A Hybrid Controller for Multi-Agent Collision Avoidance via a Differential Game Formulation

D. Cappello, S. Garcin, Z. Mao, M. Sassano, A. Paranjape and T. Mylvaganam

***Abstract*—We consider the *multi-agent collision avoidance problem* for a team of wheeled mobile robots. Recently, a *local* solution to this problem, based on a game theoretic formulation, has been provided and validated via numerical simulations. Due to its local nature the result is not well-suited for *online* application. In this paper we propose a novel hybrid implementation of the control inputs that yields a control strategy suited for the *online* navigation of mobile robots. Moreover, subject to a certain *dwell time* condition, the resulting trajectories are globally convergent. The control design is demonstrated both via simulations and experiments.**

***Index Terms*—Collision avoidance, multi-robot systems, path planning for multiple robots, nonlinear control, hybrid systems.**

I. Introduction

Systems consisting of several robots, which can be considered as *agents*, have gained interest in recent years, in part due to their numerous applications, including *e.g.* exploration [1] and autonomous monitoring [2], [3]. Several approaches to solve control problems related to multi-agent systems involve concepts borrowed from game theory, for instance, in the context of motion planning, pursuit-evasion problems and formation control (see, *e.g.*, [4], [5], [6], [7], [8]). However, differential games are, in general, difficult to solve. This difficulty is circumvented in [9], [10], where a systematic method for constructing approximate solutions to differential games is provided. The method has proved useful for control of multi-agent systems (see, *e.g.*, [11]).

We consider the so-called *multi-agent collision avoidance problem*: given a system of robots, each robot should (autonomously) manoeuvre itself from its initial position to a predefined target while avoiding collisions with obstacles and other agents. The problem is formulated, studied and solved in a centralized setting, *i.e.* when the agents are not subject to any communication and/or information constraints. Different approaches to this problem are available in the literature. One approach is based on *navigation functions* [12], [13], [14], [15], and ensures that the robots reach their targets (while avoiding collisions) for *almost any* initial condition. Yet, deadlocks *may* occur as observed, for instance, in [14]. In [16] decentralized, velocity-based strategies are used to guarantee multi-agent collision avoidance for a fixed amount of time in the future, however, no proof of convergence towards the target positions has been provided nor has the possibility of deadlocks been ruled out. In [17] (decentralised) control laws which ensure collision avoidance are designed using *safety barrier certificates*. Therein a "collision avoidance strategy" is combined with a generic "nominal controller" and at each time instant a static quadratic optimisation problem is solved to achieve collision avoidance. An alternative approach based on a *Lyapunov-type test function* is provided in [18], [19], where it is noted that a major drawback with this approach is that the *systematic* construction of such test function is an open problem. The multi-agent collision avoidance problem has been considered in [11], [20] wherein locally asymptotically stabilising control laws are proposed. The control laws are such that the trajectories of the robots are collision- and deadlock-free in a non-empty region of the state-space describing the overall system. The result hinges upon a differential game formulation of the problem, which enables the *systematic* construction of a Lyapunov function, thus addressing the challenge identified in [18], [19].

We propose novel *hybrid controllers* with *improved performance* with respect to the main results in [11], [20]. The improved performance of the novel approach lies in that the basin of attraction of the zero equilibrium of the resulting closed-loop system is enlarged *and*, under a certain dwell time condition, the equilibrium is provably *globally* asymptotically stable. Moreover, we propose an algorithm for *online* implementation which is demonstrated on an experiment using two Khepera IV mobile robots developed by K-team. The overall control scheme consists of two parts: the *path planner* (based on *virtual agents*, the formulation of a differential game and a hybrid controller) and the *path tracking* controllers for each individual wheeled mobile robot (WMR), as depicted in Figure 1. Under the aforementioned dwell time condition (which is easily checked online) the resulting control laws are such that the trajectories of the robots are collision-free and *globally* convergent to their target positions. The remainder of the paper is organised as follows. The multi-agent collision avoidance problem is defined in Section II before some preliminaries, concerning its solution and the underlying game theoretic formulation, are provided in Section III. The novel contributions of this paper are presented in the following three sections: an improved (with respect to [11], [20]) path

D. Cappello is with the Department of Aeronautics, Imperial College London, London SW7 2AZ, UK (e-mail: domenico.cappello16@imperial.ac.uk).

S. Garcin is with the Department of Aeronautics, Imperial College London, SW72AZ, UK (e-mail: samuel.garcin14@imperial.ac.uk).

Z. Mao is with the Department of Aeronautics, Imperial College London, SW72AZ, UK (e-mail: ziyuan.mao13@imperial.ac.uk).

M. Sassano is with the Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", Via del Politecnico 1, 00133, Roma, Italy (e-mail: mario.sassano@uniroma2.it).

A. Paranjape is with Tata Consultancy Services Ltd, TRDDC, Pune (e-mail: aditya.paranjape@tcs.com).

T. Mylvaganam is with the Department of Aeronautics, Imperial College London, London SW7 2AZ, UK (e-mail: t.mylvaganam@imperial.ac.uk).

This research has not been externally funded.

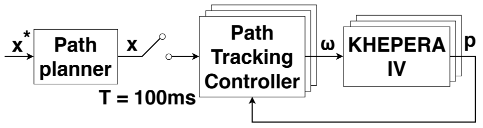

Fig. 1: Block diagram representation of the control scheme.

planning scheme, based on a hybrid controller, is provided in Section IV; an algorithm suitable for *online* implementation is provided in Section V; the combined performance of the two components of the overall control architecture is then demonstrated by means of a numerical simulation and an experiment presented in Section VI. Finally, some concluding remarks are provided in Section VII.

Notation: The set of real numbers and the set of non-negative integers are denoted by $\mathbb{R}$ and $\mathbb{Z}_{\geq 0}$, respectively. The open left-half complex plane is denoted by $\mathbb{C}^-$ and the empty set is denoted by $\emptyset$. Given a vector $x \in \mathbb{R}^n$, the gradient of a function $V : \mathbb{R}^n \to \mathbb{R}$ is denoted by $\frac{\partial V}{\partial x}$. The weighted Euclidean norm of a vector is denoted by $\|x\|_A = \sqrt{x^\top A x}$, where $A^\top = A > 0$. The spectrum of a matrix $M \in \mathbb{R}^{n\times n}$ is denoted by $\sigma(M)$. A block matrix M is denoted by $[M_{ij}]$, where M_{ij} is the i^{th}-row, j^{th}-column block. I_n denotes the n-dimensional identity matrix. $\delta_{-1} : \mathbb{R} \to \mathbb{R}$ denotes the Heaviside step function, and $sat_a : \mathbb{R} \to \mathbb{R}$ denotes the saturation function defined as $sat_a(x) \triangleq \min(a, \max(-a, x))$, with $a > 0$.

II. Problem Formulation

Consider a team of N WMRs moving on the Euclidean plane, possibly in the presence of (static) obstacles, which must be collectively steered from their initial configurations to desired target locations. Each WMR is described by unicycle-like dynamics, *i.e.* each WMR satisfies the dynamics

$$\dot{X}_i = \cos(\theta_i)v_i\,, \quad \dot{Y}_i = \sin(\theta_i)v_i\,, \quad \dot{\theta}_i = \omega_i\,, \tag{1}$$

where the subscript $i = 1, ..., N$ indicates a particular WMR/agent, $(X_i, Y_i) \in \mathbb{R}^2$ is the position of the middle point between the two actuated wheels along the direction of the axle connecting the wheels, θ_i is the orientation of the i-th robot and v_i and ω_i are the longitudinal and the angular velocity controls, respectively. The constant a_i describes the distance between the point (X_i, Y_i) and the centre of mass of the i-th robot $(x_{c,i}, y_{c,i}) \in \mathbb{R}^2$, along the segment connecting (X_i, Y_i) and the (forward) passive wheel.

Consider the case in which there are $m \geq 0$ static obstacles, each represented by its center of mass $p_j^c \in \mathbb{R}^2$ and the region of the Euclidean plane that it occupies $\mathcal{P}_j \subset \mathbb{R}^2$, $j = 1, ..., m$. We assume obstacles are elliptical, *i.e.* the boundary of the region $\mathcal{P}_j$ satisfies $\partial\mathcal{P}_j = \{x \in \mathbb{R}^2 : \|x - p_j\|^2_{E_j} - \rho_j^2 = 0\}$, where $\rho_j > 0$ and $E_j = E_j^\top > 0$, for $j = 1, \dots, m$.

Problem 1 (Multi-agent collision avoidance): Consider a multi-agent system consisting of N WMRs with dynamics (1), for $i = 1, \dots N$. The *multi-agent collision avoidance problem* consists in determining feedback control laws v_i, ω_i, $i = 1, ..., N$, that steer each agent i from its initial position to a predefined target $x_i^\star \in \mathbb{R}^2$, $i = 1, \dots, N$, while avoiding inter-agent collisions as well as collisions with static obstacles. $\diamond$

III. Background and Preliminaries

As demonstrated in [20], the collision avoidance problem can be solved (locally) in two steps. First, N *virtual agents* satisfying single-integrator dynamics are introduced and *efficient*, collision-free trajectories for the virtual agents are determined through the formulation and solution of a differential game. The trajectories of the virtual agents are then interpreted as a *path plan* for the WMRs and the control laws v_i, ω_i, $i = 1, \dots, N$, are designed such that the WMRs track the path plan with zero error. An overview of this approach is presented in this section (see [11], [20] for additional details).

A. Collision-free path planning

Consider N *virtual agents* - one per WMR - described by a pair of single-integrator dynamics, namely

$$\dot{x}_i = u_i\,, \tag{2}$$

where $x_i(t) \in \mathbb{R}^2$ denotes the position and $u_i(t) \in \mathbb{R}^2$ denotes the control input of the i-th virtual agent, for $i = 1, ...N$. Let $\tilde{x}_i = x_i - x_i^\star$, for $i = 1, \dots, N$, and let each agent i be associated with a *safety radius* $r_i > 0$, $i = 1, \dots, N$.

The *obstacle avoidance region* and the *agent avoidance region* are defined as follows.

Definition 1:

i) *Obstacle avoidance region* : Consider the open sets $\mathcal{S}_j = \left\{x \in \mathbb{R}^2 : \|x - p_j\|^2_{E_j} < \rho_j^2\right\}$. The *obstacle avoidance region*, denoted by $\mathcal{S}$, is defined as $\mathcal{S} = \bigcup_{j=1}^{m} \mathcal{S}_j$.
ii) *Agent avoidance region*: Given a time instant $\bar{t} \geq 0$, consider the open sets $\mathcal{D}_{ij}^{\bar{t}} = \left\{x \in \mathbb{R}^2 : \|x - x_j(\bar{t})\|^2 \leq (r_i + r_j)^2\right\}$, $j = 1, ..., N$, $j \neq i$. The *agent avoidance region* of the i-th virtual agent at $\bar{t}$, denoted $\mathcal{D}_i^{\bar{t}}$, is defined as $\mathcal{D}_i^{\bar{t}} = \bigcup_{j=1, j\neq i}^{N} \mathcal{D}_{ij}^{\bar{t}}$. $\diamond$

Collision-free trajectories for the i-th virtual agent (and therefore also the i-th WMR) and feasible paths are classified in the following statements, where $\bar{\mathcal{D}}_i^{\bar{t}}$ and $\bar{\mathcal{S}}$ denote the complement of the sets $\mathcal{D}_i^{\bar{t}}$ and $\mathcal{S}$, respectively.

Definition 2 (Collision-free trajectory): The trajectory of the i-th virtual agent is said to be *collision-free* if $x_i(\bar{t}) \notin \mathcal{D}_i^{\bar{t}} \cup \mathcal{S}$ for all $\bar{t} \geq 0$, or equivalently $x_i(\bar{t}) \in \bar{\mathcal{D}}_i^{\bar{t}} \cap \bar{\mathcal{S}}$, for all $\bar{t} \geq 0$. $\diamond$

Definition 3 (Feasible path): A *path*[1] is said to be feasible if its associated trajectories $x_i(t)$, or time-scaled versions thereof $x_i(\eta t)$, with $\eta > 0$, are collision-free for any $i = 1, ..., N$. $\diamond$

The problem of generating collision-free paths for each of the WMRs can be formulated as a differential game wherein the virtual agents are considered as the *players*.

Problem 2 (Collision-free path generation): Consider the system of N (virtual) agents with dynamics (2), for $i = 1, \dots N$, let $\tilde{x} = \left[\tilde{x}_1^\top, \dots, \tilde{x}_N^\top\right]^\top$, such that

$$\dot{\tilde{x}} = B_1 u_1 + \cdots + B_N u_N\,, \tag{3}$$

[1]A path is defined as a continuous curve connecting two points on $\mathbb{R}^2$. A trajectory associated to the path is a time parameterisation of the points on the curve.

where $B_1 = [I_2, 0, \ldots 0]^\top, \ldots, B_N = [0, \ldots 0, I_2]^\top$, and consider the individual cost functionals

$$J_i(\tilde{x}(0), u_1, \ldots, u_N) = \frac{1}{2}\int_0^\infty \left(q_i(\tilde{x}(t)) + \|u_i(t)\|^2\right) \mathrm{d}t\,, \quad (4)$$

$i = 1, ..., N$, where $q_i : \mathbb{R}^{2N} \to \mathbb{R}$, $q_i(\tilde{x}) > 0$, $q_i(0) = 0$, are running costs given by

$$q_i(\tilde{x}) = \left(\alpha_i + \beta_i^s g_i^s(\tilde{x}) + \beta_i^d g_i^d(\tilde{x})\right)\tilde{x}_i^\top \tilde{x}_i\,, \quad (5)$$

with constants $\alpha_i > 0$, $\beta_i^s > 0$, $\beta_i^d > 0$ and where $g_i^s(\tilde{x}) \geq 0$ and $g_i^d(\tilde{x}) \geq 0$ are continuously differentiable mappings, referred to as collision avoidance functions, such that $\lim\limits_{\tilde{x}+x^\star \to \partial\mathcal{S}} g_i^s(\tilde{x}) = +\infty$ and $\lim\limits_{\tilde{x}+x^\star \to \partial\mathcal{D}_i^t} g_i^d(\tilde{x}) = +\infty$, respectively. Determine a set of (possibly dynamic) strategies $(u_1^\star, \ldots, u_N^\star)$ satisfying the following conditions:

i) The origin of the system (3) in closed-loop with the strategies $(u_1^\star, \ldots, u_N^\star)$ is (locally) asymptotically stable.

ii) The inequalities

$$\begin{aligned} &J_i(\tilde{x}(0), u_1^\star, \ldots, u_{i-1}^\star, u_i^\star, u_{i+1}^\star, \ldots, u_N^\star) \\ &\leq J_i(\tilde{x}(0), u_1^\star, \ldots, u_{i-1}^\star, u_i, u_{i+1}^\star, \ldots, u_N^\star) + \epsilon_\alpha\,, \end{aligned} \quad (6)$$

$i = 1, \ldots, N$, where $\epsilon_\alpha > 0$, are satisfied for all $u_i \neq u_i^\star$ such that $\sigma(A_{cl} + \alpha I) \in \mathbb{C}^-$, where A_{cl} is the matrix describing the linearisation of the system (3) in closed-loop with $(u_1^\star, \ldots, u_{i-1}^\star, u_i, u_{i+1}^\star, \ldots, u_N^\star)$ about the origin. $\diamond$

Consider the following standing assumptions, formulated in terms of the virtual agents, which ensure the feasibility of Problem 2 (and of Problem 1).

Assumption 1:

i) *Obstacle collision-free initial deployment*: the initial positions of the agents satisfy $\{x_i(0)\} \cap \mathcal{S} = \emptyset$, for all $i = 1, \ldots, N$, $j = 1, \ldots, m$.

ii) *Agent collision-free initial deployment*: the initial positions of the agents satisfy $\|x_i(0) - x_j(0)\| > r_i + r_j$, for all $i = 1, \ldots, N$ and $j = 1, \ldots, N$, $j \neq i$.

iii) *Obstacle collision-free desired deployment*: the target positions of the agents satisfy $\{x_i^\star\} \cap \mathcal{S} = \emptyset$, for all $i = 1, \ldots, N$, $j = 1, \ldots, m$.

iv) *Agent collision-free desired deployment*: the target positions for each agent satisfy $\|x_i^\star - x_j^\star\| > r_i + r_j$, for all $i = 1, \ldots, N$ and $j = 1, \ldots, N$, $j \neq i$.

v) *Configuration feasibility:* the static obstacles do not form an impermeable boundary about targets of one or more of the agents. Namely, there exists a continuous path l_i connecting the initial condition $x_i(0)$ and the target $x_i^\star$ satisfying $l_i \cap \left(\bigcup\limits_{j=1,\ldots,m} \partial\mathcal{P}_j\right) = \emptyset$, for $i = 1, \ldots, N$.

The main result of [11], recalled in the following statement, yields a closed-form solution of Problem 2. Let $\xi = \left[\xi_1^\top, \ldots, \xi_N^\top\right]^\top \in \mathbb{R}^{2N}$, where $\xi_i \in \mathbb{R}^2$, represents the state of a dynamic controller, $i = 1, \ldots, N$, and consider the matrix-valued mappings $P_1(\tilde{x}), \ldots, P_N(\tilde{x})$, where $P_i : \mathbb{R}^{2N} \to \mathbb{R}^{2N\times 2N}$, $i = 1, \ldots, N$, are given by $P_i(\tilde{x}) = \left[P_{kj}^i(\tilde{x})\right]^\top + \gamma_i I_{2N}$, where $P_{kj}^i \in \mathbb{R}^{2\times 2}$, $k = 1, \ldots N$, $j = 1, \ldots, N$ and $\gamma_i > 0$ is a constant parameter, $P_{kj}^i = 0$ for all $k \neq i$ and for all $j \neq i$, and $P_{ii}^i(\tilde{x}) = \left[\sqrt{\alpha_i + \beta_i^s g_i^s(\tilde{x}) + \beta_i^d g_i^d(\tilde{x})} I_2\right]$. Consider the functions $V_i : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$V_i(\tilde{x}, \xi) = \frac{1}{2}\tilde{x}^\top P_i(\xi)\tilde{x} + \frac{1}{2}(\tilde{x} - \xi)^\top R_i(\tilde{x} - \xi)\,, \quad (7)$$

where $R_i = R_i^\top > 0$ and consider the set $\mathcal{M} = \{\xi \in \mathbb{R}^{2N} : g_i^s(\xi) + g_i^d(\xi) < \infty\}$.

Lemma 1 ([11, Theorems 1,2]): Consider the system (3) and the matrix-valued mappings $P_i(\tilde{x})$, $i = 1, \ldots, N$. Let $\xi(0) \in \mathcal{M}$ and suppose that Assumption 1 holds. Then there exist $\bar{k} \geq 0$, R_i, $i = 1, \ldots, N$, and a neighbourhood $\Omega \subseteq \mathbb{R}^{2N} \times \mathbb{R}^{2N}$ containing the origin of the extended state-space (x, ξ) such that the inequalities

$$\begin{aligned} \mathcal{HJ}_i(\tilde{x}, \xi) &= -\frac{1}{2}\frac{\partial V_i}{\partial \tilde{x}} B_i B_i^\top \frac{\partial V_i}{\partial \tilde{x}}^\top + \frac{1}{2}q_i(\tilde{x}) \\ &- \sum_{j=1, j\neq i}^N \frac{\partial V_i}{\partial \tilde{x}} B_j B_j^\top \frac{\partial V_j}{\partial \tilde{x}}^\top - k\sum_{j=1}^N \frac{\partial V_i}{\xi}\left(\frac{\partial V_j}{\partial \xi}\right)^\top \leq 0\,, \end{aligned} \quad (8)$$

$i = 1, \ldots, N$, are satisfied *by construction* and the dynamic strategies

$$u_i = -B_i^\top \frac{\partial V_i}{\partial \tilde{x}}^\top\,, \quad \dot{\xi} = -k\sum_{j=1}^N \left(\frac{\partial V_j}{\partial \xi}\right)^\top\,, \quad (9)$$

$i = 1, \ldots, N$, constitute a solution for Problem 2 in the neighbourhood Ω, for all $k \geq \bar{k}$. Consequently, all trajectories which do not leave the set Ω are such that $\lim\limits_{t\to\infty} \xi(t) = 0$, $\lim\limits_{t\to\infty} \tilde{x}_i(t) = 0$ and $x_i(\bar{t}) \in \bar{\mathcal{D}}_i^{\bar{t}} \cap \bar{\mathcal{S}}$, for all $\bar{t} \geq 0$, $i = 1, \ldots, N$. Lemma 1 implies that the dynamic control laws (9), $i = 1, \ldots, N$, are such that the trajectories $x_i(t)$, $i = 1, \ldots, N$, are collision-free an deadlock-free *locally*.

The solution of Problem 2 given in Lemma 1 is such that the resulting path is feasible and the requirements of Definition 3 hold for any $\eta > 0$, namely any time-scaled trajectories $x_i(\eta t)$, $i = 1, \ldots, N$, with $\eta > 0$ common to all virtual agents, are collision-free.

B. Path tracking

The differential game in Problem 2 is essentially a *tool* used to generate feasible paths and the result in Lemma 1 can be used to solve Problem 1 as summarized here [20]. Let $p_i = (x_{c,i}, y_{c,i})$ denote the centre of mass of the i-th WMR, recall that x_i denotes the position of the $i-th$ virtual agent and consider the notation $x_i = (x_i^1, x_i^2)$, $i = 1, \ldots, N$.

Lemma 2 ([20, Theorem 2]): Consider a team of N WMRs described by the dynamics (1) and a set of desired target locations $x_i^\star = (x_{c,i}^\star, y_{c,i}^\star) \in \mathbb{R}^2$, $i = 1, ..., N$. Suppose $p_i(0) = x_i(0)$, for all $i = 1, \ldots, N$, and suppose Assumption 1 holds. Then, the dynamic control law (10) (see top of next page) with $\kappa > 0$, solves the multi-agent collision avoidance problem for the WMRs *locally*, *i.e.* for all trajectories such that $(\tilde{x}(t), \xi(t)) \in \Omega$ for all $t \geq 0$. $\diamond$

While the result of Lemma 1 is well-suited for planning collision-free paths *offline*, it is not suitable for an *online* implementation. The main drawbacks towards such an extension are related to the local nature of the result: the initial conditions must be such that $(\tilde{x}(0), \xi(0)) \in \Omega$, where the neighbourhood

$$\dot{\xi} = -k \sum_{j=1}^{N} \left(\frac{\partial V_j}{\partial \xi}\right)^\top, \quad \begin{bmatrix} \dot{x}_i^1 \\ \dot{x}_i^2 \end{bmatrix} = -B_i^\top \frac{\partial V_i}{\partial \tilde{x}}^\top, \tag{10a}$$

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) \\ -\frac{1}{a}\sin(\theta_i) & \frac{1}{a}\cos(\theta_i) \end{bmatrix} \begin{bmatrix} \kappa(x_i^1 - X_i - a_i\cos(\theta_i)) + \dot{x}_i^1 \\ \kappa(x_i^2 - Y_i - a_i\sin(\theta_i)) + \dot{x}_i^2 \end{bmatrix} \tag{10b}$$

Ω is not known *a priori*; and even provided $(\tilde{x}(0), \xi(0)) \in \Omega$, it should be additionally ensured that the trajectory of the system remains in Ω at all times, *i.e.* that $(\tilde{x}(t), \xi(t)) \in \Omega$, for any $t \geq 0$. These drawbacks are addressed in the following section.

IV. Path planning: a hybrid implementation

We propose a novel control design methodology for the virtual agents (in place of (9), for $i = 1, \ldots, N$), yielding a closed-loop system that can be interpreted as a hybrid system with state-driven jumps. The new control architecture allows us to overcome the main issues preventing online implementation of the strategy in [11]. The design approach, similar in spirit to the hybrid control proposed in [21], is aimed at *expanding* the neighbourhood Ω in which the results of Lemma 1 holds.

Lemmas 1 and 2 imply that the paths of the WMRs are *feasible* according to Definition 3 and such that the robots reach their targets provided the trajectory $(\tilde{x}(t), \xi(t))$ is such that the inequalities (8), $i = 1, \ldots, N$, are satisfied for all $t \geq 0$. Moreover, the state $\xi(t)$ used in the dynamic control laws (9) and (10), $i = 1, \ldots, N$, is a variable which has been artificially introduced as the internal state of the controller and which does not correspond directly to any physical aspect (such as position, velocity, *etc.*) of the WMRs. Exploiting these observations we propose a *hybrid implementation* of the control strategies (9), $i = 1, \ldots, N$, wherein ξ is allowed to *jump*, *i.e.* to be suitably reset. To this end, consider the sets

$$\begin{aligned} C &= \{(\tilde{x}, \xi) : \max_i \{\mathcal{HJ}_i(\tilde{x}, \xi)\} < -\rho(\tilde{x}, \xi)\}, \\ D &= \{(\tilde{x}, \xi) : \max_i \{\mathcal{HJ}_i(\tilde{x}, \xi)\} \geq -\rho(\tilde{x}, \xi)\}, \end{aligned} \tag{11}$$

for $i = 1, \ldots, N$, where ρ is continuously differentiable and such that $\rho(\tilde{x}, \xi) > 0$ for all $(\tilde{x}, \xi) \neq 0$, $\rho(0,0) = 0$ and $\rho(\tilde{x}, \xi) \leq c$, for any $\tilde{x}$ and ξ, for some $c > 0$. The set C (the "flow set") is the subset of $\mathbb{R}^{2N} \times \mathbb{R}^{2N}$ in which the inequalities (8), $i = 1, \ldots, N$, are satisfied with some *negativity margin* ρ, whereas D (the "jump set") is the closure of the complement of this set. Consider, in place of (9), the hybrid control laws

$$u_i = -B_i^\top \frac{\partial V_i}{\partial \tilde{x}}^\top, \tag{12a}$$

$$\dot{\xi} = -k \sum_{j=1}^{N} \left(\frac{\partial V_j}{\partial \xi}\right)^\top, \quad (\tilde{x}, \xi) \in C, \tag{12b}$$

$$\xi^+ = \zeta^\star, \quad (\tilde{x}, \xi) \in D, \tag{12c}$$

for $i = 1, \ldots, N$, where $\zeta^\star$ is a solution of

$$\zeta^\star = \text{argmin}_\zeta \Pi(\tilde{x}, \zeta), \tag{13a}$$

$$\text{s.t.} \quad \max_i \{\mathcal{HJ}_i(\tilde{x}, \zeta)\} \leq -\mu\rho(\tilde{x}, \zeta), \tag{13b}$$

$$W(\tilde{x}, \zeta) \leq W(\tilde{x}, \xi), \tag{13c}$$

where $W(\tilde{x}, \xi) = \sum_{i=1}^{N} V_i(\tilde{x}, \xi)$, $\Pi : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \to \mathbb{R}$ is a lower semicontinuous function such that $\Pi(\tilde{x}, \xi)$ is level-bounded in ξ, locally uniformly in $\tilde{x}$ and $\mu > 0$ is a design parameter. The main result of [21] states that, provided there exists ζ satisfying the constraints (13b) and (13c) for all $(\tilde{x}, \xi) \in \mathbb{R}^{2N} \times \mathbb{R}^{2N}$, the origin of the closed-loop system (2)-(12) is globally asymptotically stable. However, in the setting considered herein it is not possible - in general - to ensure that both conditions (13b) *and* (13c) are satisfied simultaneously. Thus, we propose an alternative hybrid controller in which the (*hard*) constraint (13c) in the optimisation problem (13) is replaced by a *soft* constraint along with a dwell time requirement. In the following statement, $[t_k, t_{k+1}]$ denotes the (continuous) time interval between the two successive discontinuous jumps indexed as k and $k+1$, $k \in \mathbb{Z}_{\geq 0}$, and t_k^- and t_k^+ denote the time instants "right before" and "right after" the k-th jump, respectively.

Theorem 1: Consider the N virtual agents and the corresponding system (3), $i = 1, \ldots, N$. Let $\zeta^\star$ be a solution of the (static) optimisation problem

$$\zeta^\star = \text{argmin}_\zeta W(\tilde{x}, \zeta), \tag{14a}$$

$$\text{s.t.} \quad \max_i \{\mathcal{HJ}_i(\tilde{x}, \zeta)\} \leq -\mu\rho(\tilde{x}, \zeta), \tag{14b}$$

for some $\mu > 1$ and for any $\tilde{x} \neq 0$. Consider the closed-loop system (3)-(12), with $\zeta^\star$ as in (14a), and suppose that the *flow* of (3)-(12) in $[t_k, t_{k+1}]$ is such that

$$N \int_{t_k}^{t_{k+1}} \rho(\tilde{x}(t), \xi(t)) \mathrm{d}t > W(\tilde{x}(t_k^+), \xi(t_k^+)) - W(\tilde{x}(t_k^-), \xi(t_k^-)). \tag{15}$$

Then the zero equilibrium of the closed-loop system is globally asymptotically stable, $x(t)$ is collision-free and such that $\lim_{t \to \infty} \tilde{x}(t) = 0$. ◇

Proof: First, note that if there are no jumps, *i.e.* if $\mathcal{HJ}_i \leq -\rho(\tilde{x}, \xi)$, $i = 1, \ldots, N$, along the closed-loop trajectories of the system, asymptotic stability follows directly from Lemma 1. Consider now the case in which jumps *do* occur. The left-hand sides of (8) can be written as quadratic forms in $[\tilde{x}^\top, (\tilde{x} - \xi)^\top]$ ([9], proof of Theorem 3), where, for fixed $\tilde{x}$, the quadratic terms in ξ are of the form $-\xi\Lambda_i\xi$, with $\Lambda_i > 0$, $i = 1, \ldots, N$. Thus, for any fixed $\tilde{x}$, the constraint (14b) can be satisfied by a sufficiently large ζ, *i.e.* a solution $\zeta^\star$ exists. Then, by continuity of ρ, it follows from (14b) and the definition of the flow set C in (11) that there is a certain *dwell time* between any two consecutive jumps. However, since the hard constraint (13c) is replaced with the soft constraint (14a), the hybrid control law (12)-(14) is such that the candidate Lyapunov function $W(x, \xi)$ *may* increase at jumps, *i.e.* considering the k-th jump, occurring at time t_k, the update in ξ may be such that $W(\tilde{x}(t_k^+), \xi(t_k^+)) > W(\tilde{x}(t_k^-), \xi(t_k^-))$.

Recalling that $W(\tilde{x}(t_{k+1}^-), \xi(t_{k+1}^-)) - W(\tilde{x}(t_k^+), \xi(t_k^+)) = \int_{t_k}^{t_{k+1}} \dot{W}(\tilde{x}(t), \xi(t))\mathrm{d}t$, and $\dot{W} = \sum_{i=1}^N \dot{V}_i \leq -N\rho(\tilde{x}, \xi) \leq 0$ during flows, clearly $W(\tilde{x}(t_{k+1}^-), \xi(t_{k+1}^-)) - W(\tilde{x}(t_k^+), \xi(t_k^+)) \leq -N\int_{t_k}^{t_{k+1}} \rho(\tilde{x}(t), \xi(t))\mathrm{d}t$. Then, by (15), $W(\tilde{x}(t_{k+1}^-), \xi(t_{k+1}^-)) < W(\tilde{x}(t_k^-), \xi(t_k^-))$ follows, *i.e.* any increase in W due to a jump is recovered during the following flow, hence implying global asymptotic stability of the origin for the closed-loop hybrid system (3)-(12). Moreover, the *dwell time* condition (15) rules out the possibility of Zeno behaviour, thus guaranteeing that each agent will asymptotically converge to its desired position. □
The combination of global asymptotic stability and the exclusion of Zeno behaviour, established in Theorem 1, ensures that the paths of each virtual agent are free of collisions *and* deadlocks and are such that each agent reaches its target.

Remark 1: The condition (15) ensures that the time between any two consecutive jumps is sufficiently large, such that any potential increase in W due to a jump is recovered during the flow immediately following such jump. This condition can be further relaxed by requiring that the *overall* increase in W due to jumps is recovered by the *overall* decrease in W during flows. Namely, (15) may be replaced by the requirement that there exists an unbounded sub-sequence of jump times indexed by $n_1 < n_2 < \dots$ as $\{t_{n_j}\}_{j=1}^{\infty}$, such that $\lim_{j\to\infty} t_{n_j} = \infty$ and

$$N\int_{t_{n_j}}^{t_{n_{j+1}}} \rho(\tilde{x}(t), \xi(t))\mathrm{d}t > W(\tilde{x}(t_{n_j}^+), \xi(t_{n_j}^+)) - W(\tilde{x}(t_{n_j}^-), \xi(t_{n_j}^-)) \,. \tag{16}$$

In practice it may be of interest to define a finite *window*, *i.e.* a certain number of jumps or a certain time interval, in which any increase due to jumps is required to be compensated by the decrease during flows. Such condition can be easily checked online to indicate desired or undesired behaviours. Moreover, if undesired behaviours are suspected, actions can be taken to drive the system away from a potential deadlock, *e.g.* by perturbing elements of $\xi(t)$. ▲

Theorem 1 represents a generalisation of [11]. By allowing the dynamic variable ξ to *reset* when the quantities $\mathcal{HJ}_i$ are not sufficiently negative, the hybrid control laws allow for the *expansion* of the region Ω in which Problem 1 is solved.

V. A DIGITAL CONTROL ALGORITHM FOR ONLINE COLLISION AVOIDANCE

A heuristic algorithm, in which the dwell time requirement (15) of Theorem 1 is further relaxed, suitable for *online* implementation, is presented in this section. The overall algorithm consists of two main ingredients corresponding to the *path planning* and *path tracking* blocks in Figure 1.

A. Algorithm 1: Online Path planning

The path planning algorithm, based on the main result of Theorem 1, is a hybrid system in which the state of the controller ξ is updated according to

$$\zeta^\star = \text{argmin}_\zeta W(\tilde{x}, \zeta)\,, \tag{17a}$$
$$\text{s.t.} \quad \max_i \{\mathcal{HJ}_i(\tilde{x}, \zeta)\} \leq -\mu\rho(\tilde{x}, \zeta)\,, \tag{17b}$$
$$\|u_i(\tilde{x}, \zeta)\| \leq \bar{u}\,, \tag{17c}$$

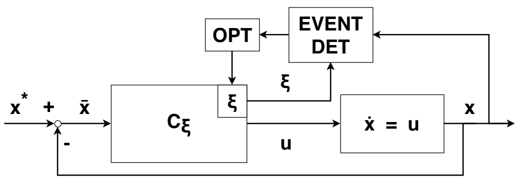

Fig. 2: Block diagram representation of Algorithm 1.

Algorithm 1: Online Path planning

```
   Input  : Set points $x_i^\star$, for $i = 1, \dots, N$
   Input  : Initial conditions $\xi(t_0)$, $\mu(t_0)$, $x_i(t_0)$, for
            $i = 1, \dots, N$
   Input  : Control and optimisation parameters, $\rho(\cdot,\cdot)$, $\mu$
   Input  : Sampling time $\Delta t$
   Output: Collision free trajectories
            $x_i : [t_0, t_0 + \Delta t] \to \mathbb{R}^2$
   Init   : $t_1 \leftarrow t_0$
1  loop:
2      try:
3          Calculate $x_i(\tau) = x_i(t_1) + \int_{t_1}^{\tau} u_i(t)dt$ and
            $\xi(\tau) = \xi(t_1) + \int_{t_1}^{\tau} \dot{\xi}(t)dt$, with $u_i$ and $\dot{\xi}$
            defined in (12a) and (12b), for $i = 1, \dots, N$,
            for $\tau \in [t_1, t_0 + \Delta t]$
4      catch $\max_i\{\mathcal{HJ}_i(x - x^*, \xi)\} \geq -\rho(x - x^*, \xi)$:
5          $t_1 \leftarrow \tau$
6          $\zeta^\star \leftarrow$ (17)
7          $\mu(t_1) \leftarrow$ (18)
8          $\xi(t_1) \leftarrow \zeta^\star$
           goto : loop
9      end
10 end
11 return $x_i : [t_0, t_0 + \Delta t] \to \mathbb{R}^2$, for $i = 1, \dots, N$.
```

whenever $(\tilde{x}, \xi) \in D$, defined in (11), where $\bar{u} > 0$. An event detector, used to indicate whether the state is in the jump set, triggers a reset signal which enables the jump. A block diagram representation of the path planning architecture is depicted in Figure 2. The hybrid controller C_ξ is driven by the error between the (online) generated path and the desired position and contributes, together with virtual single integrator dynamics, to the collision-free path planning. Discrete-time events of the controller are induced by specific values of the state of the controller and the computed path via an online optimisation strategy.

The constraint (17c) is introduced to avoid overly sharp turns in the generated path. While, in principle the addition of the constraint (17c) may influence the existence of solutions of the optimisation problem (17), in practice any bound $\bar{u}$ (even arbitrarily large) will generate a feasible path, since the unicycle can track any arbitrary "turn" by rotating about itself. For $\bar{u}$ sufficiently large a solution of the optimisation problem exists (as demonstrated in Theorem 1).

While in Theorem 1 any value of $\mu > 1$ is acceptable, in practice the magnitude of this parameter has a direct influence on the behaviour of the robots: a small μ will generally result in more resets than a large μ which, in contrast, could

```
Algorithm 2: Path tracking
  Input  : Target trajectory $x_i : [t_0, t_0 + \Delta t] \to \mathbb{R}^2$
  Input  : Sampling time $\Delta t$
  Output: Right and left velocity references $\omega_R^i$ and $\omega_L^i$
1 $x_{k-1} \leftarrow x_i(t_0)$
2 $x_k \leftarrow x_i(t_0 + \Delta t)$
3 $\Delta x_k \leftarrow \frac{(x_k - x_{k-1})}{\Delta t}$
4 Read robot current position $p$ and orientation $\theta$
5 Compute $\omega_R^i$ and $\omega_L^i$ as in (19)
6 return $\omega_R^i$, $\omega_L^i$
```

result in an empty admissible solution set of (17) for some states of the system. To strike a good balance between these two behaviours, we allow for μ to change dynamically. In particular, μ is updated according to the switching dynamics

$$\mu^+ = \begin{cases} \alpha^+ \mu, & \text{if } \exists \text{ a solution to (17)} \\ \alpha^- \mu, & \text{otherwise,} \end{cases} \tag{18}$$

where $\alpha^+ > 1$ and $\alpha^- < 1$. The trajectory planning block, *i.e.* the closed-loop system (3)-(12) (with ξ^+ and μ^+ updated according to (17) and (18), respectively), is implemented *online* according to Algorithm 1.

B. Algorithm 2: Path Tracking

The path tracking controllers use the output of Algorithm 1 to generate digital controllers based on a discretisation of (10b), in which $\dot{x}_i = \begin{bmatrix} \dot{x}_i^1 & \dot{x}_i^2 \end{bmatrix}^\top$ is replaced by the approximation $\Delta x_i = \dfrac{(x_i(t_0 + \Delta t) - x_i(t_0))}{\Delta t}$, for $i = 1, \ldots, N$, where t_0 denotes the previous time step and Δt denotes the sampling time. The resulting control inputs are then mapped to the speeds of the left and right wheels of each WMR (denoted by ω_L^i and ω_R^i, respectively, for $i = 1, \ldots, N$) according to

$$\begin{bmatrix} \tilde{v}_i \\ \tilde{\omega}_i \end{bmatrix} = \begin{bmatrix} \text{sat}_a(v_i) \\ \text{sat}_b(\omega_i) \end{bmatrix}, \quad \begin{bmatrix} \omega_L^i \\ \omega_R^i \end{bmatrix} = \begin{bmatrix} \dfrac{2\tilde{v}_i - L\tilde{\omega}_i}{2r} \\ \dfrac{2\tilde{v}_i + L\tilde{\omega}_i}{2r} \end{bmatrix}, \tag{19}$$

$i = 1, \ldots, N$, where r denotes the radius of each wheel and L is the distance between the wheels, for $i = 1, \ldots, N$. A saturation has been applied in (19) to reflect the physical constraints on the inputs of each WMR. Due to the saturation these control laws deviate from the result of Theorem 1. Nonetheless, it can be seen that there exists $\eta^\star \in (0, 1]$, such that by time-scaling (10) by a factor η, common to all the agents, the inputs (19) with v_i and ω_i replaced by the time-scaled output of (10) ensures path tracking regardless of the presence of the saturation in $\tilde{v}_i$ and $\tilde{\omega}_i$, for all $\eta \leq \eta^*$. In a real-time implementation such value can be computed online once any robot is *close* to saturation and broadcast to the entire team of WMRs. A block diagram representation of a path tracking controller is shown in Figure 3, where C_{FL} represents the digital control law (10b)-(19). The path tracking controllers for each WMR are implemented online according to Algorithm 2.

C. Function and parameter selections

The functions and parameters associated with Algorithms 1 and 2 used in the the remainder of this paper are specified here. The collision avoidance functions are set as $g_i^s(\tilde{x}) \triangleq \sum_{j=1}^m \delta_{-1}(G_i^s(\tilde{x}, p_j))\, G_i^s(\tilde{x}, p_j) + \delta_{-1}(-G_i^s(\tilde{x}, p_j))\, M$, where $G_i^s(\tilde{x}, p_j) \triangleq (\|\tilde{x}_i + x_i^\star - p_j\|_{E_j}^2 - \rho_j^2)^{-3}$, and $g_i^d(\tilde{x}) \triangleq \sum_{j=1, j\neq i}^N \delta_{-1}(G_{ij}^d(\tilde{x}))\, G_{ij}^d(\tilde{x}) + \delta_{-1}(-G_{ij}^d(\tilde{x}))\, M$, where $G_{ij}^d(\tilde{x}) \triangleq (\|\tilde{x}_i + x_i^\star - (\tilde{x}_j + x_j^\star)\|^2 - (r_i + r_j)^2)^{-3}$, with $M = 10^5$. This particular selection, where the first terms of g_i^s and g_i^d provide *barriers* and the second terms associate a high cost to regions *inside* the barriers, is made to avoid numerical issues associated with step 3 of Algorithm 1 due to *overly steep* barriers. Note that in the *flow set* C, both g_i^s and g_i^d are continuous. The function ρ represents a trade-off between the number of *jumps* in the path generation and the allowed *flowing time*. Noting that $\mathcal{HJ}_i$, $i = 1, \ldots, N$, have a quadratic rate of convergence close to the origin of $[\tilde{x}, (\tilde{x} - \xi)]$, to avoid excessive reset signals from the event detector in Algorithm 1 when the robots are close to their targets, we design ρ to converge faster than $\mathcal{HJ}_i$, $i = 1, \ldots, N$, close to the origin. Namely, we consider ρ given by

$$\rho(\tilde{x}, \xi) = \begin{cases} \exp\left(\dfrac{-2}{\|(\tilde{x}, \tilde{x} - \xi)\|^2} \right), & (\tilde{x}, \tilde{x} - \xi) \neq 0 \\ 0\,, & (\tilde{x}, \tilde{x} - \xi) = 0\,. \end{cases} \tag{20}$$

Note that this selection is such that far from the origin ρ is bounded and converges to a fixed value, *i.e.* 1.

The parameters for the hybrid control law given by (12), the optimisation problem (17) and the discrete update equation (18), as well as the parameters for the control law (10), $i = 1, \ldots, N$, used in the remainder of this paper are shown in Table I below.

VI. Simulations and Experimental results

The efficacy of Algorithms 1 and 2 is demonstrated via a simulation and an experiment presented in the following subsections (see supplementary material for videos of these and additional simulations and experiments).

A. Path planning: an example involving 10 agents

Consider 10 virtual agents with initial positions, targets and $\xi(0) = [\xi_1(0), \ldots, \xi_{10}(0)]^\top$ specified in Table II overleaf, $\mu(0) = 1000$, and the remaining parameters of Algorithm 1 given in Table I.To demonstrate the improved performance of the proposed hybrid scheme with respect to the result recalled in Lemma 1, simulations have been run using both the hybrid controller, *i.e.* Algorihm 1, and the continuous controller (9), with the same parameters. The trajectories corresponding to the hybrid controller are given by the solid lines in Figure 4, where crosses indicate the targets and the large circles are centred at the agents' final positions. While the hybrid controller yields collision-free paths, in this scenario the initial conditions are such that the continuous-time controller (9) results in collisions, with the first collision occurring between agents 3

α^+	α^-	R_i	k_i	β_i^s	β_i^d	α_i	γ_i	κ	$\bar{u}$
1.1	0.7	$1.5I_2$	1	20	20	0.5	0.3	15	30

TABLE I: Parameters defining the path planning controller and the path tracking controllers.

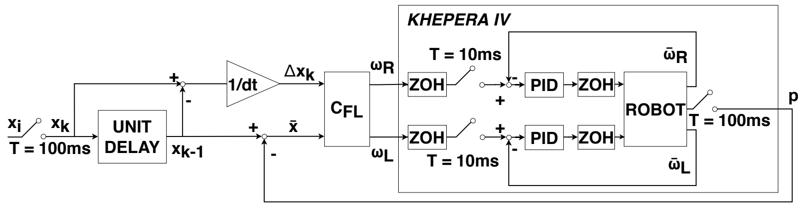

Fig. 3: Block diagram representation of Algorithm 2 for one WMR.

i	1	2	3	4	5	6	7	8	9	10
$x_i(0)^\top$	$[0,-80]$	$[40,30]$	$[-50,50]$	$[50,-50]$	$[-20,80]$	$[-60,-60]$	$[100,-100]$	$[-100,100]$	$[100,100]$	$[-100,-100]$
$x_i^{*\top}$	$[40,30]$	$[50,-50]$	$[0,-80]$	$[-50,50]$	$[-60,-60]$	$[-20,80]$	$[-100,100]$	$[100,-100]$	$[20,70]$	$[-30,-70]$
$\xi_i(0)^\top$	$[100,-50]$	$[310,22]$	$[250,-20]$	$[22,0]$	$[-300,250]$	$[50,50]$	$[-1300,-500]$	$[1300,500]$	$[0,0]$	$[0,0]$

TABLE II: Initial conditions and targets corresponding to the simulation described in Section VI-A.

and 4. The paths of agents 3 and 4 up to the first collision are indicated by the dotted, black lines in Figure 4, where it can be clearly appreciated that the hybrid trajectories (shown in green and magenta) instead maneuver in such a way that avoids collision. The time histories of W (top) and $\max_i \mathcal{HJ}_i$ (bottom) corresponding to the hybrid (solid, black line) and the continuous (dashed, orange line) controllers are shown in Figure 5. The time history of $-\rho$ (red, solid line) is shown in the bottom plot of Figure 5. Considering the continuous controller, (8) are clearly violated – resulting in collisions. Considering the hybrid controller, on the other hand, it is clear that the state-driven jumps result in that $\max_i \mathcal{HJ}_i$ decreases at jumps such that $\max_i \mathcal{HJ}_i \leq -\rho$ at all times and, while W increases at some jumps, it experiences an overall decrease over time in accordance with Remark 1.

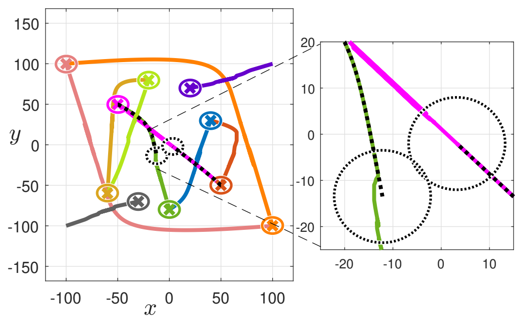

Fig. 4: Collision free paths for 10 virtual agents.

B. Experimental results: two agents exchanging positions

An experiment has been carried out using two Khepera IV mobile robots. The core code is implemented in Matlab 2018b, running on a laptop with an Intel i3-6100U processor, 8GB RAM and on Microsoft Windows 10 Pro (x64). The positions of the robots are measured using OptiTack V120: Trio, connected via USB to the laptop where the proprietary software Motive (v.1.10.3) isused to compute both the position and the orientation of the robots with respect to an inertial frame of reference. The position and orientation of every robot are then recovered from Motive by a dedicated Matlab script through a custom interface based on the NatNet SDK (provided by

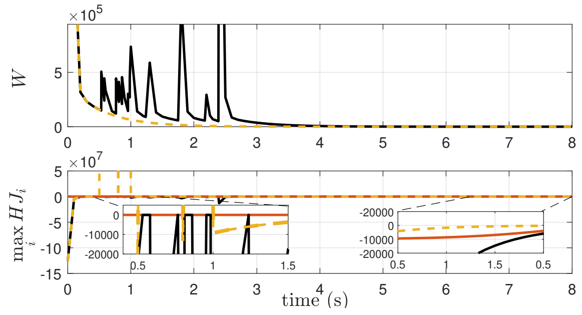

Fig. 5: The time histories of W (top) and $\max_i \mathcal{HJ}_i$ (bottom) with the continuous (orange, dashed line) and hybrid controllers (black, solid line) and $-\rho$ (bottom: red, solid line).

OptiTrack). Steps 3 and 4 of Algorithm 1, and in particular the solution to the underlying hybrid dynamical systems, are implemented using "ode45" together with its event detection mechanism. The optimisation in step 6 of Algorithm 1 has been implemented using the "sqp" optimisation algorithm provided by the "fmincon" solver. These particularly time-consuming tasks were usually finished before the end of the sampling time $\Delta t = 100$ms. Whenever these tasks lasted longer than Δt, the last available output of Algorithm 1 - *i.e.* the current points to be tracked by the robots - were provided to the *path tracking* module, namely Algorithm 2. Algorithm 2 consists of a Matlab routine scheduled to be executed every $\Delta t = 100$ms. First the *navigation* data, *i.e.* the position and orientation of each robot, is fetched by Motive, then the speed references for the right and left wheels of every robot are calculated as in (19), with $a = 25$, $b = 5$, $L = 10.54$cm and $r = 2.1$cm (in accordance with the Kepera IV User Manual), using the output of the *trajectory planning* module. The reference velocities are then sent through a serial bluetooth communication to the robots, where the proprietary program *kh4server* implements a PID controller (with sampling time 10ms) to track the velocity reference of each wheel.

Consider the scenario in which two WMR are to exchange positions - a highly symmetric situation which can easily lead to deadlock. The virtual agents' initial conditions are $x_1(0) = [-27,-27]^\top$, $x_2(0) = [27,27]^\top$. The initial positions and orientations of the WMRs are $p_1(0) = [-29.3642,-26.7597]^\top$,

$p_2(0) = [29.3461, 26.6896]^\top$, $\theta_1(0) = -164.8°$ and $\theta_2(0) = 15.3°$. The parameters defining Algorithms 1 and 2 are as defined in Table I and $\mu(0) = 200$ and $\xi(0) = [-98.1276, -88.3148, 98.1276, 88.3148]^\top$ (which is the solution of the optimisation problem (17) with $\tilde{x} = \tilde{x}(0)$). The planned paths for the first (solid, black line) and second (solid, grey line) WMRs as well as the actual trajectories of the first (dotted, teal line) and second (dotted, orange line) WMRs are shown in Figure 6. The large circles are centred at the final positions of the first (teal) and second (orange) virtual agent and indicate their safety radii, whereas the smaller circles and triangles indicate the final positions and orientations of the first (teal) and second (orange) WMRs, respectively. The arrows indicate the directions of travel and the crosses indicate the targets. The time histories of the control inputs of the first WMR, *i.e.* ω_R^1 (solid, blue line) and ω_L^1 (dotted, red line), and of the second WMR, *i.e.* ω_R^2 (solid, blue line), ω_L^2 (dotted, red line), are shown in the top and bottom graphs of Figure 7, respectively. No time-scaling is applied in the experiments. On certain occasions saturation does occur (according to (19)); the successful experiments demonstrate the benefits of the intrinsic robustness properties coming from the Lyapunov-based arguments in Theorem 1.

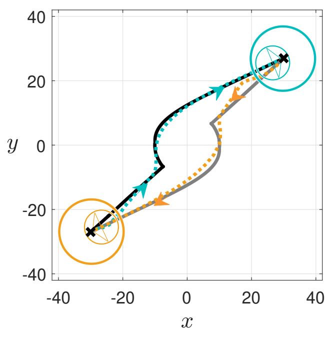

Fig. 6: The paths of x_1 (solid, black line) and x_2 (solid, grey line), and the corresponding trajectories p_1 (dotted, teal line) and p_2 (dotted, orange line).

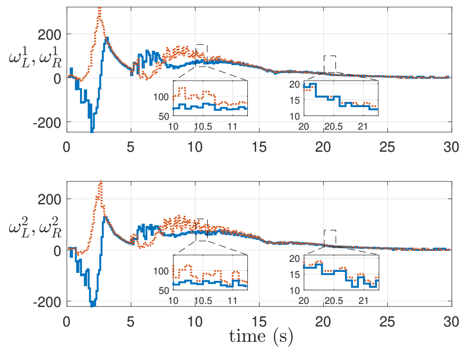

Fig. 7: The time histories of ω_R^i (solid, blue line) and ω_R^i (dotted, red line) for $i = 1$ (top) and $i = 2$ (bottom).

VII. Conclusion

In this paper we propose a novel approach, based on a differential game formulation and a hybrid controller, for solving the multi-agent collision avoidance problem *online*. The resulting control scheme ensures, subject to a certain condition, that the robots *globally* converge to their desired targets. Moreover, when this condition is not satisfied, it provides a measure to indicate any potential undesired behaviour (such as deadlocks). Based on the developed control scheme we provide an algorithm whose efficacy is demonstrated both through simulations and experimentally. Directions for future work include considerations of distributed settings, which will lead to improvements in terms of the scalability of the approach.

References

[1] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[2] T. Mylvaganam and A. Astolfi, "Approximate optimal monitoring: preliminary results," in *American Control Conference*, 2012.

[3] ——, "Approximate optimal monitoring," in *14th European Control Conference*, 2014.

[4] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[5] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.

[6] T. Mylvaganam and A. Astolfi, "A differential game approach to formation control for a team of agents with one leader," in *American Control Conference*, 2015, pp. 1469–1474.

[7] D. Cappello and T. Mylvaganam, "Distributed control of multi-agent systems via linear quadratic differential games with partial information," in *57th IEEE Conference on Decision and Control*, 2018.

[8] ——, "A game theoretic framework for distributed control of multi-agent systems with acyclic communication topologies," in *58th IEEE Conference on Decision and Control*, 2019.

[9] T. Mylvaganam, M. Sassano, and A. Astolfi, "Constructive ϵ-Nash equilibria for nonzero-sum differential games," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 950–965, 2015.

[10] M. Sassano, T. Mylvaganam, and A. Astolfi, "An algebraic approach to dynamic optimisation of nonlinear systems: a survey and some new results," *Journal of Control and Decision*, vol. 6, no. 1, pp. 1–29, 2019.

[11] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, 2017.

[12] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[13] H. G. Tanner and A. Kumar, "Towards decentralization of multi-robot navigation functions." in *IEEE International Conference on Robotics and Automation*, vol. 4, 2005, p. 4132.

[14] H. G. Tanner and A. Boddu, "Multiagent navigation functions revisited," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1346–1359, 2012.

[15] C. S. Karagoz, H. I. Bozma, and D. E. Koditschek, "Coordinated navigation of multiple independent disk-shaped robots," *IEEE Transactions on Robotics*, vol. 30, no. 6, 2014.

[16] J. van den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research. Springer Tracts in Advanced Robotics*, vol. 70, 2011.

[17] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[18] G. Leitmann and J. Skowronski, "Avoidance control," *Journal of Optimization Theory and Applications*, vol. 23, no. 4, pp. 581–591, 1977.

[19] G. Leitmann, "Guaranteed avoidance strategies," *Journal of Optimization Theory and Applications*, vol. 32, no. 4, pp. 569–576, 1980.

[20] T. Mylvaganam and M. Sassano, "Autonomous collision avoidance for wheeled mobile robots using a differential game approach," *European Journal of Control*, vol. 40, pp. 53 – 61, 2018.

[21] T. Mylvaganam, C. Possieri, and M. Sassano, "Global stabilization of nonlinear systems via hybrid implementation of continuous-time local controllers," *Automatica*, vol. 106, pp. 401–405, 2019.