Distributed Motion Misbehavior Detection in Teams of Heterogeneous Aerial Robots

Simone Martini*, Davide Di Baccio*, Francisco Alarcón Romero†,
Antidio Viguria Jiménez†, Lucia Pallottino*‡ Gianluca Dini*‡ and Aníbal Ollero†§

*Research Center "E. Piaggio," University of Pisa, Pisa, Italy
†Fundación Andaluza para el Desarrollo Aeroespacial (FADA-CATEC), Seville, Spain
‡Dept. of Ingegneria dell'Informazione, University of Pisa, Pisa, Italy
§Dept. of Ingenierìa de Sistemas y Automàtica, University of Seville, Seville, Spain

Abstract

This paper addresses the problem of detecting possible misbehavior in a group of autonomous mobile robots, which coexist in a shared environment and interact with each other and coordinate according to a set of common cooperation rules. Such rules specify what actions each robot is allowed to perform while achieving its individual goals. The rules are distributed, i.e., they can be evaluated based only on the knowledge of the individual robot, and on information that can be gathered through communication with neighbors. We consider misbehaving robots as those robots whose behaviors deviate from the nominal assigned one, i.e., do not follow the rules, because of either spontaneous failures or malicious tampering. The main contribution of the paper is to provide a methodology to detect such misbehaviours by observing the congruence of actual behavior with the assigned rules as applied to the actual state of the system. The presented methodology is based on a consensus protocol on the events observed by robots. The methodology is fully distributed in the sense that it can be performed by individual robots based only on the local available information, it has been theoretically proven and validated with experiments involving real aerial heterogeneous robots.

I. Introduction

The availability of distributed systems gave rise in the late 80s to a profound rethinking of many decision making problems and enabled solutions that were impossible before. A similar trend is now happening in control and will soon enable a formidable number of new robotic applications. Various distributed control policies have been proposed for formation control, flocking, sensor coverage, and intelligent transportation (see e.g. [1], [2], [3]). The adoption of similar notions of decentralization and heterogeneity in Robotics is advantageous in many tasks, where a cooperation among agents with analogous or complementary capabilities is necessary to achieve a shared goal. More specifically, we are interested in distributed multi-agent systems where each agent is assigned with a possibly different private goal, but needs to coordinate its actions with other neighboring agents. The flexibility and robustness of such distributed systems, and indeed their ability to solve complex problems, have motivated many works that have been presented in literature (see e.g., [4], [5], [6], [7], [8], [9]). Although in most cases agents are modeled as identical *copies* of the same prototype, this assumption is often restrictive as the different agents that forms a society may be implemented by different makers, and with different technologies etc. heterogeneity in these artificial systems are advantageous when e.g. a problem requires interaction of agents with similar skills as well as agents with complementary capabilities. Most important, heterogeneity may be introduced to model the existence of malfunctioning agents, also called *intruders* [10], [11]. The complexity needed to represent such behaviors can be successfully captured by hybrid models, in which a continuous–time dynamics describes the physical motion of each agent, while an event–based one describes the sequence

Gianluca Dini is the corresponding author (g.dini@iet.unipi.it).

of interactions with its neighbors. This paper addresses the problem of detecting possible misbehavior in a group of autonomous robots, which coexist in a shared environment and interact with each other while coordinating according to a set of common distributed rules. The objective is to provide robots the capability of detecting agents whose behavior deviates from the assigned one, due to spontaneous failures or malicious tampering. The objective is ambitious and indeed very difficult to be achieved without a-priori knowledge of the interaction rules, but a viable solution can be found if the hybrid models describing the behavior is known in advance. The proposed methodology is fully distributed in the sense that it can be performed by individual robots based only on the local available information. It is based on a two–step process: first, agents combines the information gathered from on–board sensors and from neighbors by using communication and compute an a–priori prediction of the set of possible trajectories that the observed agent should execute based on the cooperative rules (*prediction phase*); then, the predicted trajectories are compared against the one actually executed and measured by the aircraft itself and if none result close enough, the observed aircraft is selected as uncooperative (*verification phase*). The motion misbehavior detection ability of a single local monitor (used in the verification phase) is limited by its partial visibility. Robots need hence to combine the locally available information and reach an agreement on the reputation of the observed robot. To do this, we propose a Boolean consensus protocol that differs from those provided in [12], [13], [14]. Indeed, in this paper a consensus protocol on the events (more precisely on the encoder map defined in the following) observed by robots is proposed. On the contrary in the other works the consensus was on the reconstruction on the surrounding area of the observed robot [10]. In other words, we use the consensus to reconstruct the possible presence of a robot in an area that is not visible from all observing robots while in the other approaches a consensus protocol was used to reconstruct the robot position in the area. Hence, in our approach the computational cost is limited.

Although the proposed method is general and can be applied to a wide range of applications, it has been tested with experiments involving real aerial robots where the problem of detecting an intruder is fundamental for the safety of the system.

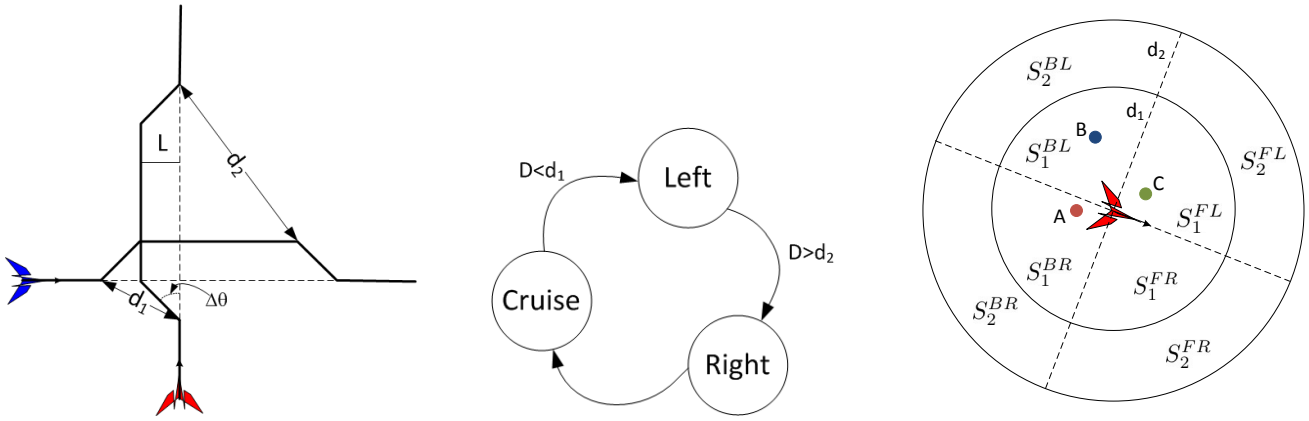
The paper is organized as follows. We star introducing in Section II a case study example to help the reader following the notation introduced in Section III where the hybrid model of the proposed cooperation protocol is reported. The misbehave detection problem is formally defined in Section IV. The Boolean consensus misbehaviour detection strategy is described in Section V where the convergence in a finite number of steps is formally proved. Finally, in Section VI experimental setup is described and experiments' results are reported.

II. A Case Study Example

In order to introduce the formal definitions and concepts of the paper we first start introducing a case study example that will be used to give an intuitive idea of the formalism introduced in next sections. The example is a simplified version of the collision avoidance strategy proposed in [15] and proved to be safe for two aircraft. Simplifications are introduced to make the concepts introduced later simpler, hence the example is not intended to be a realistic description of UAV scenarios.

Example 1: Consider two identical aircraft cruising at a given altitude with constant and equal linear velocity v. Aircraft can be represented by vector $(x,\, y,\, \theta) \in \mathbb{R}^2 \times S^1$. Referring to Fig. 1(a), each aircraft flights straight in *Cruise* mode until the other aircraft is detected at distance closer than d_1. Whenever it occurs, it changes instantaneously its heading angle of amount $\Delta\theta$ and proceeds straight until a distance L from nominal trajectory is reached then it changes its heading of amount $-\Delta\theta$ and proceed straight (*Left* mode). As soon as the other aircraft is at distance larger than d_2 (where $d_1 < d_2$) the aircraft changes instantaneously its heading angle of amount $-\Delta\theta$ and proceeds straight until nominal trajectory is reached then it changes its heading of amount $\Delta\theta$ (*Right* mode) and then switch to the *Cruise* mode.

Let $D(t)$ be the distance between the two aircraft at time t, the behaviour of each aircraft is reported in Fig. 1(a) with a graphical representation of the associated hybrid model in terms of operating modes and switching conditions, see Fig. 1(b).



(a) Evolution of aircraft based on the collision avoidance policy

(b) Hybrid system associated to the collision avoidance policy

(c) Sectors associated to each aircraft

Fig. 1. Simplified version of collision avoidance strategy proposed in [15] reported in Example 1.

To describe the motion of aircraft based on those rules we may consider a configuration vector $(x,\, y) \in \mathbb{R}^2$ and the control input $\theta \in \{0,\, \pm\Delta\theta\}$ with kinematics equations

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \end{cases} \tag{1}$$

To apply the described collision avoidance policy, each aircraft must be able to recognize the presence of another aircraft in a *detection disc* centered in its position and of radius d_2. We then consider aircraft with limited sensing that are able to detect the presence of other aircraft in such a detection disc. The detection disc is then subdivided in eight sectors based on the orientation of the vehicle and the radii d_1 and d_2, see Fig. 1(c), i.e. the front left (S_1^{FL}, S_2^{FL}), front right (S_1^{FR}, S_2^{FR}), back left (S_1^{BL}, S_2^{BL}) and back right (S_1^{BR}, S_2^{BR}) sectors. In order to monitor the behaviour of target aircraft h, we suppose that an observing aircraft i, that lays in one of the sectors of h, is able to detect the presence of a third aircraft k that is at distance less than d_2 and that lays in one of the visible sectors. For example, if aircraft i lays in sector S_1^{BR} of aircraft h, it can detect aircraft k only if it is at distance less than d_2 and inside sectors S_1^{BL}, S_2^{BL}, S_1^{FR}, S_2^{FR}, S_1^{BR} and S_2^{BR} but not in S_1^{FL} or S_2^{FL}. For example, referring to Fig. 1(c), an aircraft i in A can detect the aircraft k in B but not the one in C.

In case of a larger number of aircraft there exist several collision avoidance policies that have been proved to guarantee safety of the system but are far too complex to be used as a simple illustrative example, see e.g. the round–about policies in [16], [17].

III. A Model of Cooperation Protocols for Robotics Agents

Toward our goal of designing a distributed motion misbehavior detection system that applies to very general, heterogeneous robots, it is necessary to introduce a formalism that allows to model uniformly a large variety of possible robots sharing sets of common rules.

Consider n robotic agents $\mathcal{A}_1, \ldots, \mathcal{A}_n$, where $\mathcal{A}_i$ is described by a vector $\mathbf{q}_i$ in a continuous configuration space $\mathcal{Q}$. Such agents have their own dynamics, but need to collaborate with each other in order to accomplish a common task or to achieve possibly conflict goals. We consider systems where agents' interaction can be described by rules that are *decentralized* and *event–based*, i.e. the cooperation actions that every agent can perform are specified according to a shared set $R \stackrel{\text{def}}{=} \{\mathsf{rule}_1, \ldots, \mathsf{rule}_m\}$ of rules based only on locally measured events. To give an example, agents can be vehicles or robot moving in a shared environment and following common driving rules so as to avoid collisions [18], [19] as also described in the case study in previous Section. Each vehicle determines its current maneuver based on the presence or absence of other neighboring vehicles and on its own destination. To model such cooperating networked

and distributed systems in the general case, we adopt a simplified version of the formalism introduced in [20], according to which an agent $\mathcal{A}_i$ is specified by:

- A configuration vector $\mathbf{q}_i \in \mathcal{Q}$, where $\mathcal{Q}$ is a configuration space ($\mathbf{q} = (x,\, y) \in \mathbb{R}^2$ in the case study Example 1).
- An input vector $\mathbf{u}_i \in \mathcal{U}$, where $\mathcal{U}$ is a set of admissible input values; ($\mathcal{U} = \{0,\, \pm\Delta\theta\}$ in the case study).
- A discrete state $\sigma_i \in \Sigma$, where Σ is the set of operating modes; ($\Sigma_i = \{\textit{Cruise},\, \textit{Left},\, \textit{Right}\}$ in the case study).
- A dynamic map f_i describing how the agent's configuration is updated:

$$\dot{\mathbf{q}}_i(t) = f_i(\mathbf{q}_i(t), \mathbf{u}_i(t)) \tag{2}$$

  (for Example 1 the dynamic map is reported in (1)).
- A decoder map $\mathcal{G}_i$ describing which control values are applied in different operating modes σ_i, i.e.

$$\mathbf{u}_i(t) = \mathcal{G}_i(\mathbf{q}_i(t), \sigma_i(t_k))\,, \quad \text{for } t \in [t_k, t_{k+1})\,.$$

  In Example 1 we have $w(t) = \mathcal{G}_i(q_i(t), \textit{Cruise}) = 0$, while $w(t) = \mathcal{G}_i(\mathbf{q}_i(t), \textit{Left})$ is a sequence of $\Delta\theta$, 0, $-\Delta\theta$ and 0 again. Finally, $w(t) = \mathcal{G}_i(\mathbf{q}_i(t), \textit{Right})$ is a sequence of $-\Delta\theta$, 0, $\Delta\theta$ and 0 again.
- A set of *topologies* $\eta_{i,1}(\mathbf{q}), \ldots, \eta_{i,\kappa_i}(\mathbf{q})$ on $\mathcal{Q}$, whose union defines the agent's neighborhood in $\mathbf{q}$, i.e. $N(\mathbf{q}_i) = \cup_{j=1}^{\kappa_i} \eta_{i,j}(\mathbf{q}_i)$. The set of neighbouring agents is hence $N_i = \{\mathcal{A}_k | \mathbf{q}_k \in N(\mathbf{q}_i)\}$, while the set of neighbors' configurations is $I_i = \{\mathbf{q}_k \in \mathcal{Q} \,|\, \mathcal{A}_k \in N_i\}$, referred in the following as *influence set* of $\mathcal{A}_i$.
  Referring to the case study described in Example 1, for aircraft i in $\mathbf{q}$ we have eight topologies corresponding to the eight sectors of the detection disc centered in $\mathbf{q}$. The neighbouring agents are aircraft in the detection disc, i.e. aircraft that are closer to i more than d_2.
- An event vector $\mathbf{s}_i \in \mathbb{B}^{\kappa_i}$ (whose components later will be referred to as *sub-events*) and a detection map $\mathcal{S}_i$ involving conditions over $\mathcal{Q}$ (as e.g. the presence of another agent in a specific region):

$$s_{i,j}(t) = \sum_{\mathbf{q}_k \in I_i} \mathbf{1}_{\eta_{i,j}(\mathbf{q}_i)}(\mathbf{q}_k)$$

  where $\sum$ represents the logical sum (*or*), and $\mathbf{1}_A(x)$ is the Indicator function of a set A. Events for the case study are the presence of aircraft closer than d_1 and the absence of aircraft at distance less than d_2. In general events can be compositions of sub–events and different events may depend on the same sub–events. Hence, a vector of sub–events s_i is considered.
- A static decision map or *encoder* φ_i indicating the detector condition $\mathbf{c}_i$ based on events vector $\mathbf{s}_i$:

$$\mathbf{c}_i(t_k) = \varphi_i(s_{i,1}(t_k), \ldots, s_{i,\kappa_i}(t_k))\,;$$

  in other words, $\mathbf{c}_i(t_k)$ is a vector of logic operations of sub–events $s_{i,j}$.
- An automaton δ_i describing how the agent's current discrete state (or mode of operation) σ_i is updated based on the detector condition $\mathbf{c}_i$:

$$\sigma_i(t_{k+1}) = \delta_i(\sigma_i(t), \mathbf{c}_i(t_k))\,. \tag{3}$$

Those two last concepts applied to the case study are reported in Fig. 1(b).

It is clear that the set of rules describes the set of p *operating modes*, $\Sigma \stackrel{\text{def}}{=} \{\mathsf{mode}_1, \ldots, \mathsf{mode}_p\}$, and the set of ν logical propositions, or *events*, $E \stackrel{\text{def}}{=} \{\mathsf{event}_1, \ldots, \mathsf{event}_\nu\}$. The occurrence of any of these events require the current mode σ_i of the generic agent $\mathcal{A}_i$ to be changed. The generic event event_l measured from $\mathcal{A}_i$ can be assigned with a logical variable $c_{i,l} \in \mathbb{B}$ taking the value true if event_l has been recognized by $\mathcal{A}_i$ and false otherwise. Although event_l depends on Q, that continuously evolves with the time t, it only switches from true to false or vice–versa at particular times t_k, with $k \in \mathbb{N}$, when the agents's mode σ_i must be updated. Hence, the *cooperation manager* can be seen as a Discrete Event

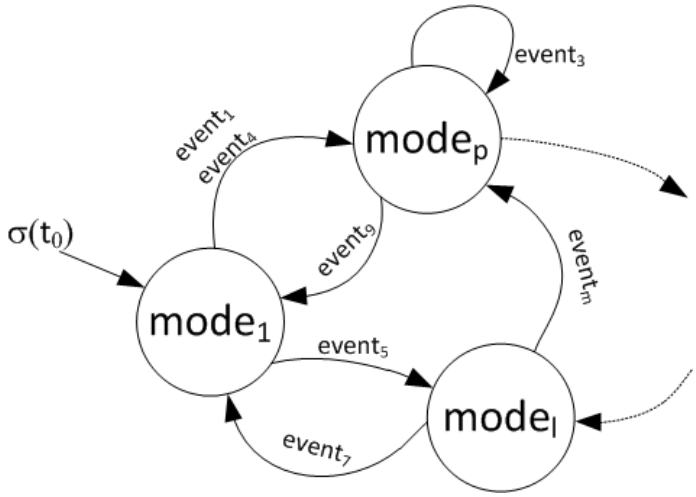


Fig. 2. Representation of dynamics δ of the automaton of the generic cooperation manager.

System (DES) [21], and indeed an *automaton* (see Fig. 2), that receives $\mathbf{c}_l$ as input and updates its state σ_i according to rules of the forms

- $\mathsf{rule}_0 \stackrel{\text{def}}{=} (\sigma(t_0) = \mathsf{mode}_1)$; $\leftarrow$ start in mode_1;
- $\mathsf{rule}_j \stackrel{\text{def}}{=}$ (**if** $\sigma_i(t_k) = \mathsf{mode}_l$ **and** $c_m(t_k) = \mathsf{true}$ **then** $\sigma_i(t_{k+1}) = \mathsf{mode}_p$).

Referring to the case study, mode_1 is *Cruise* and one of the rules is (if $\sigma_i(t_k) = \mathsf{Cruise}$ and $c_l(t_k) =$"aircraft detected at distance less than d_1" $= \mathsf{true}$ then $\sigma_i(t_{k+1}) = \mathsf{Left}$).

The *decoder*, connects the output of the the event–based system in (3) with the input of the time–driven system in (2). It translates or decodes the current maneuver $\sigma_i(t_k)$ into a control law, typically involving a feedback of the agent's configuration of the form

$$\mathbf{u}_i(t) = \mathcal{G}(\mathbf{q}_i(t), \sigma_i(t_k))\,, \tag{4}$$

so that the autonomous controlled system

$$\dot{\mathbf{q}}_i(t) = f_i(\mathbf{q}_i(t), \mathcal{G}(\mathbf{q}_i(t), \sigma_i(t_k)) = \tilde{f}_i(\mathbf{q}_i(t), \sigma_i(t_k))\,,$$

correctly performs the control planned for the mode $\sigma_i(t_k)$. The decoder is an application $\mathcal{G} : \mathcal{Q} \times \Sigma \to \mathcal{U}$ that returns the actuators' input during the interval $[t_k, t_{k+1})$ (up to next event). In this perspective, $\mathcal{G}$ acts has a converter from a discrete–valued event–driven signal to a continuous– valued time–driven one. A second block, the *encoder*, realizes the reverse connection: it evaluates the logical variables $c_{i,l}$, for $l = 1, \ldots, \nu$, from current value of the system configuration Q. However, in decentralized scenarios as the ones we are considering, every agent $\mathcal{A}_i$ must be able to plan its motion based on its own configuration $\mathbf{q}_i$ and the configurations of only the agents that lay in its vicinity. Hence, the encoder output will only depend on the influence set I_i of agent that instantaneously affects the behavior of $\mathcal{A}_i$.

Due to limited visibility of its sensors, an agent $\mathcal{A}_i$ is able to measure the configuration $\mathbf{q}_j$ of another agent $\mathcal{A}_j$ laying in its *visible region* $\mathcal{V}_i$. This region changes with time depending on the configurations of $\mathcal{A}_i$ and its neighbors, i.e. $\mathcal{V}_i(t) = \mathcal{V}(\mathbf{q}_i(t), Q(t))$. The remaining part of the configuration space, namely $\bar{\mathcal{V}}_i(t) = \mathcal{Q} \setminus \mathcal{V}_i(t)$, is the non–visible region and is composed of all those configurations that can not be "seen" from $\mathcal{A}_i$. Note that our problem requires the knowledge of also the state (σ_j) of an agent's cooperation manager, that is unmeasurable and thus will be estimated. For simplicity, we assume that $I_i \subseteq \mathcal{V}_i$, i.e. every agent is able to directly measure all information needed for planning its motion, as otherwise data received from possibly deceiving neighbors must be further validated. As a whole, the evolution of the continuous–valued time–driven dynamics of the physical system and that of the discrete–valued event–driven dynamics of the cooperation manager are entangled as it happens in a hybrid system

$\mathcal{H} = (\mathbf{q}_i, \sigma_i, I_i)$. The behavior of $\mathcal{A}_i$ can be written more compactly as

$$\begin{cases} (\dot{\mathbf{q}}_i(t), \sigma_i(t_{k+1})) = \mathcal{H}_i(\mathbf{q}_i(t), \sigma_i(t_k), I_i(t)) \,, \\ (\mathbf{q}_i(0), \sigma_i(0)) = (\mathbf{q}_i^0, \sigma_i^0) \,, \end{cases} \tag{5}$$

where $\mathcal{H}_i : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to T_\mathcal{Q} \times \Sigma_i$ is the agent's *hybrid dynamic map* [22] and T_Q is the tangent space of Q. We will denote with $\phi_{\mathcal{H}_i}(\mathbf{q}_i(t), \sigma_i(t), I_i(t))$ the evolution of system (5).

IV. Misbehaviours and Local Detection

Since our goal is to detect misbehaving agents, we first need to define how a misbehavior may manifest i.e. how a behavior may deviate from the nominal one. The first assumption is that an agent $\mathcal{A}_h$ may execute trajectories $\mathbf{q}_h(t)$ that do not comply with the rules of cooperation, but the information it exchanges with its neighbors is always correct. This can be guaranteed by the use of emerging trusted computing platforms (see e.g. [23], [24], [25]). Secondly, we consider the fact that the cooperation manager of a robot is implemented as a control task that runs periodically and that is scheduled every T seconds. This means that a mode σ is started at the generic discrete time $t_k \stackrel{\text{def}}{=} kT$ and is run up to t_{k+1}. Then, we assume that the local monitor of each robot and all the robots cooperation manager are *synchronized*, which can be obtained by means of e.g. the distributed solution proposed in [26]. The section presents the architecture of a *monitor* that can detect such misbehaviours, by using only information available to the agent. For the sake of clarity we refer to the robot $\mathcal{A}_i$ with an on–board monitor as the *observer robot* and to $\mathcal{A}_h$ as the *target robot*.

To begin with, consider that, if I_h is completely visible from $\mathcal{A}_i$, it is sufficient to *verify* that the trajectory $\bar{\mathbf{q}}_h(t)$ measured by the observer robot is close enough to the evolution of the cooperative model $\mathcal{H}$, i.e.

$$\|\bar{\mathbf{q}}_h(t) - \pi_\mathcal{Q}(\phi_{\mathcal{H}_i}(\bar{\mathbf{q}}_i(t_k), \bar{\sigma}_i(t_k), I_i(t)))\| \leq \epsilon, \quad \forall t,$$

where $\|\cdot\|$ is the Hausdorff distance, $\pi_\mathcal{Q}$ is the projector over the set $\mathcal{Q}$, and ϵ is an *accuracy* based on the quality of available sensors.

Nonetheless, it typically holds that $I_h \not\subseteq \mathcal{V}_i$ which makes it impossible to directly apply this simple solution. In other words, it may occur that a robot that influences the behaviour of $\mathcal{A}_h$ is not visible by $\mathcal{A}_i$, For example, referring to Fig. 1(c) B is visible by A while C is not. Hence, the influence region is partitioned as

$$\begin{aligned} I_h &= I_h \cap \mathcal{Q} = I_h \cap (\mathcal{V}_i \cup \bar{\mathcal{V}}_i) = \\ &= (I_h \cap \mathcal{V}_i) \cup (I_h \cap \bar{\mathcal{V}}_i) \stackrel{\text{def}}{=} I_h^{obs} \cup I_h^{unobs} \,. \end{aligned}$$

Reorder the model's inputs as $I_h = (I_h^{obs}, I_h^{unobs})$, where $I_h^{obs} \stackrel{\text{def}}{=} q_{i,1}, \ldots, q_{i,v_i}$ is the list of configurations known to $\mathcal{A}_i$, and $I_h^{unobs} \stackrel{\text{def}}{=} q_{i,v_i+1}, \ldots, q_{i,n_h}$ is the list of remaining configurations that are unknown to i.

Misbehavior of agent $\mathcal{A}_h$ during the period $[t_k, t_{k+1})$ can be found by solving the following

Problem 1 (Decentralized Intrusion Detection): Given a trajectory $\bar{\mathbf{q}}_h(t)$, a list $I_h^{obs}(t_k)$ of known configurations, a visibility region $\mathcal{V}_i$, and a desired accuracy ϵ, determine if there exists an input $\hat{I}_h(t_k) = (I_h^{obs}(t_k), \hat{I}_h^{unobs}(t_k))$ s.t.

$$\|\bar{\mathbf{q}}_h(t) - \pi_\mathcal{Q}(\phi_{\mathcal{H}_i}(\bar{\mathbf{q}}_i(t_k), \bar{\sigma}_i(t_k), \hat{I_i(t)}))\| \leq \epsilon, \quad \forall t \in [t_k, t_{k+1}) \,.$$

where $\hat{I}_h$ represents an "unobservable explanation" whose notion was introduced in DES [27] and used in [28], [29]. In other words, the problem is to determine if, given the available information, there exist unobserved conditions that influence the target robot and justify its motion based on the predefined rules.

A. Construction of the Local Monitor

The proposed approach is a two–step process: first, $\mathcal{A}_i$ computes an a–priori prediction of the set of possible trajectories that $\mathcal{A}_h$ can execute based on the cooperative model $\mathcal{H}_h$ and the partially known influence region (*prediction phase*); then, the predicted trajectories are compared against the one actually executed by $\mathcal{A}_h$ and measured by $\mathcal{A}_i$ and if none of them result close enough, $\mathcal{A}_h$ is selected as uncooperative (*verification phase*).

The prediction phase involves constructing a predictor $\tilde{\mathcal{H}}_h$ that encodes all the observer's uncertainty. The model is composed of a nondeterministic automaton whose state $\tilde{\sigma}_h \in \Sigma_h$ represents the set of operating modes that $\mathcal{A}_h$ can perform based on local information, and whose transitions $\tilde{\delta}$ are the same as in δ. The main challenge in the construction of the automaton is the estimation of an upper approximation $\tilde{\mathbf{c}}_i$ of each detector condition $\mathbf{c}_h$, that is achieved through the results that can be found in [30] and that are omitted for the sake of space. We hence suppose that each monitor is able to construct a predictor $\tilde{\mathcal{H}}_h$.

V. Consensus for Misbehavior Detection

The motion misbehavior detection ability of a single local monitor is limited by its partial visibility. In this section we show how agents can combine the information and reach an agreement on the reputation of other agents through communication so as to cooperatively react against intruders.

We assume that agents can communicate via one–hop links in order to reduce their detection uncertainty and "converge" to a unique network decision. In this respect we need to introduce concepts involving procedures and algorithms aiming to reach an agreement in networks.

A. The Consensus Algorithms

Consider a network whose communication topology is represented by an undirected graph G which is composed by a set of nodes $V = \{v_1, \ldots, v_n\}$ linked by edges s.t. an edge $e_{i,j}$ means that the node v_i is able to communicate with node v_j. In our case for each agent $\mathcal{A}_i \in N_j$ there exist an arc from node i to node j associated to robots $\mathcal{A}_i$ and $\mathcal{A}_j$ respectively.

Given a graph G, a *consensus algorithm* is an iterative interaction rule that specifies how each node v_i updates its estimates of the generic information $s \in S$ shared among neighbors based on any received value v_j, i.e. it specifies the function $\xi : S \times S \to S$ which is used to compute

$$s_i^+ = \xi(s_i, s_j), \qquad \text{for } i, j = 1, \ldots, n.$$

If the iteration of each node converges toward a common value, a consensus is reached. Typical consensus algorithms available from the literature assume that exchanged data is represented by real numbers [31], [32] and is typically combined according to a weighted average rule. More general cases may require even a nonlinear combination [33], that is still not applicable in our case in which we have n uncertain measures,

In more general cases the quantities of interest could be possibly non-convex sets, intervals, or logical values. Motivated by this fact, we need to involve a more general class of consensus algorithms so as to permit agents sharing locally collected information and eventually "converge" to a unique network decision. Referring to our scenario, nodes are robots that are monitoring a common neighbor and that are supposed to communicate as in G in order to reach an agreement on the reputation of the observed robot $\mathcal{A}_h$. Consider the vector

$$\mathbf{R}_h(t_k) = \begin{bmatrix} r_h^{(1)}(t_k) \\ r_h^{(2)}(t_k) \\ \vdots \\ r_h^{(n)}(t_k) \end{bmatrix} \tag{6}$$

where $r_h^{(i)}(t_k)$ represents the reputation of agent $\mathcal{A}_i$ about agent $\mathcal{A}_h$ after t_k steps of the consensus iterative procedure. Our aim is to design a distributed consensus algorithm allowing us to have $\lim_{k\to\infty} \mathbf{R}_h(t_k) = \mathbf{1}r_h^*$, where r_h^* is the *centralized reputation vector* defined as the vector that would be constructed by a monitor collecting all initial measures and combining them according to ξ.

A possible solution allowing us to reach an agreement consists in letting agents to share the locally estimated *encoder map* φ_h of target robot $\mathcal{A}_h$. In other words, we propose a solution where agents share any information that is directly measured or reconstructed by inspecting its neighborhood through logical consensus [12]. After having established an agreement for the value of the encoder map for a generic agent, they will use the same decision rule and hence decide for the same classification vector. The proposed idea will be formalized and proved in following section. It is worth noting that, the proposed consensus on the encoder map φ_h (or equivalently on the events vector $\mathbf{c}_h$ associated to agent $\mathcal{A}_h$) is the novel contribution of this paper with respect to related works [12], [13], [14].

B. Convergence of Consensus Algorithm

Given a target robot $\mathcal{A}_h$ and n observing agents, the goal of the proposed intrusion detection problem is hence to let the observing agents to exchange locally available information on the events $\mathbf{c}_h$ that influence the behaviour of $\mathcal{A}_h$. Such information is elaborated by each agent and flows among them through a communication network. Once an agreement on the events $\mathbf{c}_h$ is reached the evolution of $\mathcal{A}_h$ is compared with the evolution determined by $\mathbf{c}_h$ and the hybrid dynamic map in (5). If they are not sufficiently close, $\mathcal{A}_h$ is classified as a *misbehaving agent*.

As mentioned, the information exchanged is the event vector $\mathbf{s}_h$ estimated by each robot. More formally, we consider a state vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,\kappa_h}) \in \mathbb{B}^{1\times\kappa_h}$, that is a string of bits representing the values that observing agent $\mathcal{A}_i$ may assign to all sub–events that influence the evolution of agent $\mathcal{A}_h$.

Let $X(t) = (\mathbf{x}_1(t)^T, \ldots, \mathbf{x}_n(t)^T)^T$ be the matrix in $\mathbb{B}^{n\times\kappa_h}$ that represents the network state at the time t. We assume that each agent is a dynamic node that updates its local state $\mathbf{x}_i$ through a distributed logical update function F that depends on its state, on the state of its neighbors and on the observed inputs which is used to initialize the value of the state, i.e. $\mathbf{x}_i(t+1) = F_i(X(t))$. Moreover we assume that every agent is able to produce the logical output vector $Y = (\mathbf{y}_1(t)^T, \ldots, \mathbf{y}_n(t)^T)^T \in \mathbb{B}^{n\times\nu_h}$ that correspond to the detector condition (or events vector) $\mathbf{c}_h$, estimated by the n observing robots, by using an output function D depending on the local state, i.e. $\mathbf{y}_i = D_i(X_i) = \mathbf{c}_h^{(i)} = (c_{h,1}^{(i)}, \ldots, c_{h,\nu_h}^{(i)})$ is the event vector $\mathbf{c}_h$ estimated by robot $\mathcal{A}_i$.

From the *heterogeneous* peculiarity of the system, in terms of sensors and communication devices as well as the fact that agents are placed at different locations w.r.t. the target agent, the generic observing agent i may or may not be able to measure the value of the j–th sub–event associated to $\mathcal{A}_h$, i.e. $s_{h,j}$. In this sense, we can conveniently introduce a *visibility matrix* $V \in \mathbb{B}^{n\times\kappa_h}$ where $V_{i,j} = 1$ if, and only if, agent $\mathcal{A}_i$ is able to measure $s_{h,j}$ and $V_{i,j} = 0$ otherwise. Moreover, each agent is able to communicate only with a subset of other agents. Therefore, to effectively accomplish the given decision task, we need that such an information *flows* from one agent to another, consistently with available communication paths.

The system can hence be described by the logical functions:

$$\begin{cases} X(t+1) &= F(X(t)) \\ \mathbf{x}_i(0) &= \tilde{\mathbf{U}}^{(i)} \\ Y(t) &= D(X(t)) \end{cases} \tag{7}$$

where $F : \mathbb{B}^{n\times\kappa_h} \to \mathbb{B}^{n\times\kappa_h}$, $\Phi = Diag(\varphi, \ldots, \varphi)$ with $\varphi : \mathbb{B}^{\kappa_h} \to \mathbb{B}^{\nu_h}$, and $\tilde{\mathbf{U}}^{(i)}$ provide $\tilde{\mathbf{s}}_h^{(i)}$ that is an initial lower approximation of $\mathbf{s}_h = (s_{h,1}, \ldots, s_{h,\kappa_h})$ based only on observation of the neighborhood of $\mathcal{A}_h$ operated by the i–th agent. In this paper, component–wise inequalities are considered, i.e. a lower approximation is a vector whose j–th component is less or equal to the j–th component of $\mathbf{s}_h$.

We can now introduce the consensus algorithm, i.e. the logical function F, and state the following result

Theorem 1: Given a connected communication graph G, n initial estimates $X(0) = (\tilde{\mathbf{s}}_h^{(1)T}, \ldots, \tilde{\mathbf{s}}_h^{(n)T})^T$, and a visibility matrix V with non–null columns, the distributed logical consensus system

$$\begin{cases} x_{i,j}^+ = V_{i,j}\, x_{i,j} + \neg V_{i,j}\, \left(\sum_{k\in N_i} x_{k,j}\right) , \\ x_{i,j}(0) = \tilde{s}_{h,j}^{(i)} , \end{cases} \tag{8}$$

with $i = 1, \ldots, n$ and $j = 1, \ldots, \kappa_h$ converges to the consensus state $X = \mathbf{1}_n\, \mathbf{s}_h$ in a number of steps that is less than the graph diameter.

Proof: To prove the proposition, consider factorizing the update rule as follows. If $V_{i,j} = 0$, agent i is unable to autonomously compute $s_{h,j}$. In this case equation (8) reduces to

$$x_{i,j}^+ = \neg V_{i,j} \left(\textstyle\sum_{k\in N_i} x_{k,j}\right) = \textstyle\sum_{k\in N_i} x_{k,j} \, .$$

Moreover, since each column of V is non–null by hypothesis, there is at least one observing robot, say the m–th, with complete visibility on the j–th topology $\eta_{h,j}$, which implies $\tilde{s}_{h,j}^{(m)} = s_{h,j}$. Note that we have $\tilde{s}_{h,j}^{(m)} \geq \tilde{s}_{h,j}^{(i)}$ since $\tilde{s}_{h,j}^{(i)}$ is a lower approximation of $s_{h,j}$. Since G is connected, the real value $s_{h,j}$ is propagated from agent m to the rest of the network, which implies that there exists a time $\bar{N} \leq Diam(G) < \infty$ after which

$$x_{i,j} = \sum_{k=1}^{n} x_{k,j} = \tilde{s}_{h,j}^{(q)} = s_{h,j} \, .$$

If instead $V_{i,j} = 1$, agent i has complete knowledge of the j–th topology $\eta_{h,j}$ and its update rule (8) specializes to

$$x_{i,j}^+ = V_{i,j}\, x_{i,j} = x_{i,j} \, ,$$

and its initial estimate is $\tilde{s}_{h,j}^{(i)} = s_{h,j}$. It trivially holds that $x_{i,j} = s_{h,j}$, which proves the theorem. ■

It is worth noting that the hypothesis on the column of V corresponds to the fact that each topology of $\mathcal{A}_h$ is visible by at least one of the observing agents.

Once a consensus has been reached on the event vector $\mathbf{s}_h$ we still need to prove that the output of system (7) solves the intrusion detection problem and allows to identify if the target robot follows or not the predefined rules.

Theorem 2: Given a connected communication graph G, a visibility matrix V with non-null columns, an output function $\Phi = (\varphi, \ldots, \varphi)$ s.t. $\varphi(\mathbf{s}_h) = \mathbf{c}_h$, the distributed logical consensus system (7) where F is given by (8) solves Problem 1.

Proof: To prove the theorem we need to prove that the vector (6) is such that $r_h^{(i)}(\bar{N}) = r_h^*$ with $\bar{N} < \infty$ and $i = 1, \ldots, n$. With Theorem 1 we proved that $X(\bar{N}) = \mathbf{1}_n\, \mathbf{s}_h$. This means that $\Phi_i(\mathbf{x}_i) = \varphi(\mathbf{s}_h) = \mathbf{c}_h$ and the predictor $\tilde{\mathcal{H}_h}^{(i)}(\tilde{\mathbf{q}}_h^{(i)}, \tilde{\sigma}_h^{(i)}, \tilde{I}_h^{(i)})$, $i = 1, \ldots, n$, (introduced in Section IV-A) is initialized with the value $\tilde{\sigma}_h^{(i)}(0)$ corresponding to the most conservative hypothesis on the activation of $\mathbf{c}_h$ which is the same for all observing agents, i.e. $\tilde{\sigma}_h^{(i)}(0) = \tilde{\sigma}_h^*(0)$, $i = 1, \ldots, n$ where $\tilde{\sigma}_h^*(0) = \tilde{\sigma}_h(0)$ in the case the estimated event vector $\tilde{\mathbf{c}}_h$ is equal to $\mathbf{c}_h$. Thus, we have that the estimated state $\tilde{\sigma}_h^{(i)}$, $i = 1, \ldots, n$ becomes

$$\begin{aligned} \tilde{\sigma}_h^{(i)}(t_{k+1}) &= \tilde{\delta}(\tilde{\sigma}_h^{(i)}(t_k), \tilde{\mathbf{c}}_h^{(i)}(t_{k+1})) \\ &= \tilde{\delta}(\tilde{\sigma}_h^*(t_k), \tilde{\mathbf{c}}_h(t_{k+1}))) = \tilde{\sigma}_h^*(t_{k+1}) \end{aligned}$$

According to this, we can compute α_r as

$$\begin{aligned} \|\bar{\mathbf{q}}_h(t) - \pi_{\mathcal{Q}}(\phi_{\tilde{\mathcal{H}}}(\bar{\mathbf{q}}_h(t),^i \tilde{\sigma}_h^{(i)}(t_k), I_h^i(t)))\| = \alpha_r, \\ \forall t \in [t_k, t_{k+1},\, i = 1, \ldots, n. \end{aligned} \tag{9}$$

If $\alpha_r > \varepsilon$ we have that the trajectory of $\mathcal{A}_h$ is not compatible with the nominal one and hence it is considered as misbehaving, i.e. $r_h^{(i)} = \mathsf{misbehaving},\ i = 1, \ldots, n$ which proves the theorem. ■

VI. Application to real Scenario: a Case Study Specification

In the EU Project PLANET, we consider an Highly Automated Airfield scenario in which UAVs operations (including taking off, flying and landing) are highly automated with none or minor human intervention. In the scenario we consider multiple UAVs that share the same aerial space and that cooperate according to a set of predefined cooperation rules in order to safely perform operations. In this context, a UAV misbehaves when its behavior deviates from the assigned rules due to any cause including failures, malicious tampering, wind, and guidance problems. Given, the possible safety consequences, a misbehavior must be thus detected by the other UAVs so that appropriate countermeasures can be taken.

In order to practically verify the effectiveness of the MMD theory presented in the previous sections, we consider a scenario involving four UAVs that cooperate to avoid collisions. We assume that UAVs, in order to avoid collisions and maintain safety, can take the same set of maneuvers, namely, i) to accelerate up to maximum speed (FAST); ii) to change route to the right with a predefined angle (RIGHT) upon detecting a possible collision with another UAV. We also assume that UAVs have the same model of dynamics and controllers and that they can communicate in one-hop.

A. Experimental Setup

The described scenario will involve two real UAVs and two *simulated* UAVs, i.e., UAVs whose behavior will be simulated by a Simulink Model. In the following paragraph we will briefly describe the main characteristics of the simulator and of the 2 real UAVs: the "Skywalker" and the "Locomove" aircraft.

1) Simulator: In Fig. 3 is reported the Simulink model used for the simulations aiming at evaluating the MMD.

It is composed by two kind of different blocks: the *Airplane block* (Fig. 5(a)) and the *Monitor Block* (Fig. 5(b)). The airplane block represents the aircraft standard model which has been implemented according to the agents model defined in Section III. In particular, the event–driven dynamics implements a cooperative behavior based on simple rules to avoid collisions among aircraft while executing the assigned task. Indeed, each aircraft follows the assigned flight plan, and if a collision is detected it changes its route to right with a predefined angle following the rules represented in Fig. 4. It is worth noting that the minimum distance that triggers a collision alarm in aircraft is chosen such as, if the behavior is correct, aircraft are allowed to avoid themselves in any conditions.

The time driven dynamics has been provided by CATEC and it approximates the behavior of the real aircraft that will be then used in the experiments (see Section VI-A).

The monitor block represents the distributed Motion Misbehavior Detection system presented in Section V. In particular it implements the proposed two–step process: first, the aircraft combines the information gathered from on–board sensors and from neighbors by using communication and computes an a–priori prediction of the set of possible trajectories that the observed aircraft should execute based on the cooperative rules (*prediction phase*); then, the predicted trajectories are compared against the one actually executed and measured by the aircraft itself and if none result close enough, the observed aircraft is selected as uncooperative (*verification phase*).

2) The Locomove and the Skywalker Aircraft: The "Locomove" (Fig. 6(a)) is an UAV, designed, developed, and built in FADA-CATEC. It is sufficiently lightweight and has an adequate size such that it can be hand launched. The maximum take-off weight is $5.5\ kg$. A minimum endurance of 40 min is achieved and as power plant an electrical motor powered with LiPo batteries is used . The payload is $500-600\ g$ which allows the user to implement different sensors for a wide range of applications. In full autonomous mode, the aircraft is able to land in flat grounds with wide clearance to obstacles. The UAV lands over its belly, which has a reinforcement to avoid any damage in the fuselage. The aircraft uses a ground based realtime barometric pressure corrections server to precisely measure the altitude over the ground level at the landing area. For the touchdown phase, the aircraft also uses a sonar range finder.

The "Skywalker" (Fig. 6(b)) is instead a commercial product. It is lightweight and it has an adequate size such that it can be hand launched. The maximum take-off weight is $3\ kg$. It has a minimum endurance

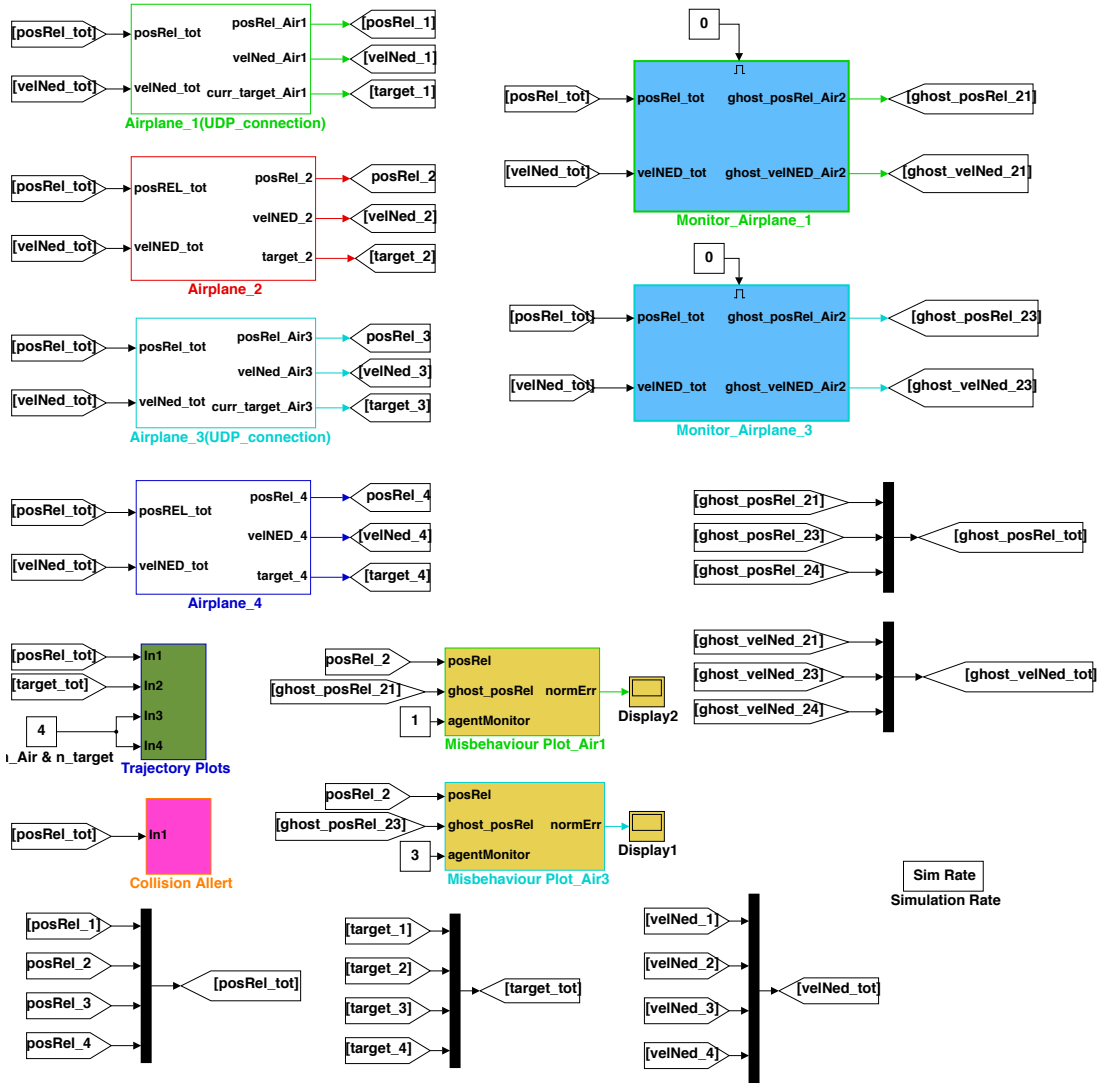


Fig. 3. The Simulink Model used for the simulations.

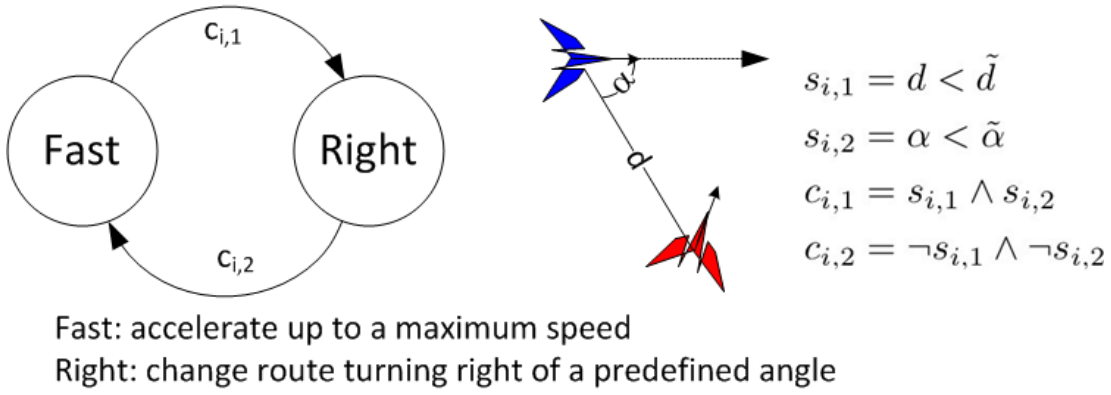
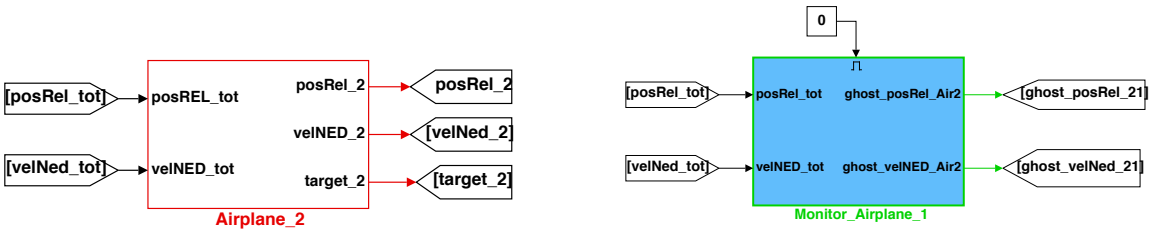


Fig. 4. Rules with related automaton used in the experiments.

of 30 min since it uses an electrical motor powered with LiPo batteries. The payload is $250 - 300\ g$ which allows the user put on–board the sensors for different kind of applications.



(a) Airplane Simulink block (b) Monitor Simulink block

Fig. 5. Components of the Simulink Model.

(a) (b)

Fig. 6. The Locomove (Fig. 6(a)) and the Skywalker (Fig. 6(b)).

3) The Autopilot: Both the Skywalker and the Locomove in fully autonomous mode are flying under the control of an autopilot which has been developed by CATEC. CATEC's autopilot has been designed to serve as a generic prototyping platform of GNC algorithms allowing the adoption of a model based design approach with rapid prototyping capability. It provides a wide enough diversity of interfaces to be able to connect different sensors and payloads and implements appropriate safety mechanisms in order to assure safe operations (Fig 7). This autopilot has been used in the PLANET project to develop guidance, navigation and control algorithms and allows the integration of Simulink models, so developing and testing different algorithms is easy and fast.

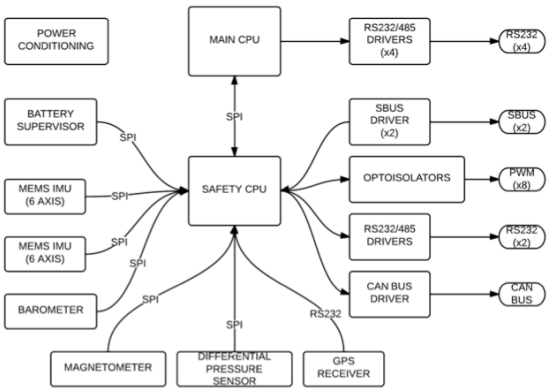


Fig. 7. The main hardware subsystems that are part of the autopilot.

The software of the autopilot has been designed with the Matlab Simulation tool Simulink. It is designed in a modular way, so it is made up of modules and each of them has a certain and specific function. They connect/communicate each other by means of a set of interfaces that has been defined in order to standardize the design methodology. The different modules that form the overall system are: estimator, navigator, controller, RC signals manager and a manager of signals for servos (Fig 8). Each module is independent and exchangeable, so different estimation, navigation or control algorithms can be tested by

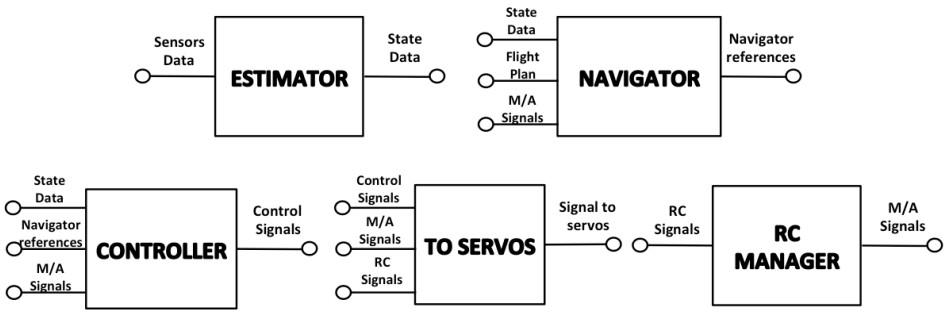


Fig. 8. Autopilot Software Overview: modules and interfaces.

adding a new module that fulfills the required interfaces.

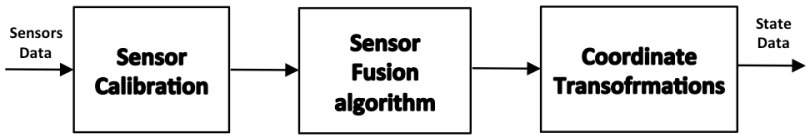


Fig. 9. Estimator blocks diagram.

The estimator is reported in Fig. 9. It receives the data from the sensors: position and velocity from GPS, accelerations, rates and magnetic field components from the IMU and pressure data from the ADS. The main function of this module is to provide an accurate estimation of the aircraft states (global and relative position, ground speed, airspeed and attitude). For that purpose, the estimator combines inertial measurements with GPS information in an optimal way in order to have states information with high frequency and similar or lower position error than the GPS data has.

The navigator deals with the commands given in the Flight Plan. It combines the states of the aircraft (State Data) with the flight plan commands in order to generate the references (Navigator References) that must be followed by the controller. The references are generated depending on the flight mode: route of waypoints, take-off, landing, cruise flight, loitering, etc. The controller is responsible for the generation of signals for the actuators in order to have the required behavior in the aircraft so it follows the commands of the flight plan. The controller receives the Navigator References and the State Data, computes the error between the desired state and the current state and after a sequence of several loops generates the commands for the actuators. The output signals from the controller must be adapted to signals for the servos. In this module, the conversion from controller signal to I2C or PWM signal is done.

B. Experimental Results

Consider the distributed system composed of 4 autonomous UAVs presented above (Fig 11). The four UAVs, namely $\mathcal{A}_1$ (cyan), $\mathcal{A}_2$ (red), $\mathcal{A}_3$ (green), $\mathcal{A}_4$ (blue) are flying over waypoints on the ground that are represented by black points. $\mathcal{A}_1$ (cyan), $\mathcal{A}_4$ (blue) are simulated , while $\mathcal{A}_2$ (red) is the Locomove, and $\mathcal{A}_3$ (green) is the Skywalker. An image of the real scenario with plotted trajectories and waypoints is reported in Figure 10.

With reference to Figure 11, assume that the observed Locomove $\mathcal{A}_2$ is performing a FAST maneuver even though a collision with $\mathcal{A}_1$ should be detected. The UAV is uncooperative as its (actual) trajectory is different from that expected from the cooperation rules. Actually, according to the expected trajectory, $\mathcal{A}_2$ should perform a RIGHT maneuver to avoid collision with $\mathcal{A}_1$.

Monitors on UAVs $\mathcal{A}_1$, $\mathcal{A}_3$, and $\mathcal{A}_4$ combine the information from the on-board sensors and the information gathered from the other UAVs to learn whether $\mathcal{A}_2$ is cooperative or not. Experiments shows

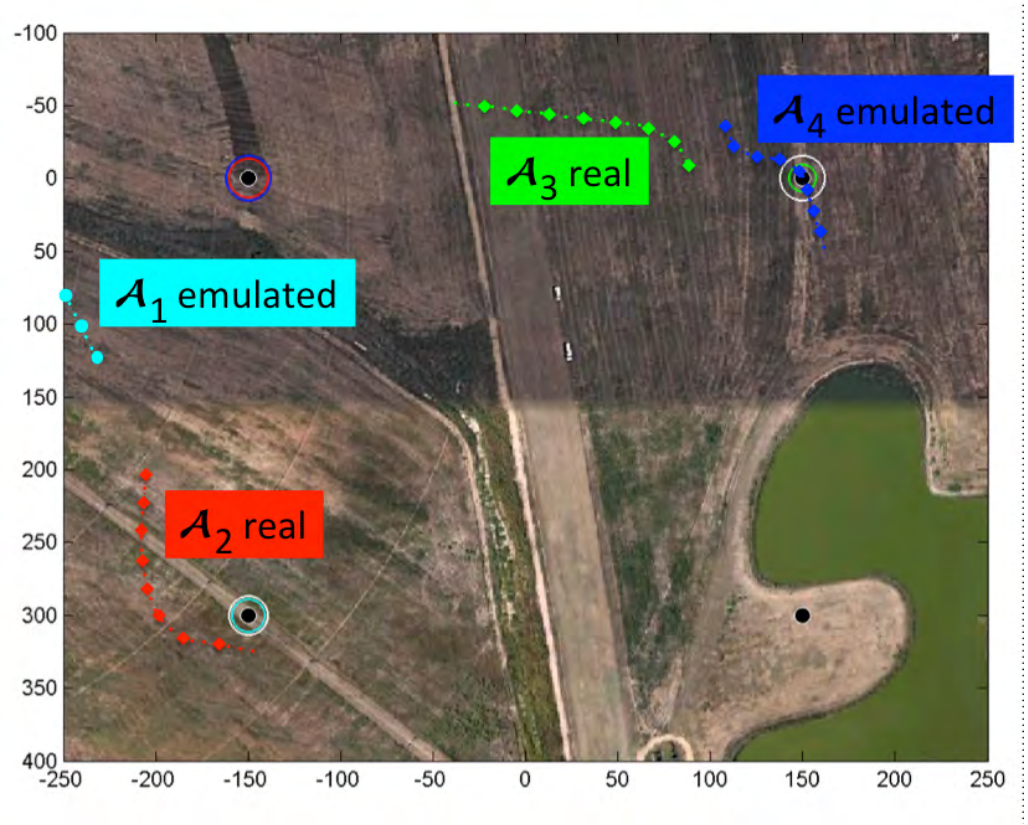
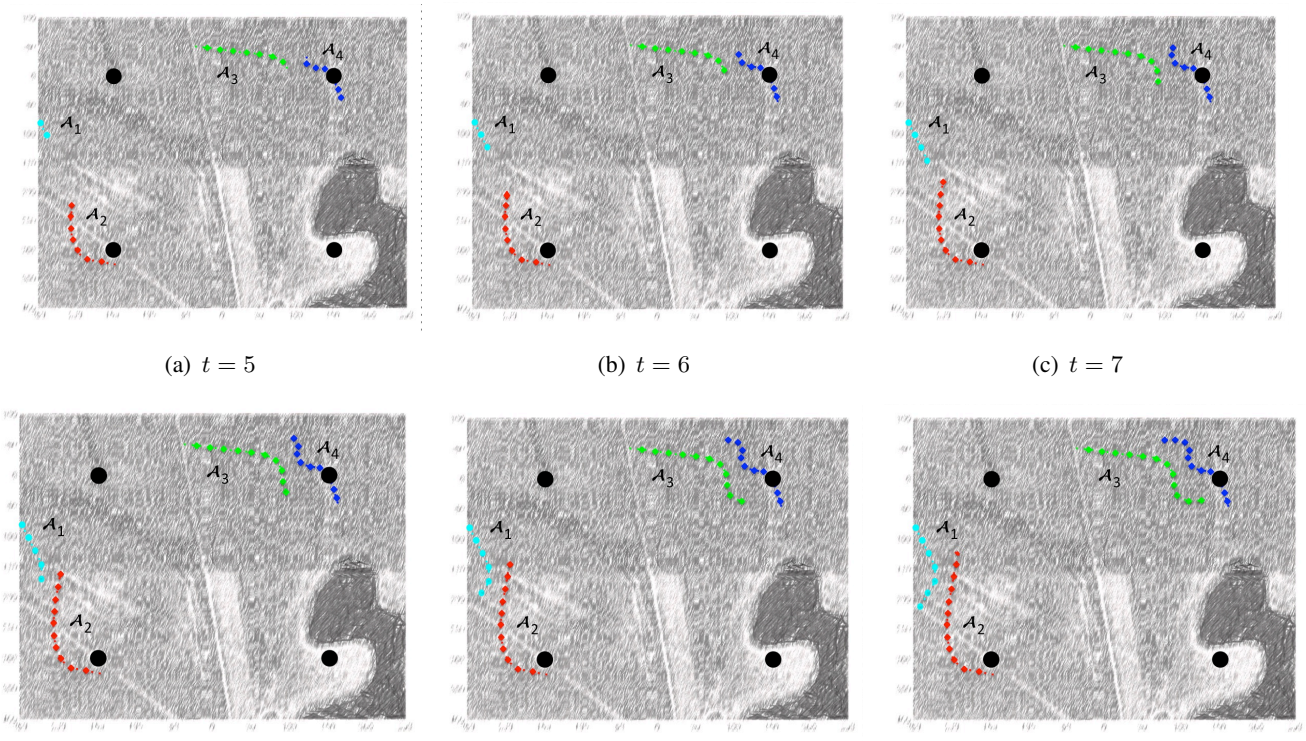


Fig. 10. Picture of the real scenario with plotted trajectories and waypoints.

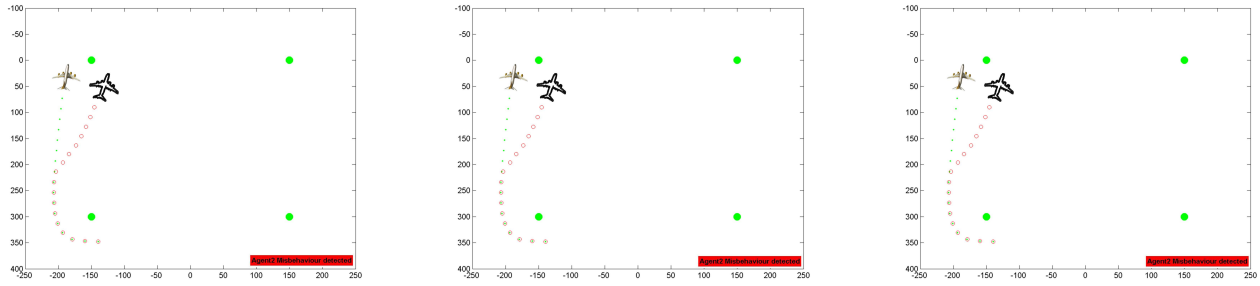


(a) $t = 5$ (b) $t = 6$ (c) $t = 7$

(d) $t = 8$ (e) $t = 9$ (f) $t = 10$

Fig. 11. Experiments run of the scenario described in Section VI-A. The Locomove (red) UAV is violating the assigned rules since it is not applying any collision avoidance rule.

that UAVs $\mathcal{A}_1$, $\mathcal{A}_3$, and $\mathcal{A}_4$ correctly execute the monitor and consent on the uncooperativeness of UAV $\mathcal{A}_2$. Indeed, Figures 12(a)–12(b) show the run of monitors of UAVs $\mathcal{A}_1$, $\mathcal{A}_3$, and $\mathcal{A}_4$, respectively. They correctly look like the same. Each one displays that the actual trajectory of UAV $\mathcal{A}_2$ differs from the expected one. In the experiment, $\mathcal{A}_3$, the Skywalker, triggers an alarm for the misbehavior of $\mathcal{A}_2$, the Locomove, allowing the system to take an adequate countermeasures.



(a) Monitor of $\mathcal{A}_1$ (b) Monitor of $\mathcal{A}_3$ (c) Monitor of $\mathcal{A}_4$

Fig. 12. Run of the monitors of $\mathcal{A}_1$, $\mathcal{A}_3$, and $\mathcal{A}_4$ when $\mathcal{A}_2$ misbehaves (see Fig.11).

VII. Conclusion and Future Work

In this paper we have presented a method for designing distributed algorithms for detecting misbehaviours in systems composed of a group of autonomous cooperative objects. The detection mechanism that we have presented give robots the ability to monitor the behavior of neighbors and to detect robots that do not follow the assigned cooperation rules, due to spontaneous failure or malicious intent. The method is fully distributed and is based on a local monitor that can be systematically built once the cooperation rules are specified. Although the method is general and can be applied to a wide range of applications, it has been tested with simple experiments involving real UAVs: results have been encouraging and motivate future research on this topic aiming at using the MMD as a method to monitor real systems composed of aircraft sharing the environment. Furthermore, starting from the previous experience on work [14], [34], future developments of MMD will consider the Byzantine Generals disagreement problem for the consensus approach in order to add the necessary redundancy in sensors to make the MMDS robust also to the failure of them.

Acknowledgments

This work has been supported by the European Commission within the Integrated Project PLANET, “PLAtform for the deployment and operation of heterogeneous NETworked cooperating objects,” grant no. FP7-2010-257649 (http://planet.etra-id.com/); the Italian Ministry of Education, University and Research within the PRIN project TENACE, “Protecting National Critical Infrastructures from Cyber Threats,” grant no. 20103P34XC_008 (http://www.dis.uniroma1.it/~tenace/); and, the Tuscany Region within the project PITAGORA, “Innovative technologies and processes for Airport Management” under the POR CReO 2007-2013 Framework.

References

[1] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: algorithms and theory,” *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[2] A. Danesi, A. Fagiolini, I. Savino, L. Pallottino, R. Schiavi, G. Dini, and A. Bicchi, “A scalable platform for safe and secure decentralized traffic management of multi-agent mobile system,” *ACM Proc. Real–World Wireless Sensor Network*, 2006.

[3] B. McQueen and J. McQueen, *Intelligent transportation systems architectures*. Artech House Publishers, 1999.

[4] C. Kube and E. Bonabeau, “Cooperative transport by ants and robots,” *Robotics and autonomous systems*, vol. 30, no. 1-2, pp. 85–102, 2000.

[5] L. Pallottino, V. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[6] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks*. Princeton University Press, 2007.

[7] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, p. 215, 2007.

[8] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[9] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, Sept. 2004.

[10] A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi, "Consensus-based distributed intrusion detection for multi-robot systems," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, may 2008, pp. 120 –127.

[11] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, 12–14 December 2007, pp. 5594–5599.

[12] A. Fagiolini, E. Visibelli, and A. Bicchi, "Logical Consensus for Distributed Network Agreement," in *IEEE Conf. on Decision and Control*, 2008, pp. 5250–5255.

[13] A. Fagiolini, S. Martini, D. Di Baccio, and A. Bicchi, "A self-routing protocol for distributed consensus on logical information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010. IROS 2010.*, 2010.

[14] S. Martini, D. Di Baccio, A. Fagiolini, and A. Bicchi, "Robust network agreement on logical information," in *18th IFAC World Congress, IFAC 2011*, 2011.

[15] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 4, pp. 509–521, Apr 1998.

[16] J. Kosecka, C. Tomlin, G. Pappas, and S. Sastry, "Generation of conflict resolution manoeuvres for air traffic management," in *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 3, Sep 1997, pp. 1598–1603 vol.3.

[17] L. Pallottino, V. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *Robotics, IEEE Transactions on*, vol. 23, no. 6, pp. 1170–1183, Dec 2007.

[18] S. Kato, S. Nishiyama, and J. Takeno, "Coordinating mobile robots by applying traffic rules," in *Proc. of the lEEE/RSJ International Conference onIntelligent Robots and Systems*, vol. 3, 1992.

[19] A. Bicchi, A. Danesi, G. Dini, S. La Porta, L. Pallottino, I. M. Savino, and R. Schiavi, "Heterogeneous wireless multirobot system," *Robotics and Automation Magazine, IEEE*, vol. 15, no. 1, pp. 62–70, 2008.

[20] A. Bicchi, A. Fagiolini, and L. Pallottino, "Towards a society of robots: Behaviors, misbehaviors, and security," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 4, pp. 26–36, 2010.

[21] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[22] J. Lygeros, "Lecture notes on hybrid systems," in *Notes for an ENSIETA workshop*. Citeseer, 2004.

[23] S. Pearson and B. Balacheff, *Trusted computing platforms: TCPA technology in context*. Prentice Hall PTR, 2003.

[24] B. Chen and R. Morris, "Certifying program execution with secure processors," in *USENIX HotOS Workshop*, 2003, pp. 133–138.

[25] R. Gallo, H. Kawakami, and R. Dahab, "Fortuna-a framework for the design and development of hardware-based secure systems," *Journal of Systems and Software*, 2013.

[26] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization inwireless sensor network," *IEEE Conf. on Decision and Control*, Dec 2007.

[27] A. Giua and C. Seatzu, "Fault detection for discrete event systems using petri nets with unobservable transitions," in *Proc. IEEE Conference on Decision and Control and European Control Conference*, 2005, pp. 6323–6328.

[28] F. Basile, P. Chiacchio, and G. De Tommasi, "Improving on-line fault diagnosis for discrete event systems using time," 2007, pp. 26–32.

[29] Y. Ru, M. Cabasino, A. Giua, and C. Hadjicostis, "Supervisor Synthesis for Discrete Event Systems with Arbitrary Forbidden State Specifications," in *IEEE Conf. on Decision and Control*, 2008, pp. 1048–1053.

[30] S. Martini, A. Fagiolini, G. Zichittella, M. Egerstedt, and A. Bicchi, "Decentralized classification in societies of autonomous and heterogenous robots," in *ICRA. Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[31] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems." *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[32] R. Olfati-Saber, , and R. N. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time–Delays," *IEEE Transactions on Automatic Control*, 2004.

[33] J. Cortès, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, Mar. 2008.

[34] A. Bicchi, A. Fagiolini, G. Dini, and I. M. Savino, "Tolerating, malicious monitors in detecting misbehaving robots," in *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*. IEEE, 2008, pp. 109–114.