d_{\mathrm{int}}], \\
&\qquad y \in \Big[\Big\lfloor \frac{y_i}{w} \Big\rfloor w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w \Big] \Big\}, \\
&\eta_{i,5} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i - d_{\mathrm{int}}, x_i], \\
&\qquad y \in \Big[\Big\lfloor \frac{y_i}{w} \Big\rfloor w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w \Big] \Big\},
\end{aligned}
$$

where w is the width of a lane, $\lfloor \cdot \rfloor$ is the closest lower integer, and d_{int} the interaction distance. In addition, the following topologies are the front, back, and right regions possibly populated by other vehicles with similar desired speeds:

$$
\begin{aligned}
&\eta_{i,6} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \{ q \,|\, x \in [x_i, x_i + d_{\mathrm{int}}], \\
&\qquad y \in \Big[\Big\lfloor \frac{y_i}{w} \Big\rfloor w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w \Big] \\
&\qquad v^d \in \big[(1 - \phi) v_i^d, (1 + \phi) v_i^d \big] \}, \\
&\eta_{i,7} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \{ q \,|\, x \in [x_i - d_{\mathrm{int}}, x_i], \\
&\qquad y \in \Big[\Big\lfloor \frac{y_i}{w} \Big\rfloor w, \Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor + 1 \Big) w \Big] \\
&\qquad v^d \in \big[(1 - \phi) \quad v_i^d, (1 + \phi) v_i^d \big] \}, \\
&\eta_{i,8} : \mathcal{Q} \to \mathrm{Pow}(\mathcal{Q}) \\
&\quad q_i \mapsto \Big\{ q \,\big|\, x \in [x_i - d_{\mathrm{int}}, x_i], \\
&\qquad y \in \Big[\Big(\Big\lfloor \frac{y_i}{w} \Big\rfloor - 1 \Big) w, \Big\lfloor \frac{y_i}{w} \Big\rfloor w \Big], v^d \le (1 - \phi)\, v_i^d \Big\},
\end{aligned}
$$

where $\phi \in (0, 1]$ is a speed similarity constant. Then, the ith neighborhood is $\mathcal{N}_i = \cup_{\ell=1}^{8} \eta_{i,\ell}(q_i)$ and it is assumed that $\mathcal{N}_i \subset V_i$. The protocol also requires the constants: $\eta_{i,1}^*$ and $\eta_{i,2}^*$ as the left- and right-most lanes, $\eta_{i,3}^*$ and $\eta_{i,4}^*$ as the left and right edges of the current lane, and $\eta_{i,5}^*$ as the ith car being aligned with the center of the current lane; they are formally given by:

$$
\begin{aligned}
\eta_{i,1}^* &= \{ q \,|\, y \in [(m-1)w, mw] \}, \\
\eta_{i,2}^* &= \{ q \,|\, y \in [0, w] \}, \\
\eta_{i,3}^* &= \{ q \,|\, y \ge (\lfloor y_i[k]/w \rfloor + 1) w \}, \\
\eta_{i,4}^* &= \{ q \,|\, y \le (\lfloor y_i[k]/w \rfloor) w \}, \\
\eta_{i,5}^* &= \Big\{ q \,\Big|\, \Big| y - \Big\lfloor \frac{y}{w} \Big\rfloor w - \frac{w}{2} \Big| \le \Delta_y, |\theta| \le \Delta_\theta \Big\},
\end{aligned}
$$

with Δ_x and Δ_y being tolerance parameters. The ith encoder map V_i then is $s_i : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}^{13}$, with $s_i = (s_{i,1}, \ldots, s_{i,13})$. The ith visibility map returns the region of $\mathcal{Q}$ that lies within a distance R_i and is not hidden by other cars (cf. the sweeping line algorithm in Thrun (2002) and Figure 7 for examples).The ith discrete state set is $\Sigma_i = \{\mathrm{F}, \mathrm{S}, \mathrm{L}, \mathrm{R}, \mathrm{P}\}$. All events, the detector map e_i, and automaton δ_i are listed in Table 4.

The ith decoder map $u_i : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to \mathcal{U}_i$, $u_i = (a_i, \omega_i)$ computes suitable feedback laws through the linear acceleration and angular speed, given the current maneuvers. Inspired by Solyom and Coelingh (2013), in order to allow $\mathcal{A}_i$ to execute a platoon maneuver one must distinguish whether $\mathcal{A}_i$ is: ① in the middle, ② the last car, ③ the leader car, and ④ kept in a platoon temporarily, as the next left lane

is occupied. Denote with $\overline{a} > 0$ an acceleration, x_f, v_f, x_b, and v_b longitudinal positions and speeds of the preceding and following car, and b_b, b_f, γ, k^l, and d suitable positive constants. Denote also the abbreviations: $x_i^b = x_i - x_b$, $x_i^f = x_i - x_f$, $v_i^b = v_i - v_b$, $v_i^f = v_i - v_b$, $v_i^d = v_i - v^d$. Then, the acceleration command is

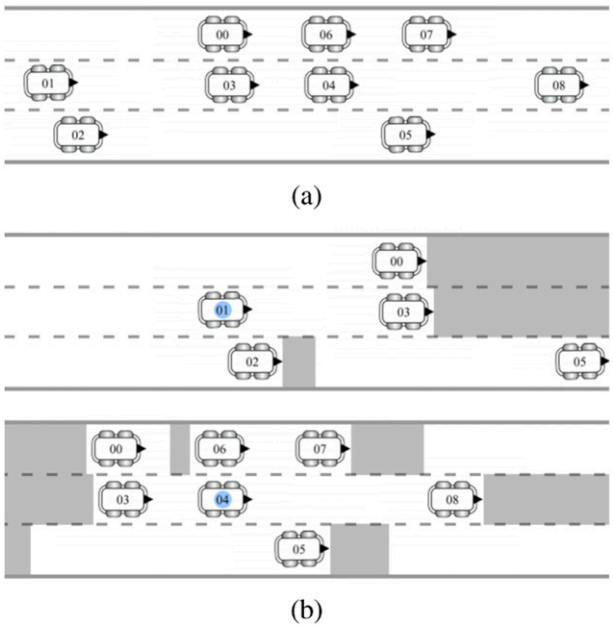

Figure 7. Examples of visibility regions: (a) complete scenario, (b) gray regions are not visible by $\mathcal{A}_1$ and $\mathcal{A}_4$.

Table 4. Motorway case-study: events $e^{i,j}$, Detector map e_i, and automaton δ_i. Self-transitions (e.g., F → F) are omitted for brevity, as they occur in the absence of triggering events.

Event	Detection condition	Transition
$e^{i,1}$	$s_{i,1}\ s_{i,2}\ \neg s_{i,6}\ s_{i,13}$	F → S
$e^{i,2}$	$s_{i,1}\ \neg s_{i,6}\ s_{i,11}\ s_{i,13}$	F → S
$e^{i,3}$	$s_{i,1}\ \neg s_{i,7}\ \neg s_{i,13}$	F → S
$e^{i,4}$	$\neg s_{i,1}\ s_{i,3}\ s_{i,6}\ \neg s_{i,8} \neg s_{i,10}$	F → S
$e^{i,5}$	$s_{i,1}\ \neg s_{i,2}\ \neg s_{i,6}\ \neg s_{i,11}\ s_{i,13}$	F → L
$e^{i,6}$	$\neg s_{i,1}\ \neg s_{i,3}\ \neg s_{i,10}\ s_{i,13}$	F → R
$e^{i,7}$	$\neg s_{i,1}\ s_{i,9}$	S → F
$e^{i,8}$	$\neg s_{i,1}\ \neg s_{i,3}$	S → F
$e^{i,9}$	$s_{i,1}\ \neg s_{i,2}\ \neg s_{i,11}\ s_{i,13}$	S → L
$e^{i,10}$	$\neg s_{i,7}\ s_{i,12}$	L → F
$e^{i,11}$	$\neg s_{i,7}\ s_{i,13}$	R → F
$e^{i,12}$	$\neg s_{i,10}\ \neg s_{i,7}$	P → F
$e^{i,13}$	$\neg s_{i,3}\ \neg s_{i,4}\ s_{i,5}\ s_{i,9}$	P → F
$e^{i,14}$	$\neg s_{i,1}\ s_{i,3}\ \neg s_{i,8}$	P → S
$e^{i,15}$	$s_{i,1}\ \neg s_{i,2}\ \neg s_{i,6}\ \neg s_{i,11}$	P → L
$e^{i,16}$	$s_{i,10}$	L → F
$e^{i,17}$	$s_{i,3}\ s_{i,7}$	F → P
$e^{i,18}$	$s_{i,1}\ s_{i,2}\ \neg s_{i,6}\ s_{i,7}$	F → P
$e^{i,19}$	$s_{i,1}\ \neg s_{i,6}\ s_{i,7},\ s_{i,11}$	F → P
$e^{i,20}$	$s_{i,10}\ s_{i,12}$	L → P
$e^{i,21}$	$s_{i,10}\ s_{i,13}$	R → P

$$a_i : \mathcal{Q} \times \Sigma_i \times Q^{n_i} \to \mathbb{R}$$

$$(q_i, \{\mathrm{F}, \mathrm{L}, \mathrm{R}\}, I_i) \mapsto \begin{cases} \overline{a} & \text{if } v_i < v_i^d, \\ 0 & \text{otherwise,} \end{cases}$$

$$(q_i, \mathrm{S}, I_i) \mapsto \begin{cases} -\overline{a} & \text{if } v_i > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$(q_i, \mathrm{P}, I_i) \mapsto \begin{cases} -b_b\left(x_i^b + \gamma v_i^b\right) - b_f\left(x_i^f + \gamma v_i^f\right) & \text{if } ①, \\ -b_f\left(x_i^f + d + \gamma v_i^f\right) & \text{if } ②, \\ -b_b\left(x_i^b - d + \gamma k^l v_i^d\right) & \text{if } ③, \\ -b_b\left(x_{i,b} - d + \gamma v_i^b\right) + & \\ -b_f\left(x_i^f + d_f + \gamma v_i^f\right) & \text{if } ④, \end{cases}$$

and the angular speed command is

$$\omega_i : \mathcal{Q} \times \Sigma_i \times Q^{n_i} \to \mathbb{R}$$

$$(q_i, \{\mathrm{F}, \mathrm{S}, \mathrm{P}\}, I_i) \mapsto \left((y_i^* - y_i)\frac{\sin\theta_i}{\theta_i} - \mu\,\theta_i\right) v_i,$$

$$(q_i, \mathrm{L}, I_i), \mapsto \begin{cases} \overline{\omega} & \text{if } \theta_i < \theta_{\max} \\ 0 & \text{otherwise} \end{cases},$$

$$(q_i, \mathrm{R}, I_i) \mapsto \begin{cases} -\overline{\omega} & \text{if } \theta_i > -\theta_{\max} \\ 0 & \text{otherwise} \end{cases},$$

where $y_i^* = (\lfloor y_i/w \rfloor + 1/2)w$ is the current lane center, $\theta_{\max}$ is $\mathcal{A}_i$'s maximum curvature angle, μ and $\overline{\omega}$ are suitable positive constants ensuring stability. The expression of ω_i for the maneuvers F and S is obtained using the Lyapunov function $V = 1/2(y^*(q_i) - y_i)^2 + 1/2\theta_i^2$. The ith dynamics map $f_i : \mathcal{Q} \times \Sigma_i \to \mathrm{Tan}(\mathcal{Q})$ is as in Section, with an additional null component for the desired travel speed.

Some examples of *motion misbehavior* that can be discovered with the proposed methodology are described below. For this purpose, the cooperation protocol is implemented using the software tool described in Appendix, and local monitors and consensus nodes are automatically generated and compiled. In Figure 8(a) four cars have different desired speeds so that the P maneuver is never triggered. Car $\mathcal{A}_0$ performs an uncooperative behavior by keeping an F maneuver in the second lane when the next lane to its right is free; according to $\mathcal{P}_i$, the car should start an R maneuver to return to the first lane. The behavior of $\mathcal{A}_0$ performing an F maneuver in the second lane requires that part of the region $\mathcal{A}_{0,3}$ is occupied by at least one other car. Three local monitors try to determine whether this condition is true, notwithstanding their partial visibility. Figure 8(b) illustrates the three estimated occupancy maps, I_0^1, I_0^2, and I_0^3, reconstructed by the local monitors; the figure shows that only the local monitor on board car $\mathcal{A}_3$ successfully discovers the inconsistency, while the other two monitors do not. The third monitor immediately classifies $\mathcal{A}_0$ as uncooperative and the other two as possibly-cooperative.

To continue, in Figure 9, car $\mathcal{A}_1$ is subject to the same misbehavior and four local monitors use the set-valued consensus in Theorem 2, which allows them to agree on

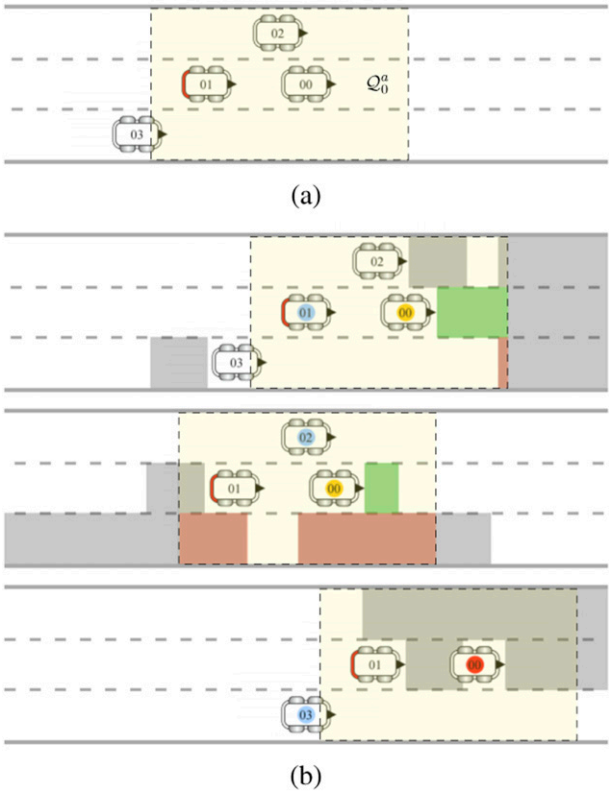

Figure 8. Motorway case-study (#1). (a) Misbehaving car $\mathcal{A}_0$ is in F in the second lane regardless that the next lane on its right is free. (b) Estimated occupancy maps η_0^h, for $h = 1, 2, 3$; only car $\mathcal{A}_3$ discovers the uncooperative behavior of $\mathcal{A}_0$ (red circle), while the other two monitors remain uncertain to possibly-cooperative (yellow circle).

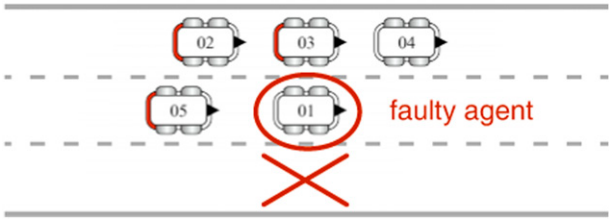

Figure 9. Motorway case-study (#2). Centralized estimate $\overline{\eta}_1$ revealing the behavior of $\mathcal{A}_1$.

the centralized estimate of the occupancy map $\overline{\eta}_1 = \bowtie_{h\in\{2,3,4,5\}}\eta_1^h$. The available communication graph is $\mathcal{G}_1 = (\mathcal{W}_1, \mathcal{E}_1)$, with $\mathcal{W}_1 = \{2, 3, 4, 5\}$ and $\mathcal{E}_1 = \{2 \to w_2 = \{2, 3, 5\}, 3 \to w_3 = \{2, 3, 4\}, 4 \to w_4 = \{3, 4\}, 5 \to w_h = \{2, 5\}\}$. $\mathcal{G}_1$ is connected and $\text{diam}(\mathcal{G}_1) = 3$. The instance of set-valued consensus is $X_h(\kappa + 1) = \bowtie_{\ell\in w_h} X_\ell(\kappa)$, for all $h \in \mathcal{W}_1$. Figure 10 reports the consensus evolution and shows how the four local monitors successfully consent on the centralized estimate $\overline{\eta}_1$ in three steps.

Finally, consider Figure 11(a) where the desired speeds of all cars are similar and a P maneuver is triggered. Here, the output of the decoding map u_i depends on the positions of the preceding and following cars. Thus, each local monitor must consider the exact positions of possible hidden cars. Assuming that no more than one car can be found in the hidden portion of each topology, an equally spaced grid is applied to such a portion. Car $\mathcal{A}_4$ performs an uncooperative behavior, switching from an S to an F maneuver, regardless of the fact that there is no vehicle in front of it, whereas a P should be performed. A local monitor onboard $\mathcal{A}_0$ calculates an estimated occupancy map I_4^0 and merges it via a set-valued consensus with that one of its own neighbors monitor onboard $\mathcal{A}_5$. Figure 11(b) shows how $\mathcal{A}_4$ is first classified as possibly-cooperative and then as uncooperative. Note that, after every consensus step, estimated hidden cars are only kept in a narrower area (gray region), due to the cut of the tolerance ϵ, which removes all the predictions too far from the measured data. Finally, note in Figure 11(c) how a local monitor onboard $\mathcal{A}_6$ can correctly classify the behavior of another car $\mathcal{A}_3$ as cooperative.

7.2. The power grids case-study

In this section, a final case study involves a power grid system with a distributed controller Zeng (2015) and its purpose is to show that the methodology is not limited to self-driving systems. A power grid network comprises many nodes/agents, each of which is a power machine requiring synchronization. The continuous state of the ith power machine and its stabilizer is $q_i = (\delta, \omega, V, x_c)$, with δ a phase angle, ω an angular speed, V a voltage and x_c the state of the controller; the ith decoding map is $u_i = (u_p, u_c)$ where u_p is the control voltage of the power machine and u_c that of the stabilizer. The ith dynamics map is

$$f_i : \mathcal{Q} \times \mathcal{U}_i \times \text{Pow}(\mathcal{Q}) \to \text{Tan}(\mathcal{Q})$$
$$(q_i, u_i, I_i) \mapsto \left(\omega,\ -a\omega + c,\ -V\left(\sum_j C_{i,j}, u_p\right), u_c\right),$$

where index j runs for all $\mathcal{A}_i$-neighbors and $C_{i,j} = V_j Z \sin(\delta_i - \delta_j + \alpha)$, with a, c, Z, and α machine-dependent constants. The ith discrete state set is $\Sigma_i = \{\sigma^1, \sigma^2, \sigma^r\}$, the first two of which are the *modes* where the controller tends to correct V to a nominal value V^*, while in the third one, a flush/reset mode is activated to prevent the controller from removing too much energy from the power machine. Referring to a generic state $q = (\delta, \omega, V, x_c)$, the cooperation protocol uses the following constants topologies: $\eta^*_{i,1} = \{q | x_c(V - V^*) > 0\}$, $\eta^*_{i,2} = \{q | x_c^2 > (V - V^*)^2\}$, $\eta^*_{i,3} = \{q | |x_c| < \overline{\epsilon}\}$, where $\overline{\epsilon}$ is a small constant to be suitably choose. Accordingly, the ith encoder map is $s_i = (s_{i,1}, s_{i,2}, s_{i,3})$. All events, the detector map e_i, and automaton δ_i are listed in Table 5. Denote $D_{i,j} = -(x_{c,i} - x_{c,j})$ sign $(\Delta V \Delta S)$, $\Delta V = x_{c,i}^2 - x_{c,j}^2$, and $\Delta S = x_{c,i} x_{c,j}$. The ith decoder map is $u_i : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to \mathcal{U}_i$, $u_i = (u_p, u_c)$, with

$$u_p : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to \mathbb{R}$$
$$(q_i, \sigma^1, I_i) \mapsto -l_1\ x_c,$$
$$(q_i, \sigma^2, I_i) \mapsto l_1\ x_c,$$
$$(q_i, \sigma^r, I_i) \mapsto 0,$$

and

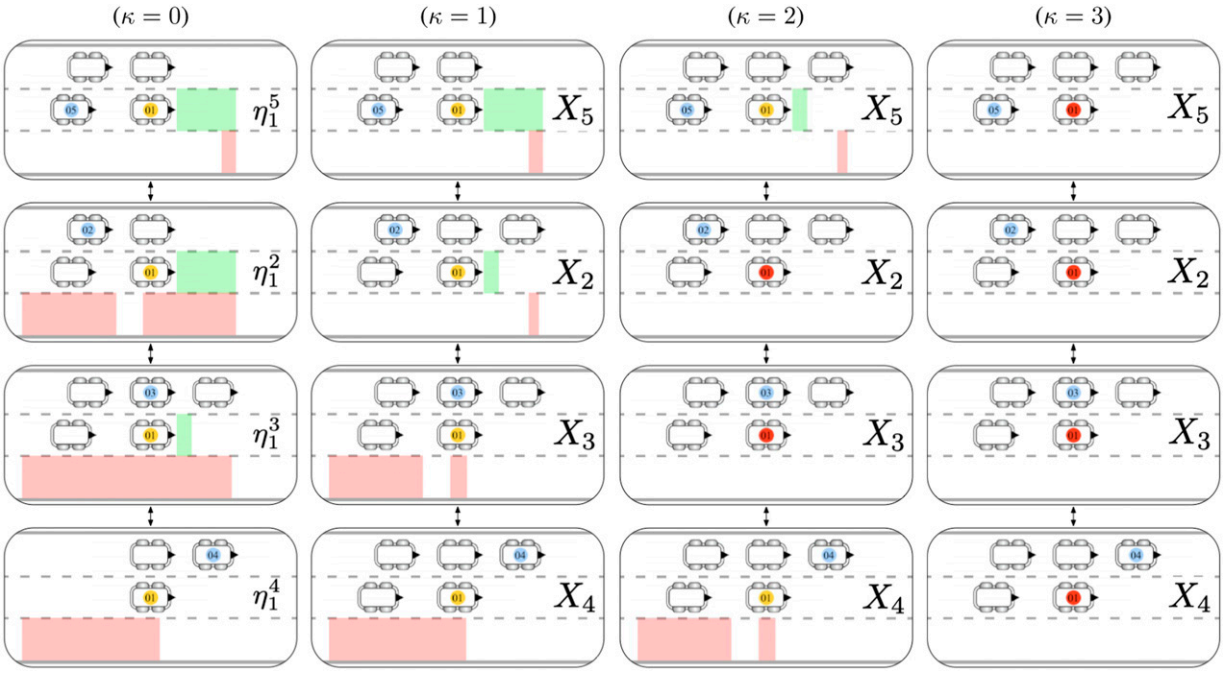

Figure 10. Motorway case-study (#2) - Consensus run. Each row refers to a different local monitor indicated by a blue circle, while each column is a different consensus step. The misbehavior is initially undetected by any local monitor, but is finally discovered after monitors consent on $\overline{\eta}_1$ (a yellow circle indicates possibly-cooperative, while a red one uncooperative).

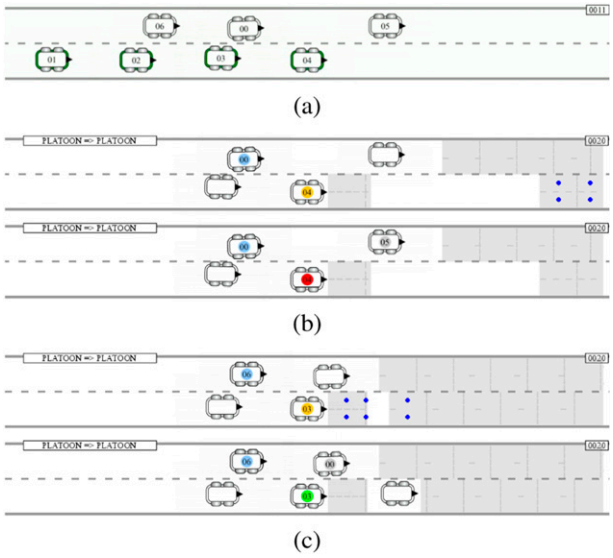

Figure 11. Motorway case-study (#3) (a) scenario with a platoon where a car $\mathcal{A}_4$ wrongly switches from S to F; (b) a local monitor on $\mathcal{A}_0$ reconstructs the estimate η_4^0 (blue dots are possible $\mathcal{A}_0$-neighbors) (above), and merges it with that of $\mathcal{A}_5$, thereby discovering the non-cooperation (below). (c) A local monitor on $\mathcal{A}_6$ correctly classifies $\mathcal{A}_3$ as cooperative, after merging its estimate η_3^6 with that of a local monitor on $\mathcal{A}_0$.

$$
\begin{aligned}
u_c : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} &\to \mathbb{R} \\
(q_i, \sigma^1, I_i) &\mapsto l_1 \left(V - V^*\right) + {}_jD_{i,j}, \\
(q_i, \sigma^2, I_i) &\mapsto -l_1 \left(V - V^*\right) + {}_jD_{i,j}, \\
(q_i, \sigma^r, I_i) &\mapsto -A \quad x_c.
\end{aligned}
$$

Table 5. Power grid case-study: events $e^{i,j}$, detector map e_i, and automaton δ_i. Self-transitions are again omitted for brevity.

Event	Detection condition	Transition
$e^{i,1}$	$\neg r_{i,1}$	$\sigma^1 \to \sigma^2$
$e^{i,2}$	$r_{i,2}$	$\{\sigma^1, \sigma^2\} \to \sigma^3$
$e^{i,3}$	$r_{i,1}$	$\sigma^2 \to \sigma^1$
$e^{i,4}$	$s_{i,1}s_{i,3}$	$\sigma^r \to \sigma^1$
$e^{i,5}$	$\neg s_{i,1}s_{i,3}$	$\sigma^r \to \sigma^2$

Two examples of misbehavior are now considered in a network of five power machines, using the numerical values (Zeng, 2015): $a = 0.625$, $c = 52.2556$, $Z = 51.2579$, $\alpha = 0.113$, $\overline{\epsilon} = 10^{-4}$, $l_1 = 1$, $A = 100$. To begin with, assume that the first four machines operate correctly at a nominal voltage $V^* = 1.4941$, while the fifth wrongly drifts to $V(0) = 1.2$ due to a *voltage peak*. The initial conditions of the correct power machines are $\delta(0) = 0.0$, $\omega(0) = 1.033$, $V(0) = V^*$, and $x_c = 0.0$, and that of the fifth machine differs for $V(0)$. All machines operate initially in the discrete state σ^1. The local monitor is run on the very same machine with a predicting horizon of 0.7 s, and a noise error of 1% affects the measurement of V. Figure 12 shows in the first row the predictions obtained for all discrete states and reveals how the misbehavior is discovered. To continue, assume now that the controller of the fifth machine has a temporary failure described as a unitary pulse signal in u_p for $t \in [0.05, 0.15]$ seconds, which again results in a voltage spike. Figure 12 shows that all predictions differ significantly from the measured data, although they are not distinguishable from each other. Yet, they allow the misbehavior to be

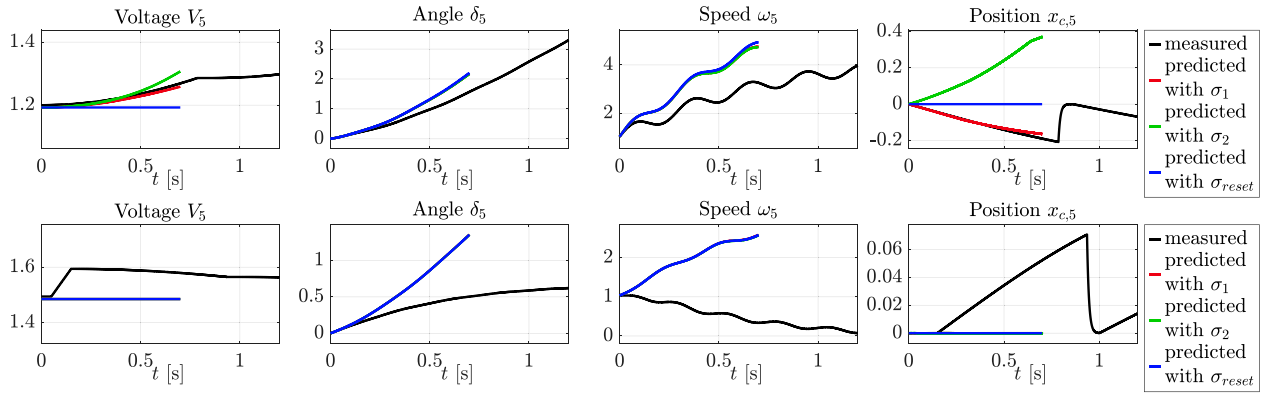

Figure 12. Power grid case-study: (first row) The first four power machines operate correctly at $V_i = 1.4941$, while the fifth one (whose state evolution is reported here) is affected by a system failure causes its voltage to drop to V_5 (0) = 1.2; the predictions of the local monitor (reported for all continuous and discrete states) shows that the misbehavior is discovered. (second row) The controller of the fifth power machine fails and generates a unitary pulse in the signal in u_p for $t \in [0.05, 0.15]$ seconds, which causes a temporary voltage spike; the misbehavior is detected even though the discrete state is not uniquely identified.

detected, even though no unique value is reconstructed for the discrete state.

8. Conclusion and discussion

This work focused on the problem of misbehavior detection in socially organized robotic agents. A new framework was proposed that allows social rules to be formally embedded in the definition of agent models. The framework is general and applies to several domains, from electrical machines to automotive systems, and, most importantly, it allows for the automatic generation of monitoring and agreement processes, which are executed on board each agent and allow for socially-agreed detection of the misbehavior of their neighbors. The effectiveness and robustness of the proposed detection methodology were shown in simulation and experimentally. The rest of the section describes how the computational complexity of the approach scales with social rules and a list of other relevant observations.

8.1. Complexity and scalability analysis

Before delving into the evaluation of the complexity and scalability of the proposed approach, it is worth clarifying that the cooperation protocol $\mathcal{P}$ represents a convenient description framework, made of eight "containers", T_i, V_i, E_i, Γ_i, Σ_i, δ_i, u_i, f_i, which are combined in such a way as to allow systems with rich behaviors to be described in a quite flexible and modular way. In general, the complexity of these components results from the richness of the behaviors that the considered multi-agent system has. Based on this, the complexity of the monitor and consensus nodes are evaluated against that of the elements of $\mathcal{P}$.

8.1.1. Local monitor. To begin with, in order to assess the computational cost of each iteration of monitor $\mathcal{M}_h$, one needs first to break down the specific complexity of each involved component as follows:

- *Computing map* v_i. Each component $v_{i,j}$ needs checking if the (i, j)th topology is in V_h; denoting with $C^v_{i,j}$ the cost of such set inclusion test, the total cost is bounded by $\mathcal{O}((\kappa_i + h_i)C^v_i)$, where $C^v_i = \max_j\{C^v_{i,j}\}$.
- *Computing map* Δ_i. The nondeterministic automaton Δ_i involves by (14) computing the automaton δ_i for every pair of predicted discrete state and event; denoting with C^δ_i the cost of the latter automaton, the total cost is conservatively bounded by $(r_i v_i C^\delta_i)$, where r_i and v_i are the cardinality of Σ_i and E_i.
- *Generating the extended input set* $\widehat{I}^h_i$. A suitable discretization scheme must be adopted to sample the hidden portion of the (i, j)th topology; note that this element is necessary only when the decoder map u_i also depends on the configurations of $\mathcal{A}_i$-neighbors; its total cost is $\mathcal{O}(\overline{\alpha})$, where $\overline{\alpha}$ is the maximum number of sampling points.
- *Computing map* $\widehat{s}_i$. Each component $\widehat{s}_{i,j}$ needs checking if the continuous state of any agent $\mathcal{A}_j$, that is visible from $\mathcal{A}_h$, is in the (i, j)th topology; denoting with $C^s_{i,j}$ the cost of the (i,j)th topology membership test, the total complexity is bounded by $\mathcal{O}((\kappa_i + h_i)C^s_i)$, with $C^s_i = \max_j(C^s_{i,j})$.
- *Computing map* $\widehat{e}_i$. This map requires the indexing of v_i truth tables, one for each binary function of the input vector $\widehat{s}_i$, thus leading to a total cost that is $\mathcal{O}(v_i)$.
- *Computing the occupancy estimator map* η^h_i. In the worst case, κ_i set differences need to be computed between $\eta_{i,j}$ and V_h (note that the test needs to be only done for the first κ_i components as the remaining ones are constant and already known to $\mathcal{M}_h$), thus yielding a total cost of $\mathcal{O}(\kappa_i C^o_i)$, where C^o_i is the cost of a single set difference.

Based on these component costs, the kth iteration of monitor $\mathcal{M}_h$ involves the following:

- *Executing a prediction phase.* In this phase, a forward integration of the controlled dynamics set map F_i^* starting for all $q_{i,\alpha}$ must be performed, contributing with a complexity of $\mathcal{O}(\overline{\alpha}C_i^f)$, along with a one-step calculation of the nondeterministic automaton $\tilde{\delta}_i$, which adds the cost of $\mathcal{O}(v_i C_i^\delta)$; in addition, the evaluation of the forward link map $\mathcal{L}_i$ is constant because it involves only the storage of a bidirectional pointer into the list. Thus, the overall complexity of the prediction phase is $\mathcal{O}(\overline{\alpha}C_i^f + v_i C_i^\delta)$.
- *Executing an update/matching phase.* Here, at most $\overline{\alpha}$ pair-wise tests of ϵ-similarity between $\widehat{q}_{i,\alpha}(t)$ and $\tilde{q}_i^{\text{meas}}(0,\Delta)$ must be performed that are necessary because of the operator $\bowtie_\epsilon$; also, an execution of the occupancy estimator has to be done. Hence, denoting with C_i^ϵ the cost of each pair-wise comparison in $\bowtie_\epsilon$, the overall complexity of this phase is $\mathcal{O}(\overline{\alpha}C_i^\epsilon + \kappa_i C_i^o)$.

Putting all together, it can be established that the total computational cost of each iteration of $\mathcal{M}_h$ is $(\overline{\alpha}(C_i^f + C_i^\epsilon) + v_i C_i^\delta + \kappa_i C_i^o)$, which scales linearly with the complexity of the components f_i and δ_i. A remark should be made about parameter $\overline{\alpha}$ that depends on the sampling scheme: a naive uniform sampling generally leads to high values of $\overline{\alpha}$; these values can be much reduced, with a prior analysis of the cooperation protocol $\mathcal{P}$ via simulation by finding a trade-off between accuracy and computation cost, which is beyond the scope of this work.

8.1.2. Consensus node. A final observation concerns the complexity and scalability of the set-valued consensus. First of all, it should be noted that the set-valued consensus algorithm starts after each time that the local monitor $\mathcal{M}_h$ has calculated an estimated occupancy map η_i^h. Therefore, it does not depend on the number of $\mathcal{A}_i$-neighbors nor the complexity of the dynamics, decoder, automaton, and encoder maps and on the number of events because all these entities have already been accounted for to produce the above estimate.

Another relevant parameter is the number m of local monitors participating in the social agreement. In this regard, it was shown in Theorem 2 that the convergence towards the centralized estimate occurs, under the assumption that the communication graph $\mathcal{G}_i$ is connected, within a number of steps equal to its diameter $\overline{\kappa} = \text{diag}(\mathcal{G}_i)$. Thus, on the whole, the execution of the entire consensus requires each robot to execute at most $\overline{\kappa}$ steps; this number is by definition bounded by the number of nodes, i.e., $\overline{\kappa} \leq m$, although the equality happens when $\mathcal{G}_i$ degenerates to a line. In this respect, it should be noted that current mobile communication technologies (such as those used in MANETs and VANETs) make this occurrence very unlikely. Hence, it can be said that the consensus algorithm scales, in general, linearly with the number of monitors, but its dependence is, in practice, sub-linear.

A final aspect is related to the number of topologies $\kappa_i + h_i$ and the granularity with which the occupancy estimates obtained from each local monitor are represented. Concerning the first parameter, the hth component X_h of the collective state contains at most one list of subsets of $\mathcal{Q}$ for each (i,j)th topology; thus leading to a linear dependence on the sum $\kappa_i + h_i$. With regard to the second parameter, the following should be noted. From a mathematical point of view, the intersection operations involved by the operator $\bowtie$ are to be performed in the continuous space $\mathcal{Q}$; in this way, the calculation is more efficient, and the result is typically of less complexity: starting from any two complex sets, the number of subsets resulting through the intersection typically tends to diminish. However, a precise value cannot be found as it depends both on the complexity of the regions represented by each (i,j)th topology and the visibility condition of each local monitor. A practical assumption leading to a worst-case qualitative evaluation is to assume that each (i,j)th topology is discretized into sub-areas and that $\bowtie$ is applied to the finite number C_η of all such elements; after every consensus step, the number of elements is at most C_η, and thus the overall consensus algorithm scales $\mathcal{O}((\kappa_i + h_i)C_\eta)$.

8.2. Other relevant observations

Firstly, this work has not addressed the automatic generation and verification of social rules from a set of specifications for a multi-robot system. As is well known, these problems are orders of complexity higher than proving theorems (Cook, 1971). Here, the logical conditions of the given specifications do not have a finite number of input combinations, which can, at least in theory, be enumerated combinatorially; conversely, they depend also on time (Kress-Gazit et al., 2009; Platzer, 2010; Rungger et al., 2013) in an implicit way, through the nonlinear evolution of the components of all robots. This makes the enumeration of all cases that may occur impractical, because it would be too computationally expensive, or incomplete, if interrupted after a large number of enumeration steps.

Secondly, dealing with *corrupted messages* that are exchanged by consensus nodes and that contain false information is another challenge that is, however, beyond the scope of this work. Currently, the integrity and authenticity of messages can be ensured by the use of digital signature-based techniques, which protect against fake messages generated and injected by an external intruder, i.e., an external entity pretending to be a legitimate consensus node (Raya et al., 2006). The problem becomes more complicated if an actual member of the group generates the messages. To date, the integrity of the on-board software can be ensured using Trusted Computing Platforms (TCP) (England et al., 2003; ISO Central Secretary, 2009, 2015); these platforms rely on a hardware component with built-in tamper-resistance; they have been standardized in two phases (ISO Central Secretary, 2009, 2015) and are now commercially available (cf., e.g., Infineon's OPTIGA TPC 2.0)

for use in the automotive and mobile robotics sectors (Hoeller and Toegl, 2018; Petri et al., 2016; Schneider et al., 2017; Sumra et al., 2011); in addition, swarm attestation techniques can be used to verify software integrity on cooperating cyber-physical devices in a scalable and secure manner (Ambrosin et al., 2016; Asokan et al., 2015; Wedaj et al., 2019). In the absence of these solutions, tolerating inconsistent messages that could spread is still an open problem; Bicchi et al. (2008) presented a preliminary study into it.

A third aspect is related to the *counteractions* to implement in response to the detection of an uncooperative robot. In sensor networks, a selfish node that does not forward received messages according to the routing protocol, as well as a node that sends inauthentic or compromised messages, can be isolated by invalidating its cryptographic key and discarding its future messages (Pfleeger et al., 2015). Here, because a robot is a physical system that shares the environment $\mathcal{Q}$ with its neighbors, deciding what counteractions to implement in order to escape from or be resilient to an uncooperative robot is arbitrarily complex since logical disconnection from communications is not sufficient: in the warehouse example, a stuck forklift requires redefining the priorities so as to allow other forklifts to overtake it, but in the motorway scenario, a vehicle moving erratically along the road requires a more involved reaction. Possible solutions may involve encoding counteractions directly into the cooperation protocol $\mathcal{P}$ or activating another protocol $\mathcal{P}'$ that is specifically designed to mitigate the effect of an adversary attack. In this work, the activation of counteractions was not considered, as this topic is beyond the scope of this paper. In fact, the work focused on providing a formalism for the modeling of socially organized robots, providing a theoretical basis for the development of a distributed misbehavior detector, and enabling the development of a computer tool for the automatic generation of the detection code.

Additionally, the considered protocols are characterized by an intrinsic *locality* of information, entailed in the definition of each ith neighborhood, which generally prevents the occurrence of cascading phenomena in the misbehavior detection. For example, a vehicle in the leftmost lane proceeding slowly, as well as a vehicle constantly changing lanes unjustifiably, may slow down the entire traffic and thus have a non-local impact on performance. This is generally not the case for the misbehavior detection problem, as a monitor has to determine whether a robot $\mathcal{A}_i$ performs behaviors that conform to $\mathcal{P}_i$ and are consistent with the instantaneous value of its neighborhood $\mathcal{N}_i$. Thus, a generic uncooperative robot $\mathcal{A}_j$, which is a neighbor of $\mathcal{A}_i$ and is observed by monitor $\mathcal{M}_h$, will be classified according to the data available for $\mathcal{M}_h$; meanwhile, as long as $\mathcal{A}_i$ continues to perform cooperative behaviors, monitor $\mathcal{M}_h$ will classify it as possibly or certainly cooperative. Consider, e.g., a correct car $\mathcal{A}_0$ in the motorway scenario. Assume that $\mathcal{A}_0$ is traveling in the second lane to overtake another car $\mathcal{A}_1$; suppose $\mathcal{A}_1$ mistakenly moves into the second lane, violating the minimum distance constraint from $\mathcal{A}_0$, which is approaching from behind. The correct vehicle $\mathcal{A}_0$ then switches to braking mode and slows down. Consistently, all neighboring monitors will evaluate $\mathcal{A}_0$'s behavior as (possibly) cooperative.

To further understand the generality of the proposed method, consider three robots and a human operator that need to grasp and move a deformable yet fragile object cooperatively. Assume that the four agents approach and grasp the object from one side each and that only adjacent agents can communicate with one another. The proposed approach can model this system and, thereby, allow the misbehavior of any of these four agents to be discovered. In this respect, intuitively, the complex physical dynamics of the robots and the human can be described by dynamic maps f_i, with $i \in Z = \{\text{robot-1, robot-2, robot-3, human}\}$; the sets of controllers allowing them to carry out different actions, i.e., approach the object, establish safe contact, and effectively grasp it and move it, can be described by decoding maps g_i, with $i \in Z$; the conditions of proximity of each agent's end-effector to the object and those that ensure force closure on the object without its damage can be modeled by four sets of topological maps $\eta_{i,j}$, with $i \in Z$; finally, the logic by which each agent switches between one action to another can be included in the encoding maps e_i with $i \in Z$. Then, the ability of each robot to measure its own continuous state (including its joint and end-effector states), the contact force with the object, and the end-effector positions of the two adjacent robots or of the nearby human operator can be described by three visibility maps, $V_{\text{robot-1}}$, $V_{\text{robot-2}}$, and $V_{\text{robot-3}}$; likewise, the ability of the human operator to measure its own continuous state and to estimate the contact force with the object along with the end-effector positions of the two adjacent robots is modeled in a fourth visibility map V_{human}. Furthermore, misbehavior of any of the four agents, such as the application of contact forces that are too small or too large, due to incorrect regulation of their end-effector position, can be discovered via monitor and consensus nodes automatically generated through our approach. This allows solving such a task even when no single agent has enough information to detect the misbehavior, which occurs since agents cannot simultaneously measure all other end-effector positions; in this context, first, local monitors enable every single agent to obtain continuous sets representing possible force ranges, and then, consensus nodes allow to combine them iteratively and to finally discover the possible misbehavior.

It should also be noted that existing solutions are, in theory, applicable to specific scenarios in which the cooperative model is linear, the event maps are linear, or there is only one robot, etc. However, a performance comparison between the present solution and any of them can only be made on a case-by-case basis.

In conclusion, one main reason for adopting the proposed method is that it is the only tool that integrates at least the following features into a unique solution: (1) the ability to describe socially organized cooperative multi-agent systems, with possibly nonlinear physical dynamics, different types of nature (mechanical, hydraulic, electrical,

cyber-physical, etc.) and rather general rules of interaction; (2) the ability to automatically generate the code for local monitor and consensus processes that do not require knowing the internal expressions of the agents' dynamics and controllers. The first feature is achieved through the adoption of the cooperation protocol $\mathcal{P}$, which enables behavior richness and flexibility to be made available for the user, who can decide how to decompose the model into its components. The second feature builds upon the inherently modular nature of $\mathcal{P}$ and is fundamental, since the functions appearing in the decoder maps are typically part of a closed software library, provided by robot manufacturers or control developers, and can only be invoked as black boxes. In this regard, the proposed solution implements the idea of enhancing safety and security in multi-robot systems through rule-based cooperation, which has been advocated for at least three decades (see, e.g., the seminal work by Shoham and Tennenholtz (1995). It allows model components to be easily changed and reused, which also makes it helpful in analyzing the threats and weaknesses of a multi-agent system, even before its actual implementation.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been funded by Project SELF4COOP (Self-optimizing Networked Edge Control for Cooperative Vehicle Autonomy), funded by Italian MUR PRIN 2022 under grant number 2022SKLZAY.

ORCID iDs

Adriano Fagiolini https://orcid.org/0000-0001-9943-1975
Antonio Bicchi https://orcid.org/0000-0001-8635-5571

Supplemental Material

Supplemental material for this article is available online.

References

Agarwal S, Vora A, Pandey G, et al. (2020) Ford multi-av seasonal dataset. *The International Journal of Robotics Research* 39(12): 1367–1376. DOI: 10.1177/0278364920961451.

Ajoudani A, Zanchettin AM, Ivaldi S, et al. (2018) Progress and prospects of the human–robot collaboration. *Autonomous Robots* 42(5): 957–975.

Ambrosin M, Conti M, Ibrahim A, et al. (2016) Sana: secure and scalable aggregate network attestation *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY: Association for Computing Machinery, 731–742.

Ashkenazi Y, Dolev S, Kamei S, et al. (2023) Forgive and forget: self-stabilizing swarms in spite of byzantine robots. *Concurrency and Computation: Practice and Experience* 35(11): e6123.

Asokan N, Brasser F, Ibrahim A, et al. (2015) Seda: scalable embedded device attestation. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. New York, NY: Association for Computing Machinery, 964–975.

Ayanian N (2019) Dart: diversity-enhanced autonomy in robot teams. *The International Journal of Robotics Research* 38(12-13): 1329–1337. DOI: 10.1177/0278364919839137.

Balluchi A, Benvenuti L, Di Benedetto M, et al. (2002) Design of observers for hybrid systems. *Hybrid Systems: Computation and Control* 2289: 76–89.

Behley J, Garbade M, Milioto A, et al. (2021) Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: the semantickitti dataset. *The International Journal of Robotics Research* 40(8-9): 959–967. DOI: 10.1177/02783649211006735.

Besselink B and Johansson KH (2015) Control of platoons of heavy-duty vehicles using a delay-based spacing policy. *IFAC-PapersOnLine* 48(12): 364–369.

Bicchi A, Fagiolini A, Dini G, et al. (2008) Tolerating, malicious monitors in detecting misbehaving robots *2008 IEEE International Workshop on Safety, Security and Rescue Robotics*. New York, NY: IEEE, 109–114.

Bicchi A, Fagiolini A and Pallottino L (2010) Towards a society of robots. *IEEE Robotics and Automation Magazine* 17(4): 26–36.

Bossens DM, Ramchurn S and Tarapore D (2022) Resilient robot teams: a review integrating decentralised control, change-detection, and learning. *Current Robotics Reports* 3(3): 85–95.

Bourne JR, Goodell MN, He X, et al. (2020) Decentralized multi-agent information-theoretic control for target estimation and localization: finding gas leaks. *The International Journal of Robotics Research* 39(13): 1525–1548. DOI: 10.1177/0278364920957090.

Bressan A and Piccoli B (2007) *Introduction to the Mathematical Theory of Control*. Springfield, MO: American Institute of Mathematical Sciences, Vol. 1.

Cai P, Luo Y, Hsu D, et al. (2021) Hyp-despot: a hybrid parallel algorithm for online planning under uncertainty. *The International Journal of Robotics Research* 40(2-3): 558–573. DOI: 10.1177/0278364920937074.

Caporale D, Settimi A, Massa F, et al. (2019) Towards the design of robotic drivers for full-scale self-driving racing cars. *Proceedings of International Conference on Robotics and Automation*. New York, NY: IEEE, 5643–5649.

Cassandras CG and Lafortune S (2006) *Introduction to Discrete Event Systems*. Secaucus, NJ: Springer-Verlag New York, Inc.

Christensen AL, Ogrady R and Dorigo M (2009) From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation* 13(4): 754–766. DOI: 10.1109/TEVC.2009.2017516.

Cook SA (1971) The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. New York, NY: Association for Computing Machinery, 151–158.

Cortes J, Martinez S, Karatas T, et al. (2004) Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation* 20(2): 243–255.

Di Paola D, Gasparri A, Naso D, et al. (2015) Decentralized dynamic task planning for heterogeneous robotic networks. *Autonomous Robots* 38(1): 31–48. DOI: 10.1007/s10514-014-9395-y.

Dong X and Sitti M (2020) Controlling two-dimensional collective formation and cooperative behavior of magnetic microrobot swarms. *The International Journal of Robotics Research* 39(5): 617–638. DOI: 10.1177/0278364920903107.

Doyen L and Rapaport A (2001) Set-valued observers for control systems. *Dynamics and Control* 11(3): 283–296.

Duz A, Phillips S, Fagiolini A, et al. (2018) Stealthy attacks in cloud-connected linear impulsive systems. *2018 Annual American Control Conference (ACC)*. New York, NY: IEEE, 146–152.

Eilers S, Mårtensson J, Pettersson H, et al. (2015) Companion–towards co-operative platoon management of heavy-duty vehicles. *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. New York, NY: IEEE, 1267–1273.

England P, Lampson B, Manferdelli J, et al. (2003) Cover feature - a trusted open platform. *Computer* 36: 55–62.

Fagiolini A and Bicchi A (2013) On the robust synthesis of logical consensus algorithms for distributed intrusion detection. *Automatica* 49(8): 2339–2350.

Fagiolini A, Dubbini N, Martini S, et al. (2015) Convergence analysis of distributed set-valued information systems. *IEEE Transactions on Automatic Control* 61(6): 1477–1491.

Fathian K, Summers TH and Gans NR (2018) Robust distributed formation control of agents with higher-order dynamics. *IEEE Control Systems Letters* 2(3): 495–500.

Ferraro A and Scordamaglia V (2023) A set-based approach for detecting faults of a remotely controlled robotic vehicle during a trajectory tracking maneuver. *Control Engineering Practice* 139: 105655.

Fourlas G, Kyriakopoulos K and Krikelis N (2002) Diagnosability of hybrid systems. *Proceedings of the IEEE Mediterranean Conference On Control and Automation*. New York, NY: IEEE.

Gasparri A, Paola DD, Giua A, et al. (2011) Consensus-based decentralized supervision of petri nets. *2011 50th IEEE Conference on Decision and Control and European Control Conference*. New York, NY: IEEE, 1128–1135. DOI: 10.1109/CDC.2011.6161340.

Goebel R, Sanfelice RG and Teel AR (2009) Hybrid dynamical systems. *IEEE Control Systems* 29(2): 28–93.

Greenberg A (2016) *The Jeep Hackers are Back to Prove Car Hacking can Get Much Worse*. San Francisco, CA: Wired Magazine.

Guo P, Kim H, Virani N, et al. (2018) Roboads: anomaly detection against sensor and actuator misbehaviors in mobile robots. *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. New York, NY: IEEE, 574–585.

Gupta V, Langbort C and Murray RM (2006) On the robustness of distributed algorithms. *Proceedings of the 45th IEEE Conference on Decision and Control*. New York, NY: IEEE, 3473–3478.

Hoeller A and Toegl R (2018) Trusted platform modules in cyber-physical systems: on the interference between security and dependability. *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. New York, NY: IEEE, 136–144.

ISO Central Secretary (2009) Information technology - trusted platform module - part 1: overview. Standard ISO/IEC 11889-1:2009, Intl. Organization for Standardization, Geneva, CH. https://www.iso.org/standard/50970.html.

ISO Central Secretary (2015) Information technology - trusted platform module library - part 1: architecture. Standard ISO/IEC 11889-1:2015, Intl. Organization for Standardization, Geneva, CH. https://www.iso.org/standard/66510.html.

Jadbabaie A, Lin J and Morse A (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control* 48(6): 988–1001.

Kabir AM, Thakar S, Malhan RK, et al. (2021) Generation of synchronized configuration space trajectories with workspace path constraints for an ensemble of robots. *The International Journal of Robotics Research* 40(2-3): 651–678. DOI: 10.1177/0278364920988087.

Kabzan J, Valls MI, Reijgwart VJ, et al. (2020) Amz driverless: the full autonomous racing system. *Journal of Field Robotics* 37(7): 1267–1294.

Kaur U, Zhou H, Shen X, et al. (2021) Robomal: malware detection for robot network systems. *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*. New York, NY: IEEE, 65–72.

Khalastchi E and Kalech M (2018) On fault detection and diagnosis in robotic systems. *ACM Computing Surveys* 51(1): 1–24.

Khalastchi E and Kalech M (2019) Fault detection and diagnosis in multi-robot systems: a survey. *Sensors* 19(18): 4019.

Ko WR, Jang M, Lee J, et al. (2021) Air-act2act: human–human interaction dataset for teaching non-verbal social behaviors to robots. *The International Journal of Robotics Research* 40(4-5): 691–697. DOI: 10.1177/0278364921990671.

Kress-Gazit H, Fainekos GE and Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics* 25(6): 1370–1381.

Kuderer M, Gulati S and Burgard W (2015) Learning driving styles for autonomous vehicles from demonstration. *Proceedings of the IEEE International Conference on Robotics and Automation*. New York, NY: IEEE, 2641–2646.

Lau H, Bate I, Cairns P, et al. (2011) Adaptive data-driven error detection in swarm robotics with statistical classifiers. *Robotics and Autonomous Systems* 59(12): 1021–1035.

Lee S and Hauert S (2023) Building trustworthiness by minimizing the sim-to-real gap in fault detection for robot swarms. *Proceedings of the First International Symposium on Trustworthy Autonomous Systems*. New York, NY: Association for Computing Machinery, 1–3. DOI: 10.1145/3597512.3597527.

Lefèvre S, Carvalho A and Borrelli F (2015a) A learning-based framework for velocity control in autonomous driving. *IEEE Transactions on Automation Science and Engineering* 13(1): 32–42.

Lefèvre S, Carvalho A, Gao Y, et al. (2015b) Driver models for personalised driving assistance. *Vehicle System Dynamics* 53(12): 1705–1720.

Li X and Parker LE (2007) Sensor analysis for fault detection in tightly-coupled multi-robot team tasks. *Proceedings 2007 IEEE International Conference on Robotics and Automation*. New York, NY: IEEE, 3269–3276. DOI: 10.1109/ROBOT.2007.363977.

Lin H, Zhai G and Antsaklis PJ (2003) Set-valued observer design for a class of uncertain linear systems with persistent disturbance and measurement noise. *International Journal of Control* 76(16): 1644–1653.

Mavrogiannis C and Knepper RA (2021) Hamiltonian coordination primitives for decentralized multiagent navigation. *The International Journal of Robotics Research* 40(10-11): 1234–1254. DOI: 10.1177/02783649211037731.

Millérioux G and Daafouz J (2007) Invertibility and flatness of switched linear discrete-time systems. In: A Bemporad, A Bicchi and G Buttazzo (eds) *Hybrid Systems: Computation and Control*. Berlin: Springer, 714–717.

Monteriu A, Asthan P, Valavanis K, et al. (2007) Model-based sensor fault detection and isolation system for unmanned ground vehicles: theoretical aspects (part I). *Proceedings 2007 IEEE International Conference on Robotics and Automation*. New York, NY: IEEE, 2736–2743.

Morbidi F, Colaneri P and Stanger T (2013) Decentralized optimal control of a car platoon with guaranteed string stability. *2013 European Control Conference (ECC)*. New York, NY: IEEE, 3494–3499.

Nguyen TW, Catoire L and Garone E (2019) Control of a quadrotor and a ground vehicle manipulating an object. *Automatica* 105: 384–390.

Olfati-Saber R, Fax JA and Murray RM (2007) Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE* 95(1): 215–233.

Özveren C and Willsky A (1992) Invertibility of discrete-event dynamic systems. *Mathematics of Control, Signals, and Systems* 5(4): 365–390.

Pallottino L, Scordio VG, Bicchi A, et al. (2007) Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics* 23(6): 1170–1183.

Parker L (1998) Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation* 14(2): 220–240. DOI: 10.1109/70.681242.

Pasqualetti F, Bicchi A and Bullo F (2012) Consensus computation in unreliable networks: a system theoretic approach. *IEEE Transactions on Automatic Control* 57(1): 90–104. DOI: 10.1109/TAC.2011.2158130.

Pasqualetti F, Dörfler F and Bullo F (2013) Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control* 58(11): 2715–2729.

Pedone S and Fagiolini A (2020) Racecar longitudinal control in unknown and highly-varying driving conditions. *IEEE Transactions on Vehicular Technology* 69(11): 12521–12535.

Petri R, Springer M, Zelle D, et al., 2016, November. Evaluation of lightweight TPMs for automotive software updates over the air. In *Proc. of 4th International Conference on Embedded Security in Car USA* (pp. 1-15).

Pettersson O (2005) Execution monitoring in robotics: a survey. *Robotics and Autonomous Systems* 53(2): 73–88.

Pfleeger CP, Pfleeger SL and Margulies J (2015) *Security in Computing*. Saddle River, NJ: Prentice Hall.

Pitropov M, Garcia DE, Rebello J, et al. (2021) Canadian adverse driving conditions dataset. *The International Journal of Robotics Research* 40(4-5): 681–690. DOI: 10.1177/0278364920979368.

Platzer A (2010) *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Berlin: Springer Science & Business Media.

Radwan N, Burgard W and Valada A (2020) Multimodal interaction-aware motion prediction for autonomous street crossing. *The International Journal of Robotics Research* 39(13): 1567–1598. DOI: 10.1177/0278364920961809.

Ramadge P and Wonham W (1987) Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization* 25(1): 206–230.

Ramadge P and Wonham W (1989) The control of discrete event systems. *Proceedings of the IEEE* 77(1): 81–98.

Raya M, Papadimitratos P and Hubaux JP (2006) Securing vehicular communications. *IEEE Wireless Communications* 13(5): 8–15.

Rungger M, Mazo JM and Tabuada P (2013) Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. *Proceedings of the 16th Conference on Hybrid Systems: Computation and Control*. New York, NY: Association for Computing Machinery, 333–342.

Sain M and Massey J (2002) Invertibility of linear time-invariant dynamical systems. *IEEE Transactions on Automatic Control* 14(2): 141–149.

Schneider R, Kohn A, Klimke M, et al. (2017) Cyber security in the automotive domain – an overview. In: *WCXTM 17: SAE World Congress Experience*. Pittsburgh, PA, SAE International, 1–10. DOI: 10.4271/2017-01-1652.

Serlin Z, Yang G, Sookraj B, et al. (2020) Distributed and consistent multi-image feature matching via quickmatch. *The International Journal of Robotics Research* 39(10-11): 1222–1238. DOI: 10.1177/0278364920917465.

Shamma JS and Tu KY (1999) Set-valued observers and optimal disturbance rejection. *IEEE Transactions on Automatic Control* 44(2): 253–264.

Shoham Y and Tennenholtz M (1995) On social laws for artificial agent societies: off-line design. *Artificial Intelligence* 73(1-2): 231–252.

Sinopoli B, Sharp C, Schenato L, et al. (2003) Distributed control applications within sensor networks. *Proceedings of the IEEE* 91(8): 1235–1246.

Solyom S and Coelingh E (2013) Performance limitations in vehicle platoon control. *IEEE Intelligent Transportation Systems Magazine* 5(4): 112–120.

Stavrou D, Eliades DG, Panayiotou CG, et al. (2016) Fault detection for service mobile robots using model-based method. *Autonomous Robots* 40: 383–394.

Strobel V, Castelló Ferrer E and Dorigo M (2018) Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 541–549.

Sumra IA, Hasbullah H, Lail J, et al. (2011) Trust and trusted computing in vanet. *Computer Science Journal* 1(1).

Sundaram S and Hadjicostis CN (2008) Partial state observers for linear systems with unknown inputs. *Automatica* 44(12): 3126–3132.

Tarapore D, Lima PU, Carneiro J, et al. (2015) To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspiration & Biomimetics* 10(1): 016014.

Tarapore D, Timmis J and Christensen AL (2019) Fault detection in a swarm of physical robots based on behavioral outlier detection. *IEEE Transactions on Robotics* 35(6): 1516–1522. DOI: 10.1109/TRO.2019.2929015.

Thrun S (2002) Probabilistic robotics. *Communications of the ACM* 45(3): 52–57.

Trinh MT and Nguyen VT (2023) An observer-based distributed nonlinear model predictive fault-tolerant control for leader-following formation tracking control. *2023 International Conference on System Science and Engineering (ICSSE)*. New York, NY: IEEE, 134–141.

Trumić M, Grioli G, Jovanović K, et al. (2022) Force/torque-sensorless joint stiffness estimation in articulated soft robots. *IEEE Robotics and Automation Letters* 7(3): 7036–7043.

Vu L and Liberzon D (2008) Invertibility of switched linear systems. *Automatica* 44(4): 949–958.

Wedaj S, Paul K and Ribeiro VJ (2019) Dads: decentralized attestation for device swarms. *ACM Transactions on Privacy and Security (TOPS)* 22(3): 1–29.

Wehbe R and Williams RK (2021a) Probabilistic resilience of dynamic multi-robot systems. *IEEE Robotics and Automation Letters* 6(2): 1777–1784. DOI: 10.1109/LRA.2021.3060378.

Wehbe R and Williams RK (2021b) Probabilistic security for multirobot systems. *IEEE Transactions on Robotics* 37(1): 146–165. DOI: 10.1109/TRO.2020.3014024.

Xu F, Puig V, Ocampo-Martinez C, et al. (2017) Set-valued observer-based active fault-tolerant model predictive control. *Optimal Control Applications and Methods* 38(5): 683–708.

Yoo T and Lafortune S (2002) Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control* 47(9): 1491–1495.

Zad S, Kwong R and Wonham W (2003) Fault diagnosis in discrete-event systems: framework and model reduction. *IEEE Transactions on Automatic Control* 48(7): 1199–1212.

Zattoni E, Perdon AM and Conte G (2017) Output regulation by error dynamic feedback in hybrid systems with periodic state jumps. *Automatica* 81: 322–334. DOI: 10.1016/j.automatica.2017.03.037. https://www.sciencedirect.com/science/article/pii/S0005109817301681.

Zeng X (2015) *Hybrid networked control for cyber-physical network systems with applications to interconnected power grids*. PhD Thesis, Texas Tech University, Lubbock, TX.

Zeng W and Chow MY (2014) Resilient distributed control in the presence of misbehaving agents in networked control systems. *IEEE Transactions on Cybernetics* 44(11): 2038–2049. DOI: 10.1109/TCYB.2014.2301434.

Zheng B, Deng P, Anguluri R, et al. (2016) Cross-layer codesign for secure cyber-physical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35(5): 699–711. DOI: 10.1109/TCAD.2016.2523937.

Zhou L and Tokekar P (2021) Multi-robot coordination and planning in uncertain and adversarial environments. *Current Robotics Reports* 2: 147–157.

Zhuo-Hua D, Zi-xing C and Jin-xia Y (2005) Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: a survey. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. New York, NY: IEEE, 3428–3433.

Appendix

A Index to multimedia extensions

Table 6. Index to multimedia extensions.

Exten.	Media type	Description
#1	Video	Experiments of misbehavior detection in self-driving warehouse forklifts

B Event estimation with incomplete time-varying visibility

This section demonstrates the expression of the detector map in (7) given in Th. 1. Together with the nondeterministic automaton Δ_i presented in Section 4, this result extends existing solutions (Cassandras and Lafortune, 2006) insofar as they show how, given a discrete event-based model, a discrete-state monitor can be efficiently obtained even for partial and time-varying visibility of events.

Towards this goal, given a topology check v_h along with the jth and mth components, $\widehat{s}_{i,j}$ and $\widehat{s}_{i,m}$, of the known encoder map $\widehat{s}_i$, the following three propositions:

Proposition 1. Events activating on a single component of s_i. *The smallest upper approximation of a detector condition* $c = s_{i,j}$ *is* $\widehat{c} = \widehat{s}_{i,j}\ v_{h,j} \oplus \neg v_{h,j}$.

Proof. The jth component $s_{i,j}$ of the encoder map is expanded, based on the visibility region V_h, as

$$s_{i,j}(q_i, I_i) = \widehat{s}_{i,j}\left(q_i, I_i^h\right) \oplus \left(\oplus_{q_\ell \in I_i \setminus V_h} \mathbf{1}_{\eta_{i,j}(q_i)}(q_\ell)\right) = \\ = \widehat{s}_{i,j}\left(q_i, I_i^h\right) \oplus p_{i,j}(q_i, I_i),$$

which is conveniently factorized as follows. If $p_{i,j} = 0$, the detector condition reduces to $c = \widehat{s}_{i,j}$, whereas if $p_{i,j} = 1$, it becomes $c = \widehat{s}_{i,j} \oplus 1 = 1$. This leads to the factorization $c = \widehat{s}_{i,j}\ \neg p_{i,j} \oplus 1\ p_{i,j}$. Now, if the kth topology is entirely visible by the monitor ($v_{i,j} = 1$), it holds $p_{i,j} = 0$ since $I_i \setminus V_h = \emptyset$, which implies $c = \widehat{s}_{i,j}$, whereas nothing can be said on the value of $p_{i,j}$ if $v_{i,j} = 0$. Then, using the topology check $v_{h,j}$, c is factorized as $c = \widehat{s}_{i,j}\ v_{i,j} \oplus \left(\widehat{s}_{i,j} \neg p_{i,j} \oplus p_{i,j}\right) \neg v_{i,j}$. Its smallest visibility-based upper approximation is then $\widehat{c} = \max_{p_{i,j} \in \mathbb{B}} c = \widehat{s}_{i,j} v_{i,j} \oplus A \neg v_{i,j}$, with $A = \max_{p_{i,j} \in \mathbb{B}} \left(\widehat{s}_{i,j} \neg p_{i,j} \oplus p_{i,j}\right) = \max\{\widehat{s}_{i,j}, 1\} = 1$, which proves the thesis.

Proposition 2. Events activating on a single negated component of s_i. *The smallest upper approximation of a detector condition* $c = \neg s_{i,\ell}$ *is* $\widehat{c} = \neg\widehat{s}_{i,j}$.

Proof. Analogously to Prop. 1, the detector condition c is expanded using the visibility region V_h as follows:

$$\begin{aligned}\neg s_{i,j}(q_i, I_i) &= \neg\left(\widehat{s}_{i,j}\left(q_i, I_i^h\right) \oplus p_{i,j}(q_i, I_i)\right) = \\ &= \neg\widehat{s}_{i,j}\left(q_i, I_i^h\right) \neg p_{i,j}(q_i, I_i),\end{aligned}$$

where De Morgan's law has been used. If $p_{i,j} = 0$, the detector condition reduces to $c = \neg\widehat{s}_{i,j}$, whereas if $p_{i,j} = 1$, it becomes $c = 0$. Then, c is factorized as $c = \neg\widehat{s}_{i,j}\ \neg p_{i,j} \oplus 0\ p_{i,j} = \neg\widehat{s}_{i,j} \neg p_{i,j}$. Now, if $v_{i,j} = 1$, $p_{i,j} = 0$ that implies $c = \neg\widehat{s}_{i,j}$, whereas nothing can be said on its value otherwise. Therefore, c is factorized using the topology check as $c = \neg\widehat{s}_{i,j}\ v_{i,j} \oplus \neg\widehat{s}_{i,j}\ \neg p_{i,j}\ \neg v_{i,j}$. Its visibility-based smallest upper approximation is then

$$\begin{aligned}\widehat{c} &= \neg\widehat{s}_{i,j} v_{i,j} \oplus \neg\widehat{s}_{i,j} \max_{p_{i,j} \in \mathbb{B}}\left(p_{i,j}\right)\ \neg v_{i,j} = \\ &= \neg\widehat{s}_{i,j} v_{i,j} \oplus \neg\widehat{s}_{i,j}\ \neg v_{i,j} = \\ &= \neg\widehat{s}_{i,j}\left(v_{i,j} \oplus \neg v_{i,j}\right) = \neg\widehat{s}_{i,j},\end{aligned}$$

which gives the thesis.

Proposition 3. Events activating on the jth and the negated mth components of s_i. *The smallest upper approximation of a detector condition* $c = s_{i,j} \neg s_{i,m}$ *is* $\widehat{c} = \left(\widehat{s}_{i,j} v_{h,j} \oplus \neg v_{h,j}\right) \neg\widehat{s}_{i,m}$.

Proof. Based on the hth visibility region V_h, the detector condition is written as

$$\begin{aligned}c &= \left(\widehat{s}_{i,j} \oplus p_{i,j}\right)\left(\neg\widehat{s}_{i,m}\ \neg p_{i,m}\right) = \\ &= \widehat{s}_{i,j} \neg\widehat{s}_{i,m} \neg p_{i,m} \oplus p_{i,j} \neg\widehat{s}_{i,m} \neg p_{i,m}.\end{aligned}$$

Enumerating all combinations of $p_{i,j}$ and $p_{i,m}$, c is factorized as $c = \left(\neg\widehat{s}_{i,m}\right) p_{i,j}\ \neg p_{i,m} \oplus \left(\widehat{s}_{i,j}\ \neg\widehat{s}_{i,m}\right)\ \neg\ p_{i,j} \neg p_{i,m}$. Moreover, based on the topology check (recall that $v_{i,j} = 1$ implies $p_{i,j} = 0$, and $v_{i,m} = 1$ implies $p_{i,m} = 0$), the expression is further factorized as

$$\begin{aligned}c = {} & A v_{i,j} v_{i,m} \oplus B v_{i,j} \neg v_{i,m} \oplus \\ & \oplus C \neg v_{i,j} v_{i,m} \oplus D \neg v_{i,j} \neg v_{i,m},\end{aligned}$$

with $A = \widehat{s}_{i,j} \neg\widehat{s}_{i,m}$, $B = \widehat{s}_{i,j} \neg\widehat{s}_{i,m} \neg p_{i,m}$, $C = \neg\widehat{s}_{i,m} p_{i,j} \oplus \left(\widehat{s}_{i,j} \neg \widehat{s}_{i,m}\right) \neg p_{i,j}$, $D = \neg\widehat{s}_{i,m}\ p_{i,j}\ \neg p_{i,m} \oplus \widehat{s}_{i,j}\ \neg\widehat{s}_{i,m}\ \neg p_{i,j}\ \neg p_{i,m}$. Its visibility-based smallest upper approximation is then

$$\begin{aligned}\widehat{c} = {} & \widehat{s}_{i,j} \neg\widehat{s}_{i,m}\left(v_{i,j} v_{i,m} \oplus v_{i,j} \neg v_{i,m}\right) \oplus \\ & \oplus \neg\widehat{s}_{i,m}\left(\neg v_{i,j} v_{i,m} \oplus \neg v_{i,j} \neg v_{i,m}\right) = \\ = {} & \widehat{s}_{i,j} \neg\widehat{s}_{i,m} v_{i,j} \oplus \neg\widehat{s}_{i,m} \neg v_{i,j},\end{aligned}$$

which immediately gives the thesis.

Furthermore, the above results are now helpful to prove Theorem 1. This is obtained as follows:

Proof. *(of Theorem 1)* The general case of a map e_i in (2) involves general encoder index sets $\gamma_{i,j}, \gamma_{i,j}^*$. One has to prove that the previous propositions hold even when these sets are not singletons, which can be obtained by induction on the number of their elements.

To begin with, suppose that $\gamma_{i,j} = \{1, \ldots, l\}$. The case with $l = 1$ is proved by Prop. 1. Assume now the thesis holds for $l = m$, i.e., that the smallest upper approximation of $c = \otimes_{\ell \in \gamma_{i,j}} s_{i,\ell} = \otimes_{\ell=1}^{m} s_{i,\ell}$ is $\widehat{c} = \left(\otimes_{\ell=1}^{m} \widehat{s}_{i,j}\ v_{i,\ell} \oplus \neg v_{i,\ell}\right)$. The inductive step requires proving the thesis for $l = m + 1$. Defining $z = \otimes_{\ell=1}^{m} s_{i,\ell}$, the detector condition $c = \otimes_{\ell=1}^{m+1} s_{i,\ell}$ is written as $z\ s_{i,m+1} = z\ \left(\widehat{s}_{i,m+1} \oplus p_{i,m+1}\right)$, that is factorized further as follows. If $p_{i,m+1} = 0$, the detector condition reduces to $c = z\ \widehat{s}_{i,m+1}$, while if $p_{i,m+1} = 1$, it becomes $c = z$, thus giving the expression $c = z\ \widehat{s}_{i,m+1}\ \neg p_{i,m+1} \oplus z\ p_{i,m+1}$. The detector condition is factorized using the topology check $v_{i,m+1}$ as follows. If $v_{i,m+1} = 1$, it holds $p_{i,m+1} = 0$ and $c = z\ \widehat{s}_{i,m+1}$, while if $v_{i,m+1} = 0$ nothing can be said on its value. This yields to $c = z\ A\ v_{i,m+1} \oplus z\ B\ \neg v_{i,m+1}$, with $A = \widehat{s}_{i,m+1}$, and $B = \widehat{s}_{i,m+1}\ \neg\ p_{i,m+1} \oplus p_{i,m+1}$. The smallest visibility-based upper approximation is then

$$\begin{aligned}\widehat{c} &= \max_{p_{i,1}, \ldots, p_{i,m+1} \in \mathbb{B}} c = \\ &= \left(\max_{p_{i,1}, \ldots, p_{i,m}} z\right) \cdot \\ &\quad \cdot\left(\max_{p_{i,m+1}}\left(A v_{i,m+1} + B \neg v_{i,m+1}\right)\right) = \\ &= \left(\otimes_{\ell=1}^{m} \widehat{s}_{i,j} v_{i,\ell} \oplus \neg v_{i,\ell}\right)\left(\widehat{s}_{i,m+1} v_{i,m+1} \oplus \neg v_{i,m+1}\right),\end{aligned}$$

which proves the thesis in the first considered case.

Now suppose that $\gamma_{i,j}^* = \{1, \ldots, l\}$ and proceed, as above, by induction. The case with $l = 1$ is proven in Prop. 2. Assume now that the thesis holds for $l = m$, i.e., that the smallest upper approximation of $c = \otimes_{\ell \in \gamma_{i,j}^*} \neg s_{i,\ell} = \otimes_{\ell=1}^{m} \neg s_{i,\ell}$ is $\widehat{c} = \otimes_{\ell=1}^{m} \neg\widehat{s}_{i,j}$. One has to prove it for $l = m + 1$. The detector condition

$$c = \otimes_{\ell=1}^{m+1} \neg s_{i,\ell} = z\ \neg\widehat{s}_{i,m+1}\ \neg p_{i,m+1}$$

is factorized as follows. If $p_{i,m+1} = 0$, the expression reduces to $c = z\ \neg\widehat{s}_{i,m+1}$, while, if $p_{i,m+1} = 1$, it becomes $c = 0$; this leads to $c = z\ \neg\widehat{s}_{i,m+1}\ \neg p_{i,m+1}$. Moreover, the detector condition is factorized further using the topology check $v_{i,m+1}$. In this respect, if $v_{i,m+1} = 1$, it holds $p_{i,m+1} = 0$ and $c = z\ \neg\widehat{s}_{i,m+1}$, while if $v_{i,m+1} = 0$ nothing can be said on its value. This yields

$$\begin{aligned}c &= z\ \neg\widehat{s}_{i,m+1} v_{i,m+1} \oplus z \neg\widehat{s}_{i,m+1}\ \neg p_{i,m+1}\ \neg v_{i,m+1} = \\ &= z\ \neg\widehat{s}_{i,m+1}\left(v_{i,m+1} \oplus \neg p_{i,m+1}\ \neg v_{i,m+1}\right).\end{aligned}$$

The smallest visibility-based upper approximation is then $\widehat{c} = (\otimes_{\ell=1}^{m} \neg s_{i,\ell})(\neg \widehat{s}_{i,m+1} C)$, with

$$C = \max_{p_{i,m+1}} (v_{i,m+1} \oplus \neg p_{i,m+1} \neg v_{i,m+1}) = \\ = \max\{v_{i,m+1}, 1\} = 1,$$

which proves the thesis in this second case.

Finally, the cases with $\gamma_{i,j}, \overset{*}{\gamma}_{i,j} \neq \emptyset$ and of cardinality greater than one are proven by a recursive application of Prop. 3, which gives the thesis in the general setting.

C Proof of set-valued consensus

The relevance of a result such as the one in Theorem 2 stems from the number and type of applications, ranging from the convergence analysis of iterative consensus protocols on constrained lattices (Gasparri et al., 2011), to network synchronisation (Di Paola et al., 2015), and to the distributed estimation of maps (Fagiolini et al., 2015). This section focuses on proving the convergence of the consensus protocol involved in the theorem to the global centralized estimate. This is achieved as follows:

Proof of Theorem 2. Prove first that the state of the hth node after κ steps is

$$X_h[\kappa] = \bowtie_{\ell \in w_h(\kappa)} X_\ell[0], \tag{21}$$

meaning that it is the output of merging all initial values of its κ-hop c-neighbors together. One can proceed by induction on the consensus step κ. It is immediate to verify that the property is true for $\kappa = 1$ since it holds $X_h[1] = \bowtie_{\ell \in w_h(1)} X_\ell[0]$. Now, assuming the property is true after κ steps, it is required to prove it after the $\kappa + 1$ steps. Based on the consensus iteration map in (19), it holds

$$X_h[\kappa + 1] = \bowtie_{\ell \in w_h(1)} J_\ell[\kappa], \tag{22}$$

where $J_\ell[\kappa]$ is the current state of the ℓth 1-hop c-neighbor of h; this value is $J_\ell[\kappa] = \bowtie_{j \in w_\ell(\kappa)} X_j[0]$ by the inductive hypothesis. Note that the order by which every state value is processed is irrelevant because of the associativity and commutativity of $\bowtie$. Note also that the occurrence of identical estimates can be simplified by its idempotency. Now, observe that (22) involves the consensus states of all nodes j that are κ-hop c-neighbors of any 1-hop c-neighbor of h, that is all $j \in w_\ell(\kappa)$ where $\ell \in w_h$ (1), whose union is, by definition, the $[\kappa + 1]$-hop c-neighbors of h, i.e., all $j \in w_h(\kappa + 1)$. Hence, (22) is rewritten as $X_h[\kappa + 1] = \bowtie_{\ell \in w_h(\kappa+1)} X_\ell[0]$, which proves (21).

Finally note that for $\kappa \geq \overline{\kappa} = \mathrm{diag}(\mathcal{G}_i)$, the multi-hop c-neighbors are the whole graph, since $\mathcal{G}_i$ is connected, and hence it holds $w_h(\kappa) = \mathcal{W}_i$. Accordingly, for all consensus step $\kappa \leq \overline{\kappa}$, it holds

$$X_h[\kappa] = \bowtie_{\ell \in w_h(\overline{\kappa})} X_\ell[0] = \bowtie_{\ell \in w_h(\overline{\kappa})} \eta_i^\ell = \overline{\eta}_i.$$

Since the above convergence occurs for all nodes $h \in \mathcal{W}_i$, the collective consensus state $(X_1, \ldots, X_m)$ converges to $\mathbf{1}_m \, \overline{\eta}_i$, which concludes the proof.

D Software guidelines

The simulator and code for the automatic generation of distributed monitors are available at https://github.com/federicomassa/IDS. The following are general guidelines for the specification of a protocol $\mathcal{P}$. The model components of all agents must be described in a set of files in the INPUT folder. The starting point of the protocol specification is a JSON file, located in the INPUT/Simulator folder, containing one *object* with at least the elements listed below:

```
1 {
2   “dynamic_models”: [...],
3   “control_models”: [...],
4   “sensors”: [...],
5   “agents”: [...],
6     ...
7 }
```

where dots indicate further information that is omitted here or described below. The *dynamic_models* item is an *array* of objects, each of which describes a specific dynamic model f_i. The generic format of each f_i item is exemplified below for the highway example:

```
1 {
2   “name”: “Unicycle”,
3   “state_variables”: [“x”, “y”, “theta”, “v”],
4   “control_variables”: [“a”, “omega”],
5   “dynamics”: “UnicycleKinematics”,
6     ...
7 }
```

where, again, dots are used to indicate terms that are omitted here for the sake of space. The “dynamics” item indicates the file, located in the INPUT/Dynamics folder, where the dynamics map of $\mathcal{A}_i$ is described.

The *control_models* item is an *array* of objects that represent pairs of *automaton* δ_i and *decoder map* u_i. Referring again to the motorway example, the generic format is as follows:

```
1 {
2   “name”: “LeftHandDrivingRules”,
3   “maneuvers”: [“FAST”, “SLOW”, “LEFT”, “RIGHT”],
4   “automaton”: “CarAutomaton”,
5   “control_variables”: [“a”, “omega”],
6   “decoder”: “AccOmegaControl”
7 }
```

where the “automaton” and “decoder” indicate the files, located in the INPUT/Automata and INPUT/Decoders folders, respectively, where the automaton and the decoders are described. By using the elements defined above, agents are then described as an array of objects, each of which has the format as in the following extract:

```
1 {
2    "id": "0",
3    "init_states": {"x":0, "y":1.5, "theta":0.5, "v":20},
4    "init_maneuver": "FAST",
5    "dynamic_model": "Unicycle",
6    "control_model": "LeftHandDrivingRules",
7    "sensors": ["ExternalCamera", "SelfSensor"],
8    "consensus": "ON",
9    "length": "4.47",
10    "desiredV": "20",
11    "parameters": ["length", "desiredV"]
12    "image": "Vehicles/Acura_NSX_red.png".
13 }
```

where "sensors" is an object that consists of an array of simulated devices measuring the ith continuous state as well as those of all neighboring vehicles according to the *visibility map* V. This is declared as:

```
1 {
2    "sensors":
3    [
4       {"name": "Camera", "type": "external"},
5       {"name": "IntSensor", "type": "internal"}
6    ]
7 }
```

The example of a 360° external camera acting as a visibility map V_i can be found in the INPUT/Sensors folder. Further instructions for the automatic generation and compilation of the distributed monitor can be found on the GitHub site.

E Key symbols

This section summarizes the key symbols used to describe the main object of the work: Table 7 reports the ones used to describe socially organized robots, while Tables 8 and 9 those involved in the local monitor and consensus node.

Table 7. Key symbols used to describe socially organized robots.

Sym.	Name/Description
$\mathcal{Q}$	*environment*/state-space shared by all robots
n	number of robotic agents in the system
$\mathcal{A}_i$	*ith robot or agent*/a generic robotic agent
$\mathcal{P}$	*ith cooperation protocol*/octuple of components specifying the behavior of a robot $\mathcal{A}_i$ and its interactions with neighbors
q_i	*ith continuous state*/vector describing the continuous state of $\mathcal{A}_i$
σ_i	*ith discrete state*/scalar representing the discrete state of $\mathcal{A}_i$, that is the action or maneuver that the robot is performing
T_i	*ith topology set*/set of topologies characterizing all regions around $\mathcal{A}_i$ that are relevant for the protocol $\mathcal{P}_i$
$\eta_{i,j}$	(i,j)*th mobile topology*/a region nearby $\mathcal{A}_i$ where the presence or absence of other robots, named $\mathcal{A}_i$-neighbors, affects the behavior of $\mathcal{A}_i$
$\eta^*_{i,j}$	(i,j)*th constant topology*/constant region used to delimit the environment or represent obstacles
$\mathcal{N}_i$	*ith proximity space or neighborhood*/region around $\mathcal{A}_i$ that comprises all mobile topologies of $\mathcal{A}_i$
N_i	*ith neighbor set*/set of neighboring agents, $\mathcal{A}_i$-neighbors, that lay within the ith neighborhood $\mathcal{N}_i$
I_i	*ith input set*/set of configurations of all $\mathcal{A}_i$-neighbors
n_i	number of $\mathcal{A}_i$-neighbors
V_i	*ith visibility map*/map describing the region where the sensors of $\mathcal{A}_i$ can measure the continuous states of other agents
s_i	*ith encoder map*/map encoding the presence/absence of $\mathcal{A}_i$-neighbors in every topology
$s_{i,j}$	(i, j)*th encoder component*/component of the encoder map s_i associated with the (i,j)th topology
E_i	*ith event alphabet*/finite set of events that trigger the update of the current ith discrete state
$e^{i,j}$	(i, j)*th event*/specific event relevant for protocol $\mathcal{P}_i$
Γ_i	*ith encoder index set*/set of indices describing which and how the components of the encoder map determine the detection of events
$\gamma_{i,j}$, $\gamma^*_{i,j}$	*encoder index sets*/set of indices describing which encoder components $s_{i,j}$ are used to detect the occurrence of $e^{i,j}$, with positive or negated values
Σ_i	*ith discrete state set*/finite set of discrete states for the automaton of $\mathcal{A}_i$
δ_i	*ith automaton map*/deterministic automaton mapping discrete states and detected events to the next discrete states
u_i	*ith decoder map*/map determining control actions based on current continuous and discrete states and the input set
f_i	*ith dynamics map*/map describing the instantaneous motion direction of the continuous state q_i
t	*continuous time*
k	*discrete time*/discrete index at which an event is recognized by the ith detector map
$\mathcal{U}_i$	*ith control action set*/set of permissible control actions for $\mathcal{A}_i$
$\mathcal{H}_i$	*ith hybrid dynamic map*/map describing how the complete state of $\mathcal{A}_i$ is dynamically updated
$\phi_{\mathcal{H}_i}$	*solution*/explicit solution of the cooperative model describing the behavior of $\mathcal{A}_i$

Table 8. Key symbols involved in the local monitor.

Sym.	Name/Description
$\mathcal{M}_h$	*hth monitor*/monitoring process onboard a generic agent $\mathcal{A}_h$ that tries to learn whether or not the behavior of $\mathcal{A}_i$ is cooperative
ϵ	*mismatch tolerance*/tunable parameter describing the maximum tolerance when comparing two behaviors
I_i^h	*ith known input set*/subset of I_i that is known to $\mathcal{M}_h$
v_i	*ith topology check*/map describing in which topologies the *h*th monitor $\mathcal{M}_h$ has full or partial visibility
$\widehat{s}_i$	*ith known encoder*/map providing an a priori lower estimate of s_i
Δ_i	*ith nondeterministic automaton*/map describing the set of possible next discrete states based on the current set of discrete state and detected events
$\widehat{I}_i^h$	*extended input set map*/set of predicted input sets that include I_i^h and continuous states in the non-visible portion of the *i*th neighborhood $\mathcal{N}_i \backslash V_h$
$\mathcal{L}_i$	*ith forward link table*/table of triples used to "invert" the hybrid model $\mathcal{H}_i$
$\bowtie_\epsilon$	*similarity check*/map comparing measured and predicted behaviors and returning the ones that are ϵ-similar
s_i^h	*ith refined encoder map*/map computing the a-posteriori largest estimate of s_i
V_i^h	*ith refined visibility*/map computing the a-posteriori visibility over each topology
η_i^h	*ith occupancy estimator map*/a-posteriori estimates of the occupancy of each (*i*,*j*)th topology, and also the first main output of monitor $\mathcal{M}_h$
$\mathcal{C}_i$	*ith classifier*/map classifying the behavior of $\mathcal{A}_i$ as certainly cooperative, possibly-cooperative, or uncooperative, and also the second main output of monitor $\mathcal{M}_h$

Table 9. Key symbols involved in the consensus node.

Sym.	Name/Description
m	total number of local monitors
κ	*consensus step*/current index of the consensus step
χ	*consensus tolerance*/tunable parameters describing the maximum tolerance between the values estimated by difference monitors
Lift	*lifting operator*/map used to enlarge/lift each value estimated by a local monitor by a quantity χ
$\bowtie$	*consensus operator*/binary map merging two occupancy estimates
$I_{i,j}^h$	known input set available for $\mathcal{M}_h$ and restricted to the (*i*,*j*)th topology
$W_{i,j}^h$	known portion of the (*i*,*j*)th topology
$\overline{W}_{i,j}^h$	non-visible portion of the (*i*,*j*)th topology where monitor $\mathcal{M}_h$ has inferred the existence or absence of $\mathcal{A}_i$-neighbors
s^j	binary flag indicating if the region $\overline{W}_{i,j}^h$ must contain or not $\mathcal{A}_i$-neighbors
$\mathcal{G}_i$	*ith communication graph*/graph describing the current topology of communication among all consensus nodes trying to reach a socially agreed decision on $\mathcal{A}_i$
$\overline{\eta}_i$	global estimate of the occupancy
X	*consensus collective state*/vector state including all consensus nodes state
X_h	state of the *h*th consensus node
$\overline{\kappa}$	consensus convergence time, i.e., maximum number of consensus steps before all nodes agree on the value $\overline{\eta}_i$
$w_h(p)$	*p-hop c-neighbors*/set of nodes that are within p hops from the *h*th monitor/nodes