Cooperative collision avoidance for nonholonomic robots

Alonso-Mora, Javier; Beardsley, Paul; Siegwart, Roland

DOI
10.1109/TRO.2018.2793890

Publication date
2018

Document Version
Final published version

Published in
IEEE Transactions on Robotics

Cooperative Collision Avoidance for Nonholonomic Robots

Javier Alonso-Mora, *Member, IEEE*, Paul Beardsley, and Roland Siegwart, *Fellow, IEEE*

***Abstract*—In this paper, we present a method, namely ϵCCA, for collision avoidance in dynamic environments among interacting agents, such as other robots or humans. Given a preferred motion by a global planner or driver, the method computes a collision-free local motion for a short time horizon, which respects the actuator constraints and allows for smooth and safe control. The method builds on the concept of reciprocal velocity obstacles and extends it to respect the kinodynamic constraints of the robot and account for a grid-based map representation of the environment. The method is best suited for large multirobot settings, including heterogeneous teams of robots, in which computational complexity is of paramount importance and the robots interact with one another. In particular, we consider a set of motion primitives for the robot and solve an optimization in the space of control velocities with additional constraints. Additionally, we propose a cooperative approach to compute safe velocity partitions in the distributed case. We describe several instances of the method for distributed and centralized operation and formulated both as convex and nonconvex optimizations. We compare the different variants and describe the benefits and tradeoffs both theoretically and in extensive experiments with various robotic platforms: robotic wheelchairs, robotic boats, humanoid robots, small unicycle robots, and simulated cars.**

***Index Terms*—Autonomous robots, collision avoidance, motion planning, multi-robot systems, robot control, robot motion.**

I. Introduction

SUCCESSFUL robot operation builds on at least three interconnected competence, namely, localization, mapping, and motion planning/control. The latter is concerned with computing a (lowest cost) path or trajectory between two configurations embedded in a cost field, while taking into account motion constraints, static obstacles, and moving obstacles [1]–[3]. The important case where moving obstacles are decision-making agents forms the topic of this paper. In particular, we describe a method, ε-cooperative collision avoidance (εCCA), for collision avoidance, which respects the kinodynamic constraints of the robot and accounts for the cooperation with other robots in avoiding collisions. The method is especially well suited for multirobot systems.

A. Related Works

Advances in deterministic graph search [4], graph representation [5], and randomized sampling-based methods [6] have enabled (approximately optimal) solution strategies on a global scale including obstacles, which led to a separation between global and local planning. The development of sampling-based planning algorithms [7], [8] and tree-based approaches [9] has considerably softened this separation and enabled unified system-compliant online motion planning. Nonetheless, if this operation takes place among other decision-making agents, system-compliant planning alone does not seem adequate to ensure safe navigation. Rather, it becomes important that the individual planning strategies are aware of (and take into account) that other agents are also engaging in a similar activity. This is the case for multirobot systems that are the focus of our paper.

In the context of collision avoidance for multiple robots, similar approaches to those for the single-robot case can be applied. However, the increase in robot density and collaborative interaction requires methods that scale well with the number of robots, while avoiding collisions, as well as oscillations. Decentralized control helps to lower computational cost and introduces additional robustness and flexibility to the multirobot system. Traditional approaches for collision avoidance are potential fields [10], the dynamic window [11], inevitable collision states [12], sequential convex programming [13], model predictive control [14], priority-based planning [15], and social forces [16]. Yet, they do not account for the interaction between robots that appears in multirobot systems, or when robots navigate among other decision-making agents. Recent methods for fast collision avoidance in multirobot scenarios include buffered Voronoi cells [17] and barrier certificates [18]. Our method resembles the latter in that we also employ a quadratic optimization to compute collision-free motions. Interaction can be taken into account by learning-based methods, such as Gaussian processes [19] and inverse reinforcement learning [20], [21]. Our approach provides formal guarantees and is best suited for multirobot scenarios, thanks to its low computational cost.

Manuscript received March 23, 2017; revised October 13, 2017; accepted December 23, 2017. Date of publication March 22, 2018; date of current version April 12, 2018.This paper was recommended for publication by Associate Editor K. Hauser and Editor C. Torras upon evaluation of the reviewers' comments. *(Corresponding author: Javier Alonso-Mora.)*

J. Alonso-Mora is with the Department of Cognitive Robotics, Delft University of Technology, Delft 2628 CD, The Netherlands (e-mail: j.alonsomora@tudelft.nl).

P. Beardsley is with Disney Research Zurich, Zurich 8092, Switzerland (e-mail: pab@disneyresearch.com).

R. Siegwart is with the Autonomous Systems Laboratory, ETH Zurich, Zurich 8092, Switzerland (e-mail: rsigwart@ethz.ch).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The material consists of a video, viewable with most players, such as VLC, containing experiments with several robotic platforms (e-puck robots, small boats, humanoid robots and simulated cars). The accompanying video is available at https://youtu.be/WqSW3Soi0vM. Contact j.alonsomora@tudelft.nl for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2018.2793890

The reciprocal velocity obstacle (RVO) method [22] models robot interaction both in a decentralized manner and pairwise optimally. Under the assumption that other agents also continue their present motion along a straight-line trajectory, future collisions may be estimated as a function of relative velocity alone. Its success paved the road toward several extensions and revisions of the basic framework: The optimal reciprocal collision-avoidance (ORCA) method [23] prevents reciprocal dances and casts the problem into a linear programming framework, which can be solved efficiently. Snape *et al.* [24] accounted for simple robot kinematics and sensor uncertainty by enlarging the velocity cones, without formal guarantees. Wilkie *et al.* [25] generalized RVO for robots with nonholonomic constraints by testing sampled controls for their optimality, which required extensive numeric computation and relied on probabilistic sampling. Solutions limited to robots with unicycle and bicycle kinematics are introduced in [26]–[28] and [29], generalizations of the RVO method to second-order and *n*th-order integrator dynamics are described in [30] and [31], and generalization of the RVO method to heterogeneous teams of robots are presented in [32], yet it required all robots to be controlled with the same type of inputs. The extension RRVO [33] applies to polygonal robots and COCALU [34] accounts for uncertainty in the measurements. Extensions to aerial vehicles navigating in three-dimensional spaces have also been proposed, which rely on LQG obstacles [35] and LQR control [36]. The latter follows the same concept of motion constraints as this paper.

Despite the large body of related contributions, we note that most of the extensions of the RVO method are limited toward a specific vehicle model, or a specific order of solution continuity, or only apply to homogeneous teams of robots.

B. Contribution

We present a method, namely ε CCA, for collision-free navigation among, homogeneous or heterogeneous, groups of decision-making robots. The main contributions are as follows.

1) A method for cooperative collision avoidance, which accounts for the interaction with other robots and the kinematic model and dynamic constraints of the robots.
2) A detailed discussion of convex and nonconvex, centralized, and distributed implementations of the method.
3) An extension of the method to cooperative distributed collision avoidance where a prediction over future velocities of the neighboring agents can be taken into account in the computation of the velocity partition.
4) Extensive experimental evaluation of the proposed algorithms, including various robot kinematics, such as differential-drive, carlike, and boats.

This paper describes a method for planar robots that unifies and generalizes our previous conference contributions [28], [29], [37], and [38]. Additionally, we introduce the extension to cooperative avoidance with a velocity prediction and show new experimental results with boats and wheelchairs. Not discussed in this paper, our method also extends to aerial vehicles [36].

C. Organization

The remaining part of this paper is organized as follows. In Section II, we introduce the required definitions and problem formulation. In Section III, we describe the motion constraints that arise from the robot kinodynamic model. In Section IV, we describe the collision avoidance constraints. In Section V, we describe the εCCA method in detail. In Section VI, we present experimental results with various robotic platforms followed by a discussion of lessons learned. Finally, Section VII concludes this paper.

II. Preliminaries

We now provide the needed definitions, the problem formulation for cooperative avoidance, and an overview of the method. Throughout this paper, vectors are denoted in bold $\mathbf{x}$, matrices in capital M, and sets in mathcal $\mathscr{X}$. The Minkowsky sum of two sets is denoted by $\mathscr{X} \oplus \mathscr{Y}$ and $x = ||\mathbf{x}||$ denotes the Euclidean norm of vector $\mathbf{x}$. The super index $\cdot^k$ indicates the value at time t^k, and the appropriate relative time $\tilde{t} = t - t^k$. Subindex $\cdot_i$ indicates agent i and relative vectors are denoted by $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. For ease of exposition, no distinction is made between estimated and real states.

1) Agents, i.e., Robots and Dynamic Obstacles: Consider a set of n agents, where one or more are controlled robots within the system. All other agents are dynamic obstacles. In the remainder of this paper, and for simplicity, we refer to all agents as robots, which move on the plane $\mathscr{W} = \mathbb{R}^2$. For each robot $i \in \mathscr{I} = \{1, \ldots, n\} \subset \mathbb{N}$, its position at time t is denoted by $\mathbf{p}_i(t) \in \mathbb{R}^2$, and its state, which may contain also its velocity $\mathbf{v}_i(t) = \dot{\mathbf{p}}_i(t)$ and acceleration $\mathbf{a}_i(t) = \dot{\mathbf{v}}_i(t)$, is denoted by $\mathbf{z}_i(t)$. We model all robots by the smallest enclosing disk (the method also applies to arbitrary robot shapes with the assumption that they do not rotate during the local planning horizon) of radius r_i. Let $D(\mathbf{p}, r)$ denote a disk centered at position p and of radius r. Further denote the area occupied by robot i at position $\mathbf{p}_i$ by $\mathscr{A}_i(\mathbf{p}_i) = D(\mathbf{p}_i, r_i) \subset \mathbb{R}^2$.

2) Static Obstacles: Consider a set of static obstacles $\mathscr{O} \subset \mathbb{R}^2$ defining the global map. Further denote by $\bar{\mathscr{O}}_r$ the set $\mathscr{O}$ dilated by radius r, i.e., the positions p for which a robot of size $D(\mathbf{p}, r)$ would be in collision with any of the obstacles, formally

$$\bar{\mathscr{O}}_r = \{\mathbf{p} \in \mathbb{R}^2 \quad | \quad D(\mathbf{p}, r) \cap \mathscr{O} \neq \emptyset\}. \tag{1}$$

The set of dilated obstacles $\bar{\mathscr{O}}_r$ can be stored in an occupancy grid map. Alternatively, they can be stored as a list of polytopes. In both cases, given a position $\mathbf{p}$, it is possible to compute a convex polygon $P(\mathbf{p}, r)$ in its neighborhood, which is fully contained in free space $P(\mathbf{p}, r) \subset \mathscr{W} \setminus \bar{\mathscr{O}}_r$, for example, with the method of [39].

3) Time: We denote by the current time t_0 and by the time horizon τ of the motion planner. Further, denote $t_1 = t_0 + \tau$. We may employ $\mathbf{z}_i^0 = \mathbf{z}_i(t_0)$ and $\mathbf{p}_i^0 = \mathbf{p}_i(t_0)$ for ease of exposition.

4) Preferred Velocity: For each controlled robot, we assume that a *preferred velocity* $\bar{\mathbf{u}}_i \in \mathbb{R}^2$ is available at time t_0. This preferred velocity is typically given by a global guidance system. To converge to a desired goal location, the preferred velocity can be given by a proportional controller, saturated at a preferred

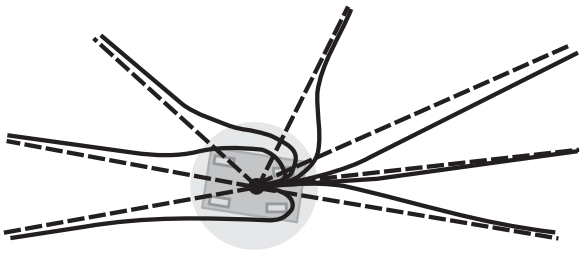

Fig. 1. Example of motions from the set of primitives. Each kinodynamically-feasible trajectory (continuous line, $(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$) is given by a trajectory tracking controller toward the straight-line reference of constant control velocity (dashed, $\mathbf{p}(t_0) + \mathbf{u}(t - t_0)$).

speed [28]. To follow a predefined path, the preferred velocity can be given by a trajectory tracking controller [29], such as the one given in [40]. For shared control of a robot, the preferred velocity can also be given by a human driver, e.g., proportional to the driving wheel or joystick position in the relative frame of the vehicle [38].

A. Motion Primitives

To model the kinematic and dynamic constraints of the robots, we employ a continuous set of motion primitives, see Fig. 1. The trajectory of the robot is given by a bijective mapping from a *control velocity* $\mathbf{u} \in \mathbb{R}^2$, which indicates the target direction and velocity of the robot.

A *control velocity* $\mathbf{u}$ defines a constant-velocity reference, which starts at the current position $\mathbf{p}(t_0)$ of the robot

$$\mathbf{p}_{\text{ref}}(t) = \mathbf{p}(t_0) + (t - t_0)\mathbf{u}, \text{ for } t \geq t_0. \tag{2}$$

The associated trajectory for the robot is given by an appropriate tracking controller $\mathbf{p}(t) = f(\mathbf{z}(t_0), \mathbf{u}, t)$, which respects the kinematic and dynamic constraints of the robot. This trajectory shall be continuous in the initial state $\mathbf{z}(t_0) = [\mathbf{p}(t_0), \dot{\mathbf{p}}(t_0), \ddot{\mathbf{p}}(t_0), \dots]$ of the robot and converge to the control reference of (2). Examples for typical robot kinematics are discussed in Section III.

B. Velocity Utility

For cooperative collision avoidance, we consider that a utility distribution $\phi_i : \mathbb{R}^2 \to \mathbb{R}^+$ over the control velocities of each robot is available.[1] Given the environment characteristics, for each robot, we can compute a cost function χ_i

$$\begin{aligned} \chi_i : \mathbb{R}^2 &\to [0, \infty) \\ \mathbf{u}_i &\mapsto \chi_i(\mathbf{u}_i) \end{aligned} \tag{3}$$

where the cost $\chi_i(\mathbf{u}_i)$ is given by the sum of several terms, which may include the following, with $K_* > 0$ weights.

1) *Deviation from current velocity:* The robot is expected to continue with its current velocity, $K_1||\mathbf{u}_i - \mathbf{v}_i(t_0)||^2$.
2) *Static obstacles:* A high cost $K_4 >> 0$ is added for reference trajectories in collision with a static obstacle, such that $\mathbf{p}_i(t_1) + \mathbf{u}_i\tilde{t} \in \mathscr{O}_r$ for some $\tilde{t} \in [0, \tau]$.
3) *Deviation from preferred velocity:* Velocities close to its preferred velocity incur lower cost, $K_2||\mathbf{u}_i - \bar{\mathbf{u}}_i||^2$.
4) *Cost to go:* Proportional to the distance from the robot's goal position to the expected position after the time horizon of the local planner, $\mathbf{p}_i(t_1) = \mathbf{p}_i(t_0) + \mathbf{u}_i\tau$. The cost to go can be computed with a search algorithm such as A* [3] or be obtained from a precomputed distance transform map.

The first and second cost terms can be obtained by all robots with common observations. If the goal positions are not shared, for other robots, the third and fourth cost terms can only be estimated. Additional cost terms, e.g., to account for congestion, could be included.

The cost function is then transformed into a utility distribution with a mapping $[0, \infty) \to [0, 1]$. To maintain the linear characteristics for low cost, we define

$$\phi_i(\mathbf{u}_i) = \frac{\max(0, \quad 1 - \chi_i(\mathbf{u}_i)/K_0)}{\int_{\mathbf{u}_i \in \mathbb{R}^2} \max(0, \quad 1 - \chi_i(\mathbf{u}_i)/K_0)} \tag{4}$$

where K_0 is the cutoff cost. An example this velocity utility is shown in Fig. 6. If a cost function is not available, the method can be employed with a reciprocal or constant velocity assumption, see Sec. IV-C.

C. Problem Formulation

The objective of this paper is to compute collision-free trajectories for a time horizon τ.

Definition 1 (Collision): A robot i at position $\mathbf{p}_i$ is in collision with a static obstacle, if $\mathscr{A}_i(\mathbf{p}_i) \cap \mathscr{O} \neq \emptyset$. The robot is in collision with a dynamic obstacle j at position $\mathbf{p}_j$ and of area $\mathscr{A}_j(\mathbf{p}_j)$ if $\mathscr{A}_i(\mathbf{p}_i) \cap \mathscr{A}_j(\mathbf{p}_j) \neq \emptyset$.

Definition 2 (Collision-free motion): A trajectory is said to be collision-free if for all times between t_0 and t_1 there is no collision between the robot and any static or dynamic obstacle

$$\mathscr{A}_i(\mathbf{p}_i(t)) \cap \left(\mathscr{O} \underset{j \in \mathscr{I} \setminus \{1\}}{\cup} \mathscr{A}_j(\mathbf{p}_j(t)) \right) = \emptyset \qquad \forall t \in [t_0, t_1] \tag{5}$$

which is equivalent to $\mathbf{p}_i(t) \subset \mathscr{W} \setminus \bar{\mathscr{O}}_{r_i}$ and $\mathscr{A}_i(\mathbf{p}_i(t)) \cap \mathscr{A}_j(\mathbf{p}_j(t)) = \emptyset$ for all $t \in [t_0, t_1]$ and $\forall j \in \mathscr{I} \setminus \{i\}$.

In this paper, we discuss two scenarios, a *centralized* one where a central unit computes the motion for all the robots simultaneously, and a *distributed* one where each robot computes independently its motion. For the latter, we assume no communication between the robots.

Problem 1 (Centralized collision avoidance): Consider the set of n robots and a centralized computing unit that controls all the robots in the team. Given the kinodynamic model of each robot in the team, compute collision-free trajectories for all the robots in the team. The trajectories shall minimize the sum over all robots of the deviation from a preferred direction of motion and speed, given by the cost function of (20). This problem will be formalized in Section V-A.

Problem 2 (Distributed collision avoidance): For robot $i \in \mathscr{I}$, and given its kinodynamic model and the current position and velocity of all neighboring agents and obstacles, compute a collision-free trajectory for the robot under the as-

[1] In Section IV-C, we show that the constant velocity model can be written as a particular case of this utility distribution.

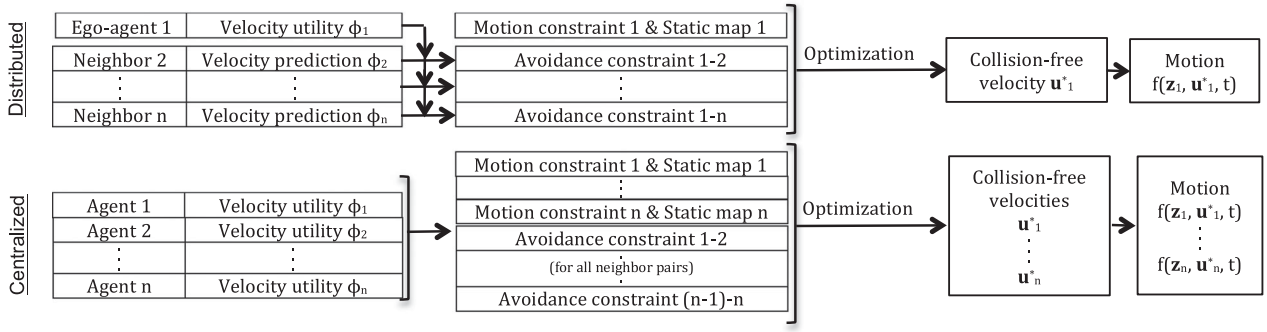

Fig. 2. Local motion planning. Overview of the approaches for distributed and centralized collision avoidance.

sumption that all other agents follow the same algorithm for collision avoidance or their velocity remains constant during the planning horizon. The trajectory shall minimize the deviation from a preferred direction of motion and speed, given by the cost function of (19). This problem will be formalized in Section V-B.

D. Approach

Fig. 2 shows a schema of our method for collision avoidance. For each controlled robot, we assume that a *preferred velocity* $\bar{\mathbf{u}}_i \in \mathbb{R}^2$ is available at time t_0. Since the optimal velocity resulting from a velocity obstacle (VO)-based constrained optimization would lead to an avoidance maneuver where robots get infinitely close, we add a repulsive term $\mathring{\mathbf{u}}_i$ to the preferred velocity $\bar{\mathbf{u}}_i$ to slightly push the robot away from static and moving obstacles. This term is inversely proportional to the distance to the closest obstacle, for example,

$$\mathring{\mathbf{u}}_i = \max\left(0, K_{r_{\max}}\left(1 - \frac{d_{\text{obst}} - r_i}{K_{r_{\text{dist}}}}\right)\right)\frac{\mathbf{p}_i - \mathbf{p}_{\text{obst}}}{d_{\text{obst}}} \tag{6}$$

where $d_{\text{obst}} = ||\mathbf{p}_i - \mathbf{p}_{\text{obst}}||$ is the distance from the center of the robot to the closest obstacle, $\mathbf{p}_{\text{obst}}$ is the closest point on that obstacle, $K_{r_{\max}}$ is the maximum repulsive velocity and $K_{r_{\text{dist}}}$ is the distance from which a repulsive velocity is added. Adding this repulsive velocity to the preferred velocity can help the planner in finding feasible collision-free trajectories since it aims at keeping a separation from obstacles, but, just by itself, it does not guarantee collision-free motion. In the remainder of this paper, we will consider $\bar{\mathbf{u}}_i := \bar{\mathbf{u}}_i + \mathring{\mathbf{u}}_i$.

In our method, we consider three types of constraints.

1) For avoidance of static obstacles.
2) For avoidance of other robots or agents, via the velocity obstacles [41] and the ORCA paradigm [23].
3) For respecting the kinodynamic model of the robot, formulated as a constraint that limits the set of motion primitives.

Given the preferred velocity $\bar{\mathbf{u}}_i$ and the set of constraints, an optimal control velocity $\mathbf{u}_i^*$ is computed, which minimizes the deviation with respect to $\bar{\mathbf{u}}_i$ and such that its associated local trajectory $f(\mathbf{z}_i(t_0), \mathbf{u}_i^*, t)$ is collision-free. An advantage of our method is that it allows for an efficient optimization in the control velocity space ($\mathbb{R}^2$) to achieve collision-free motions. This is achieved by enlarging the radius of the robots by a small value ε and restricting the motion primitives to those with an error below ε, see Fig. 3 and the following section.

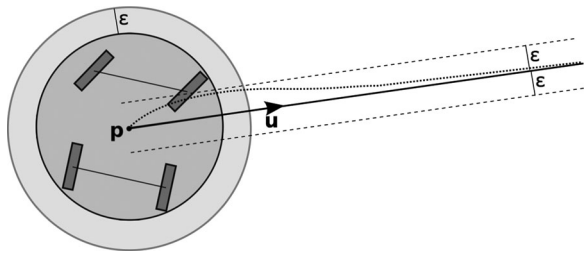

Fig. 3. *Reference* trajectory with constant control velocity (continuous) and local trajectory (dotted) for a robot with kinematic and dynamic constraints. The radius extension and error bounds are marked with ε.

III. Motion Constraints

For robot i, the set of motion primitives is given by a trajectory tracking controller $\mathbf{p}_i(t) = f(\mathbf{z}_i^0, \mathbf{u}_i, t)$ toward the reference parameterized by control velocity $\mathbf{u}_i$, see Section II-A. The set of feasible motion primitives is defined by

$$\mathscr{R}_i(\mathbf{z}_i^0) = \{\mathbf{u}_i \in \mathbb{R}^2, \text{such that the trajectory } f(\mathbf{z}_i^0, \mathbf{u}_i, t) \text{ respects all dynamic constraints}\}. \tag{7}$$

Constraint 1 (Dynamic restrictions): For a maximum tracking error ε_i and current state $\mathbf{z}_i^0 = [\mathbf{p}_i^0, \dot{\mathbf{p}}_i^0, \ddot{\mathbf{p}}_i^0]$, the set of control velocities $\mathbf{u}_i$ that can be achieved with position error below ε_i is denoted by $R_i := R(\mathbf{z}_i, \varepsilon_i)$

$$R_i := \{\mathbf{u}_i \in \mathscr{R}_i(\mathbf{z}_i^0) \mid ||(\mathbf{p}_i^0 + \tilde{t}\mathbf{u}_i) - f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t})|| \leq \varepsilon_i \quad \forall \tilde{t} > 0\} \tag{8}$$

which is invariant with respect to the initial position of the robot.

A mapping γ from initial state $\mathbf{z}_i^0$ and control velocity $\mathbf{u}_i$ to maximum tracking error can be precomputed and stored in a look-up table

$$\gamma(\mathbf{u}_i, \mathbf{z}_i^0) = \max_{\tilde{t}>0} \|(\mathbf{p}_i^0 + \tilde{t}\mathbf{u}_i) - f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t})\|. \tag{9}$$

An example of this precomputed set is shown in Fig. 4 for the case of a robotic car, where the tracking error depends on the initial speed and steering angle. A bounding box H_i can also be computed such that $R_i \subset H_i$.

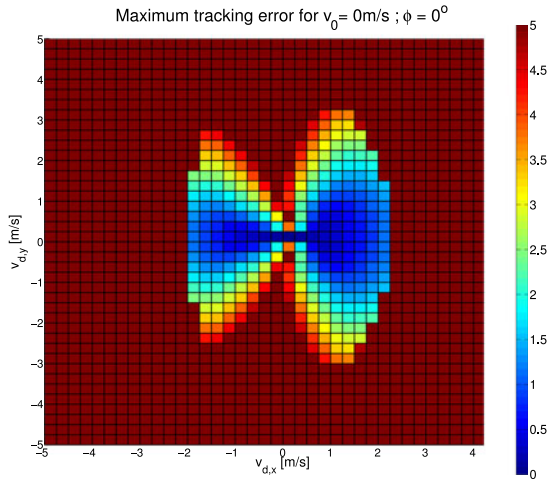

Fig. 4. Maximum tracking errors in $[m]$, and saturated at 5 m, for a carlike robot [29] and for varying values of the control velocity $\mathbf{u}$. In this example, the initial velocity is set to $v_i^0 = 0\,\mathrm{m/s}$ and the initial steering angle to $\phi = 0°$. As expected, the robot can easily track control velocities aligned with its orientation, but experiences high tracking errors for control velocities in the perpendicular direction (which would require the car to rotate).

We now provide an overview of robot models that have been implemented and tested with this framework. These cover most of the typical robotic platforms.

1) Unicycle (Differential Drive): The trajectories can be defined by a circumference arc followed by a straight-line reference at control velocity $\mathbf{u}_i$. The angular and linear velocity over the arc can be computed to achieve the correct orientation within a fixed amount of time T, we employed three times the time-step of the controller, and minimized the tracking error with respect to the control reference [28]. In this formulation, robots had no constraints in acceleration and the linear and angular velocities showed a discontinuity.

2) Bicycle (Carlike): A trajectory tracking controller [42] can be applied at the middle point of the vehicle to track the control reference. This controller was obtained by applying full-state linearization via dynamic feedback to the nonlinear system. Additional constraints, such as maximum steering angle, maximum angular and linear velocity, and maximum acceleration, can be added as saturation limits [29]. This framework guarantees continuity in both linear velocity and steering angle.

3) Unicycle With Dynamic Constraints: The method can also be applied to other robot morphologies, such as robotic wheelchairs with dynamic constraints, e.g., maximum linear speed, maximum angular speed, and maximum acceleration [38]. Since any reference point $\mathbf{p}$ to the front of the vehicle rear axle is fully controllable [42], we controlled the middle point of the robot via a second-order integrator [31] toward the control reference defined by the control velocity $\mathbf{u}$, and added saturation limits.

4) Humanoid Robots: To apply the method to Aldebaran NAO humanoid robots, we defined the motion primitives [46] by a sequence of constant linear velocity and constant angular velocity segments, similar to the case for simple unicycle robots, with the additional option for in-place rotation.

5) Holonomic Boat With Dynamic Constraints: Alternatively, one may employ an LQR-controller, or any other type of controller, in position and velocity to track the reference given by $\mathbf{u}_i$. This is the approach that we follow to control omnidirectional boats, each driven by three rotating propellers. Additional saturation limits are included to account for the boat restrictions.

To sum up, the method can be applied to any kinodynamic model by designing a specific trajectory tracking function $f(\mathbf{z}_i^0, \mathbf{u}_i, t)$ and precomputing the tracking errors. Yet, the method is best suited for robots with fast dynamics, since the set R_i is less restricted.

IV. Pairwise Cooperative Avoidance

We now describe the constraint for avoidance of other robots, which builds on VOs [41], RVOs [22] and ORCA [23]. The constraint is computed for robots of radius $\hat{r}_i = r_i + \varepsilon_i$.

A. Velocity Obstacle in Relative Velocity Space

For a pair of robots, the VO [41] is given by the relative control velocities $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ that lead to a collision between the two robots within a time horizon τ.

Constraint 2 (Inter-Robot Collision Avoidance): For every pair of neighboring robots $\{i, j\} \in \mathscr{I}$, where $i \neq j$, the collision avoidance constraint is given by the control velocities leading to a future collision, i.e., $\|\mathbf{p}_i - \mathbf{p}_j + (\mathbf{u}_i - \mathbf{u}_j)\tilde{t}\| \geq \hat{r}_{i+j} = \hat{r}_i + \hat{r}_j$, for all $\tilde{t} \in [0, \tau]$.

This constraint can be rewritten [23] as $\mathbf{u}_i - \mathbf{u}_j \notin \mathrm{VO}_{ij}^{\tau} = \bigcup_{\tilde{t}=0}^{\tau}((D(\mathbf{p}_j, \hat{r})) \oplus D(\mathbf{p}_i, \hat{r}_i)/\tilde{t})$, a truncated cone as shown in Fig. 5, which is only computed if the distance between the two robots is below a threshold ($p_{ij} = ||\mathbf{p}_{ij}|| < K_d$).

The nonconvex constraint $\mathbb{R}^2 \setminus \mathrm{VO}_{ij}^{\tau}$ can be approximated by three linear constraints of the form $\mathbf{n}_{ij}^l \cdot \mathbf{u}_{ij} \leq b_{ij}^l$, with $l \in \{1, 2, 3\}$, and given by

$$\begin{bmatrix} \cos(\gamma^+) \\ \sin(\gamma^+) \end{bmatrix} \mathbf{u}_{ij} \leq 0,$$
$$-\frac{\mathbf{p}_{ij}}{p_{ij}} \cdot \mathbf{u}_{ij} \leq \frac{p_{ij} - \bar{r}_{i+j}}{\tau},$$
$$\begin{bmatrix} \cos(\gamma^-) \\ \sin(\gamma^-) \end{bmatrix} \mathbf{u}_{ij} \leq 0 \tag{10}$$

where $\gamma^+ = \alpha + \beta$, $\gamma^- = \alpha - \beta$, $\alpha = \mathrm{atan2}(-\mathbf{p}_{ij})$ and $\beta = \mathrm{acos}(\bar{r}_{i+j}/p_{ij})$. The first and last constraints represent avoidance to the right and to the left, respectively, and the middle constraint represents a head-on maneuver, which remains collision-free up to $\tilde{t} = \tau$.

The constraint $\mathbb{R}^2 \setminus \mathrm{VO}_{ij}^{\tau}$ can be linearized directly from a velocity [22] or linearized by selecting one of the three linear constraints. Sensible choices include the following.

1) *Fixed side for avoidance:* If robots are moving toward each other, i.e., $\mathbf{v}_{ij} \cdot \mathbf{p}_{ij} < 0$, avoid on a predefined side, e.g., on the left ($l^* = 1$) or on the right ($l^* = 3$). If robots are not moving toward each other, the constraint perpendicular to

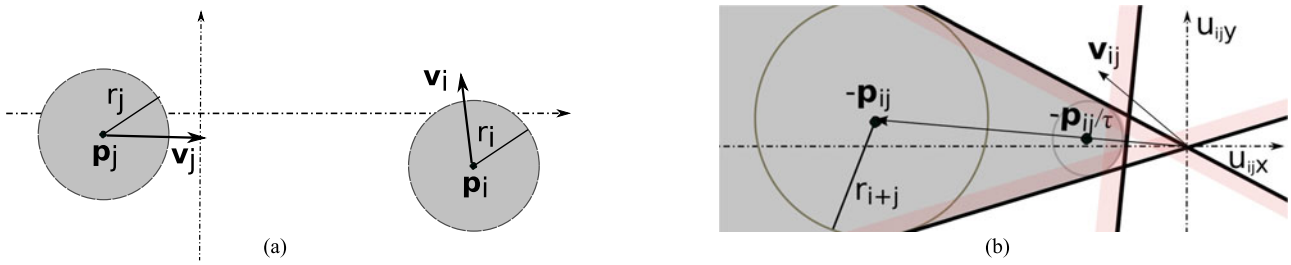

Fig. 5. Given two robots in the configuration shown in (a), the constraint for avoidance (shown in (b) in gray) is computed in relative control velocity space. The free space (white) is nonconvex and can be approximated by three linear constraints, of which at least one must be satisfied. (a) Configuration of two interacting robots, with current position and velocity. (b) Constraint (gray) of relative reference velocities ($\mathbf{u}_i - \mathbf{u}_j$) leading to a future collision.

the apex of the cone ($l^* = 2$) is selected to maximize maneuverability.

2) *Maximum constraint satisfaction with respect to the current relative velocity*

$$l^* = \arg\min_l \left(\mathbf{n}_{ij}^l \cdot (\mathbf{v}_i - \mathbf{v}_j) - b_{ij}^l\right) \tag{11}$$

which maximizes the feasible area in a neighborhood of the current velocities.

3) *Maximum constraint satisfaction with respect to the preferred velocity*, i.e.,

$$\arg\min_l \left(\mathbf{n}_{ij}^l \cdot (\bar{\mathbf{u}}_i - \bar{\mathbf{u}}_j) - b_{ij}^l\right) \text{ if centralized.}$$
$$\arg\min_l \left(\mathbf{n}_{ij}^l \cdot (\bar{\mathbf{u}}_i - \mathbf{v}_j) - b_{ij}^l\right) \text{ if distributed.}$$

This selection may provide faster progress toward the goal position, yet the optimization can become infeasible if the robot greatly deviates from its preferred trajectory.

The first option provides the best coordination results as it incorporates a social rule. The second option maximizes the feasible area of the optimization and the third option may provide faster convergence. We employ the second option.

B. Reciprocal Collision Avoidance

In the distributed case, the new reference velocity $\mathbf{u}_j$ of the neighboring robot is unknown. An assumption must be made. Naive assumptions are to consider the robot static, $\mathbf{u}_j = \mathbf{0}$, or that it follows a constant velocity, $\mathbf{u}_j = \mathbf{v}_j$. In both cases, decision making is not taken into account and oscillations can happen in multirobot scenarios.

In multirobot scenarios we may consider the case where all robots employ the same algorithm. In this line, RVOs [22] extended the concept of VOs by shifting the velocity cone to share the avoidance effort. To avoid reciprocal dances, the idea was later extended to ORCA [23].

In reciprocal collision avoidance, the goal is to obtain two sets $\mathscr{F}_{i|j} \subset \mathbb{R}^2$ and $\mathscr{F}_{j|i} \subset \mathbb{R}^2$ such that for every velocity $\mathbf{u}_i \in \mathscr{F}_{i|j}$, and for every velocity $\mathbf{u}_j \in \mathscr{F}_{j|i}$, the relative velocity is collision free

$$\left.\begin{array}{l}\mathbf{u}_i \in \mathscr{F}_{i|j} \\ \mathbf{u}_j \in \mathscr{F}_{j|i}\end{array}\right\} \Rightarrow \mathbf{u}_i - \mathbf{u}_j \in \mathbb{R}^2 \setminus \mathrm{VO}_{ij}^\tau \tag{12}$$

equivalent to $\mathscr{F}_{i|j} \oplus (-\mathscr{F}_{j|i}) \subset \mathbb{R}^2 \setminus \mathrm{VO}_{ij}^\tau$. If this holds, then robots i and j can freely select a velocity within $\mathscr{F}_{i|j}$ and $\mathscr{F}_{j|i}$, respectively.

To achieve this, first the centralized VO constraint $\mathbb{R}^2 \setminus \mathrm{VO}_{ij}^\tau$ is linearized, resulting in a constraint $\mathbf{u}_{ij} \leq b_{ij}^{l^*}$. Then, the partition is given by a value $b_{i|j} \in \mathbb{R}$ such that

$$\left.\begin{array}{l}\mathscr{F}_{i|j} = \{\mathbf{u}_i | \mathbf{n}_{ij}^{l^*} \cdot \mathbf{u}_i \leq b_{i|j}\} \\ \mathscr{F}_{j|i} = \{\mathbf{u}_j | -\mathbf{n}_{ij}^{l^*} \cdot \mathbf{u}_j \leq b_{ij}^{l^*} - b_{i|j}\}\end{array}\right\} \Rightarrow \mathbf{n}_{ij}^{l^*} \cdot \mathbf{u}_{ij} \leq b_{ij}^{l^*}. \tag{13}$$

For reciprocal collision avoidance, $b_{i|j}$ was defined as a function of the avoidance effort. Denote by $\Delta\mathbf{u}_{ij}$ the minimum change in relative velocity required to avoid a collision, then the minimum required change in velocity for robot i is $\Delta\mathbf{u}_i = \lambda_{i|j}\Delta\mathbf{u}_{ij}$ with $\lambda_{i|j}$ a constant to indicate how the avoidance effort is shared between both robots. For robot j, it is then assumed $\Delta\mathbf{u}_j = -(1 - \lambda_{i|j})\Delta\mathbf{u}_{ij}$, i.e., it reciprocates. From (14), and following [23], we then have

$$b_{i|j} = \lambda_{i|j} b_{ij}^{l^*} + \mathbf{n}_{ij}^{l^*} \cdot ((1 - \lambda_{i|j})\mathbf{v}_i + \lambda_{i|j}\mathbf{v}_j). \tag{14}$$

The parameter $\lambda_{i|j}$ defining the collaboration effort must be fixed and know, typically considering that robots equally cooperate $\lambda_{i|j} = 0.5$ or that they are dynamic obstacles $\lambda_{i|j} = 1$.

C. Cooperative Avoidance

We now extend the method to remove the assumption of known $\lambda_{i|j}$ and take into account a utility function, or prediction, over control velocities, see Section II-B, when computing the sets $\mathscr{F}_{i|j}$ and $\mathscr{F}_{j|i}$. Thus, producing a cooperative partition. In particular, we maximize a cooperation measure, which is a function of the utility functions $\phi_i(\mathbf{u}_i)$ and $\phi_j(\mathbf{u}_j)$ of both interacting robots.

The cooperation measure $\Upsilon(l, b_{i|j})$ is defined for each linearization option $l \in \{1, 2, 3\}$ of the collision-free relative velocities $\mathbb{R}^2 \setminus \mathrm{VO}_{ij}^\tau$ and for each value $b_{i|j}$ defining the partition. This measure is given by

$$\Upsilon(l, b_{i|j}) = \Upsilon(l) + \Upsilon(i, l, b_{i|j}) + \Upsilon(i, l, b_{j|i}) \tag{15}$$

where $\Upsilon(l)$ can favor a certain avoidance topology, e.g., to encode preference for avoidance on the right, and $\Upsilon(i, l, b_{i|j})$ is a function of the utility of the collision-free velocities within the partition $\mathscr{F}_{i|j}$ of robot i ($\Upsilon(i, l, b_{j|i})$ for robot j). To bind the problem, we consider disks D_i and D_j centered at the current velocity of each robot, e.g., of radius $\tau\mathbf{a}_i^{\max}$ representing the velocities that can be achieved within the time horizon, and limit the control velocities to this disk.

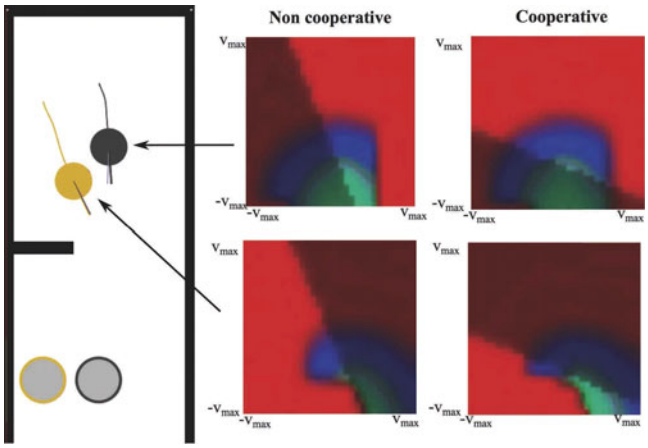

Fig. 6. Example of velocity utility for two robots (black/brown) navigating toward their goal (gray disk) in a corridor environment, and their reciprocal/cooperative partition for distributed collision avoidance (shadowed area represents the unfeasible side of the constraint). The maximum of the velocity utility is displayed in light blue, decreasing toward dark blue and with its minimum (zero value) red. Values are scaled and independent in both images. For visualization, the axes are aligned with the map.

We define each robot's term by the sum of utilities of all the control velocities within a neighborhood of the current one

$$\Upsilon(i,l,b_{i|j}) = \frac{\int_{\mathbf{u}_i \in D_i \cap \mathscr{F}_{i|j}} \phi_i(\mathbf{u}_i) d\mathbf{u}_i}{\int_{\mathbf{u}_i \in D_i} \phi_i(\mathbf{u}_i) d\mathbf{u}_i} \tag{16}$$

with $\mathscr{F}_{i|j}$ and $\mathscr{F}_{j|i}$ following (14). For each robot, and fixed l, this is a monotonically increasing/decreasing function of $b_{i|j}$, valued in the range $[0,1]$.

The optimal partition and linearization are then given by

$$(l^*, b^*_{i|j}) = \arg\max_{l, b_{i|j}} \Upsilon(l, b_{i|j}) \tag{17}$$

obtained by first computing the value of $\Upsilon(l, b_{i|j})$ for each $l \in \{1,2,3\}$ and then selecting the maximum among them. An example of this constraint is shown in Fig. 6.

Constraint 3 (Distributed Inter-Robot Collision Avoidance): For every pair of neighboring robots $\{i,j\} \in \mathscr{I}$, where $i \neq j$, the distributed cooperative collision avoidance constraint is then given by $\mathscr{F}_{i|j} = \{\mathbf{u}_i | \mathbf{n}^{l^*}_{ij} \cdot \mathbf{u}_i \leq b^*_{i|j}\}$.

In Fig. 7, we show schema of the computation of the inter-robot collision avoidance constraint in the centralized and distributed cases.

We now show that the partitions of Section IV-B can be obtained as particular cases of this approach. In particular,

1) Static robot assumption with

$$\phi_j(\mathbf{u}_j) = 1, \text{for } \mathbf{u}_j = \mathbf{0}; \quad \phi_j(\mathbf{u}_j) = 0 \text{ otherwise.}$$

2) Constant velocity assumption with

$$\phi_j(\mathbf{u}_j) = 1, \text{for } \mathbf{u}_j = \mathbf{v}_j; \quad \phi_j(\mathbf{u}_j) = 0 \text{ otherwise.}$$

3) Constant preferred velocity assumption with

$$\phi_j(\mathbf{u}_j) = 1, \text{for } \mathbf{u}_j = \bar{\mathbf{u}}_j; \quad \phi_j(\mathbf{u}_j) = 0 \text{ otherwise,}$$

where $\bar{\mathbf{u}}_j$ is the preferred velocity of robot j, see Section III.

4) The reciprocal partition is obtained, for $\lambda_{i|j} = 0.5$, by employing the cooperation measure

$$\Upsilon_r(l, b_{i|j}) = \max(\mathbf{n}^l_{ij} \cdot \mathbf{v}_i - b_{i|j}, -\mathbf{n}^l_{ij} \cdot \mathbf{v}_i - (b_{ij} - b_{i|j})).$$

V. Method

We now introduce the εCCA method for collision avoidance. First, we need an additional constraint for avoiding static obstacles. Recall $\hat{r}_i = r_i + \varepsilon_i$.

Constraint 4 (Collision Avoidance Static Obstacles): For robot $i \in \mathscr{I}$, this constraint is given by the reference velocities such that the new positions are not in collision with the enlarged obstacle map, $\mathbf{p}_i + \mathbf{u}_i \tilde{t} \notin \bar{\mathscr{O}}_{\hat{r}_i}$, for all $\tilde{t} \in [0, \tau]$. Let us denote this constraint by $\mathbf{u}_i \notin \bar{\mathscr{Q}}_i$.

In each control loop, we compute the optimal control velocity $\mathbf{u}^*_i$ via a constrained optimization, which can be centralized or distributed, and consists of constraints for respecting the kinodynamic model of the robot, avoiding other robots and avoiding static obstacles.

We define the optimization cost $J(\mathbf{u}_i)$ by a quadratic function, given by a weighted sum of two terms: A regularizing term penalizing changes in velocity and a minimizer of the deviation with respect to the preferred velocity $\bar{\mathbf{u}}_i$.

$$J(\mathbf{u}_i) := K_o ||\mathbf{u}_i - \mathbf{v}_i||^2 + (\mathbf{u}_i - \bar{\mathbf{u}}_i)^T D_i^T L D_i (\mathbf{u}_i - \bar{\mathbf{u}}_i) \tag{18}$$

where K_o is a design constant and the matrices D_i and L produce an elliptical cost to penalize changes in speed over changes in orientation. This follows the observation that pedestrians prefer to maintain a constant velocity in order to minimize energy [43]. The relative weighting is defined by

$$L = \begin{bmatrix} \Lambda & 0 \\ 0 & 1 \end{bmatrix} \text{ and } D_i = \begin{bmatrix} \cos\gamma_i & \sin\gamma_i \\ -\sin\gamma_i & \cos\gamma_i \end{bmatrix}$$

where $\Lambda > 0$ is the relative weight and γ_i the orientation of $\bar{\mathbf{u}}_i$.

Let us denote by $\mathbf{u}_{1:n} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ the union of control velocities for all robots. We can define the centralized cost $J(\mathbf{u}_{1:n})$ as the weighted sum of the cost functions $J(\mathbf{u}_i)$

$$J(\mathbf{u}_{1:n}) := \sum_{i=1}^{n} \omega_i J(\mathbf{u}_i) \tag{19}$$

where ω_i represents the weight of each individual robot in the avoidance effort and can be used to define characteristics, such as shy and aggressive behavior.

A. Centralized Collision Avoidance

Problem 1 can be solved via a single optimization where the optimal control velocities of all robots $\mathbf{u}^*_{1:n} = [\mathbf{u}^*_1, \ldots, \mathbf{u}^*_n]$ are jointly computed. This is

$$\begin{aligned} \mathbf{u}^*_{1:\mathbf{n}} &= \arg\min_{\mathbf{u}_{1:n}} J(\mathbf{u}_{i:n}) & & \\ \text{s.t. } ||\mathbf{u}_i|| &\leq u_{\max} & & \forall i \in \mathscr{I} \\ \mathbf{u}_i &\in R_i & \text{Constraint 1} & \forall i \in \mathscr{I} \\ \mathbf{u}_i, \mathbf{u}_j &\notin \text{VO}^{\tau}_{ij} & \text{Constraint 2} & \forall i,j \in \mathscr{I} \\ \mathbf{u}_i &\notin \bar{\mathscr{Q}}_i & \text{Constraint 4} & \forall i \in \mathscr{I}. \end{aligned} \tag{20}$$

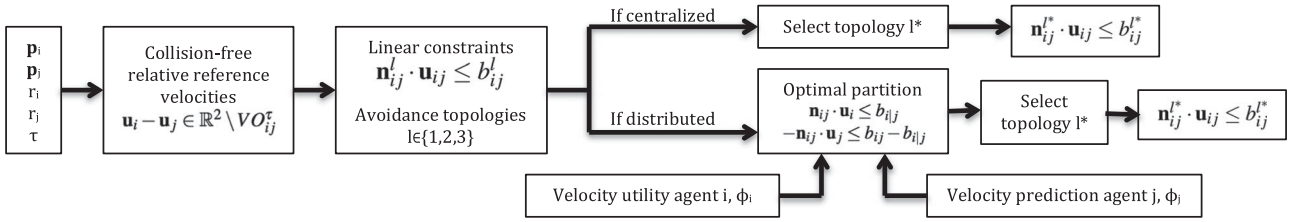

Fig. 7. Schema of the computation of the collision avoidance constraints for centralized and distributed collision avoidance.

We now describe two approaches to formulate and solve this nonconvex optimization problem.

1) Convex Optimization: The centralized optimization of (21) can be approximated by a convex problem, by linearizing all the constraints [37]. In particular, the set R_i of motion constraints, see Section III, is approximated by an inscribed polygon, and each VO, see Section IV-A and (11), is approximated by its linearization.

Static obstacles are accounted for via the convex polygon $P(\mathbf{p}_i^0, \hat{r}_i)$, which is fully contained in free space $\mathbf{p}_i^0 \in P(\mathbf{p}_i^0, \hat{r}_i) \subset \mathscr{W} \setminus \bar{\mathcal{O}}_{\hat{r}_i}$, recall (1). This polygon is converted to velocity space by considering the straight-line control reference, i.e., $\mathbf{p}_i^0 + \mathbf{u}_i \tau \in P(\mathbf{p}_i^0, \hat{r}_i)$, which results in the constraint

$$\mathbf{u}_i \in \frac{P(\mathbf{p}_i^0, \hat{r}_i) - \mathbf{p}_i^0}{\tau}. \tag{21}$$

From the convexity of $P(\mathbf{p}_i^0, \hat{r}_i)$, and since $\mathbf{p}_i^0 \in P(\mathbf{p}_i^0, \hat{r}_i)$, this constraint guarantees Constraint 4.

This optimization problem consists of $2n$ real-valued variables and $|\text{VO}| + \sum_{i \in \mathscr{I}} K_{R_i}$ constraints, where $|\text{VO}| \leq \min(n(n-1)/2, K_c n)$ is the number of VO constraints, K_c is the maximum number of neighbors taken into account in the collision avoidance and K_{R_i} is the number of faces in the polygonal approximation of R_i. The computational complexity is low and scalability is relatively good, but the solution space is partially reduced due to the linearization of the constraints, since an avoidance topology is fixed.

2) Nonconvex Optimization: To obtain the global optimum of the original optimization problem, a nonconvex optimization must be solved. An approach is to formulate the optimization as a mixed integer quadratic program (MIQP) [37], where three binary variables, $\beta^l_{ij} \in \{0, 1\}$, are added for each inter-robot collision avoidance constraint VO^{τ}_{ij} and specify which one of the three linear constraints is active. Each of these three linear constraints, see (11), is then rewritten as

$$\mathbf{n}^l_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) - M\beta^l_{ij} \leq b^l_{ij} \quad \forall l \in [1, 3] \tag{22}$$

where $M > 0$ is a large scalar. If $\beta^l_{ij} = 0$, then the original constraint is active. The additional constraint $\sum_{l=1}^{3} \beta^l_{ij} = 2$ is introduced to guarantee that at least one of the original constraints is satisfied. This MIQP can be solved via standard branch-and-bound methods.

With this method the complete solution space is explored and anytime optimality is achieved. The optimization problem consists of $2n$ real-valued variables, $3|\text{VO}|$ binary variables and $4|\text{VO}| + \sum_{i \in \mathscr{I}} K_{R_i}$ constraints. Due to the relatively large number of binary variables, this optimization can only be solved inside a real-time control loop for a low number of robots and the scalability is poor.

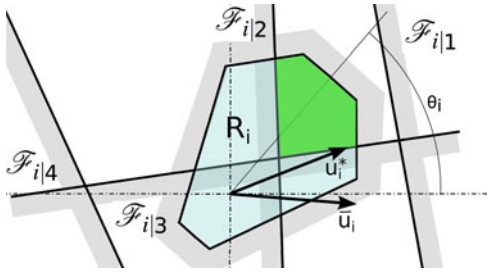

Fig. 8. Optimization in velocity space for a unicycle robot i with current orientation θ_i in a scenario with five robots. The optimal collision-free velocity $\mathbf{u}_i^*$ satisfies the linear constraints (marked with a straight line with the light blue side indicating the unfeasible half-plane) and minimizes the distance to the preferred velocity $\bar{\mathbf{u}}_i$. The feasible region is highlighted in green.

An alternative would be to solve the nonconvex optimization problem directly, without additional binary variables. In [44], we described an approach based on the alternating direction method of multipliers algorithm. Yet, kinodynamic constraints were not accounted for and the method did not guarantee global optimality.

B. Distributed Collision Avoidance

In Problem 2, each robot i independently computes its optimal reference velocity $\mathbf{u}_i^*$. We assume that the position $\mathbf{p}_j$ and velocity $\mathbf{v}_j$ of neighboring robots is known and solve

$$\begin{aligned} \mathbf{u}_i^* &= \arg\min_{\mathbf{u}_i} J(\mathbf{u}_i) \\ \text{s.t. } &||\mathbf{u}_i|| \leq u_{\max} \\ &\mathbf{u}_i \in R_i \qquad \text{Constraint 1} \\ &\mathbf{u}_i \in \mathscr{F}_{i|j} \qquad \text{Constraint 3} \quad \forall j \in \mathscr{I} \setminus \{i\} \\ &\mathbf{u}_i \notin \bar{\mathcal{Q}}_i \qquad \text{Constraint 4.} \end{aligned} \tag{23}$$

1) Convex Optimization: The distributed optimization of (24) can be approximated by a convex problem, by linearizing all the constraints [28], [29]. The set R_i of motion constraints and the free space $\mathscr{W} \setminus \bar{\mathcal{Q}}_i$ are approximated by convex polytopes as described for the centralized case in Section V-A1. For each neighboring robot j, the collision-avoidance constraint is computed, linearized, and partitioned, see Section IV-B, leading to a linear constraint $\mathbf{u}_i \in \mathscr{F}_{i|j}$. An example of these constraints is shown in Fig. 8.

This optimization problem consists of two real-valued variables and $|\text{VO}_i| + K_{R_i}$ constraints, where $|\text{VO}_i| \leq \min((n-1)/2, K_c)$ is the number of VO constraints. The computational complexity of this method is very low and scalability is very

good, yet the solution space is greatly reduced due to the linearization of the constraints and the partition of the reference velocity space.

2) Search Within Convex Region: To reduce the problem to a convex optimization, we had to approximate the sets R_i and $\mathscr{W} \setminus \bar{\mathscr{D}}_i$ by inscribed convex polytopes. We now describe a method, see Fig. 9 for schema, which combines convex (linear) and nonconvex (grid-based) constraints to explore a larger solution space while keeping the computational cost low.

The set of convex constraints is formed by the inter-robot linearized collision avoidance constraints $\mathscr{F}_{i|j}$, i.e., Constraint 3, and the bounding box H_i of the set R_i, see Section III. Denote this set by $\mathscr{C}_i$. The set of nonconvex constraints is formed by the set R_i (Constraint 1), and the static obstacles (Constraint 4). This set is denoted by $\hat{\mathscr{C}}_i$. Both nonconvex constraints are given by grid representations of identical resolution.

The optimization is divided in two parts. First, a convex subproblem is solved resulting in $\mathbf{u}_i^c$, followed by a search within the grid-based constraints restricted to the convex area defined by the linear constraints. Note that the cost function is not required to be quadratic any more. For robot $i \in \mathscr{I}$, the algorithm proceeds as follows.

Algorithm 1: Collision-free Trajectory for Robot i.

Data: $\mathbf{z}_i^0$, $\bar{\mathbf{u}}_i$ and $\hat{r}_i$; $\mathbf{p}_j$, $\mathbf{v}_j$ and r_j $\forall j$ neighbor of i
Consider $\varepsilon_j = \varepsilon_i \leq (||\mathbf{p}_i - \mathbf{p}_j|| - r_i - r_j)/2$
Result: Collision-free trajectory for time horizon τ, given by the controller $f(\mathbf{z}_i^0, \mathbf{u}_i^*, t)$ and optimal control velocity $\mathbf{u}_i^*$
Compute Constraints 1, 3, and 4 for robot i;
$\mathbf{u}_i^c \leftarrow$ solution 2-dimensional convex optimization with quadratic cost (19) and convex constraints $\mathscr{C}_i$;
// Wave expansion from $\mathbf{u}_i^c$ within convex area $\mathscr{C}_i$;
Initialize sorted list $\mathscr{L}$ (increasing cost $J(\mathbf{u}_i)$) with $\mathbf{u}_i^c$;
while $\mathscr{L} \neq \emptyset$ **do**
  $\mathbf{u}_i \leftarrow$ first point in $\mathscr{L}$; $\mathscr{L} := \mathscr{L} \setminus \mathbf{u}_i$; feasible := 'true';
  if feasibleDynamics($\mathbf{u}_i$) = 'false' **or** feasibleMap($\mathbf{u}_i$) = 'false' **then**
    $\mathscr{L} \leftarrow$ expandNeighbors($\mathscr{L}$, $\mathbf{u}_i$);
    feasible := 'false';
  end
  if feasible = 'true' **then**
    return $\mathbf{u}_i^* = \mathbf{u}_i$;
    // This assumes that the cost $J(\mathbf{u}_i)$ is convex ;
  end
end
return 0;
Track $\mathbf{u}_i^*$ with controller $f(\mathbf{z}_i^0, \mathbf{u}_i^*, t)$;

Function *feasibleDynamics($\mathbf{u}_i$)* checks in a precomputed grid if the tracking error is below ε_i, given the initial state of the vehicle.

if $\mathbf{u}_i \in R_i$ [see (9)] **then return** 'true'; **else** 'false';

Function *feasibleMap($\mathbf{u}_i$)* checks if $\mathbf{u}_i$ leads to a trajectory in collision with static obstacles given by the grid map $\hat{\mathscr{O}}$. This is efficiently checked in the precomputed dilated map $\mathscr{O}_{\hat{r}_i}$, see Constraint 4.

if segment $(\mathbf{p}_i, \mathbf{p}_i + \mathbf{u}_i\tau) \cap \mathscr{O}_{\hat{r}_i} = \emptyset$ **then return** 'true';

The function *expandNeighbors($\mathscr{L}$, $\mathbf{u}_i^{in}$)* adds the neighboring grid points if they are within the convex region defined by the convex constraints in $\mathscr{C}$, and they were not previously explored.

Algorithm 2: Function *expandneighbors($\mathscr{L}$, $\mathbf{u}_i^{in}$):*

Data: List $\mathscr{L}$, velocity $\mathbf{u}_i^{in}$
Result: Updated list $\mathscr{L}$
for each 8-connected grid neighbor $\mathbf{u}_i^{neighbor}$ of $\mathbf{u}_i^{in}$ **do**
  if $\{\mathbf{u}_i^{neighbor}$ not previously added to $\mathscr{L}\}$ **and** $\{\mathbf{u}_i^{neighbor}$ satisfies convex constraints $\mathscr{C}_i\}$ **then**
    $\mathscr{L} \leftarrow \mathscr{L} \cup \mathbf{u}_i^{neighbor}$;
    Sort list $\mathscr{L}$, increasing cost $J(\mathbf{u}_i)$;
  end
end

This optimization consists of two variables, $4 + |\text{VO}_i|$ linear constraints from the bounding box of R_i and the VOs, and a 2-D grid search within the bounded area defined by the convex constraints. The computation complexity of this problem is relatively low and scalability is good. The solution space is larger than in the convex case, yet, it is still reduced due to the linearization of the collision-avoidance constraints and the partition of the reference velocity space to guarantee safety in the distributed case.

C. Collision-Avoidance Guarantees and Remarks

Remark 1 (Radius Enlargement): Variable maximum tracking error ε_i and radius enlargement is required for feasibility. At all times it must be satisfied that the extended radii of the robots are not in collision, i.e., $r_i + r_j + \varepsilon_i + \varepsilon_j \leq ||\mathbf{p}_i - \mathbf{p}_j||$. This is achieved by letting $\varepsilon_i > 0$ and $\varepsilon_j > 0$ decrease stepwise when robots are close to each other, reaching zero in the limit, which for differentially driven robots would imply rotating in place [28].

Theorem 1: If the optimization problem is feasible, then the planned local trajectories are collision-free up to time τ under the assumption that other agents follow the same algorithm, or maintain a constant velocity.

Proof: We first show that the planned trajectories are collision-free up to time τ. The intuition is that the control reference, defined by $\mathbf{u}_i$, is collision-free for a robot whose radii is enlarged by ε and the robot stays within ε of this control reference.

Consider two robots i and j controlled with the proposed method. We show that the distance between their centers is greater than the sum of their radii for all time instances up to the time horizon. Let $\mathbf{p}_i(t)$ denote the position of robot i at time $t \geq t_0$ and recall $\tilde{t} = t - t_0$, then avoidance of robot j follows

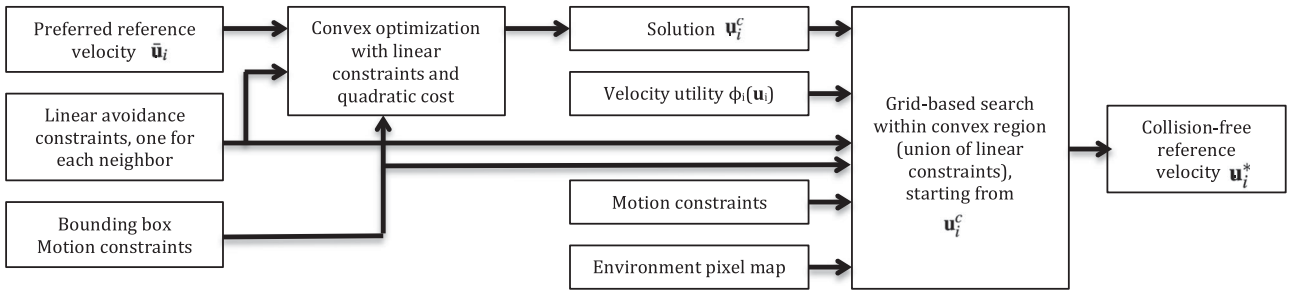

Fig. 9. Schema of the distributed method for nonconvex search within convex region.

from

$$\begin{aligned} ||\mathbf{p}_i(t) - \mathbf{p}_j(t)|| &= ||f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t}) - f(\mathbf{z}_j^0, \mathbf{u}_j, \tilde{t})|| \underset{\mathbf{u}_i \in R_i, \mathbf{u}_j \in R_j}{\geq} \\ &||(\mathbf{p}_i^0 + \mathbf{u}_i\tilde{t}) - (\mathbf{p}_j^0 + \mathbf{u}_j\tilde{t})|| - \varepsilon_i - \varepsilon_j \underset{\mathbf{u}_i - \mathbf{u}_j \notin \mathrm{VO}_{ij}^\tau}{\geq} \\ &r_i + \varepsilon_i + r_j + \varepsilon_j - \varepsilon_i - \varepsilon_j = r_i + r_j \end{aligned} \tag{24}$$

where the first inequality holds from Constraint 1, i.e., $\mathbf{u}_i \in R_i$ and $\mathbf{u}_j \in R_j$. In the centralized case, the second inequality holds directly from Constraint 2, i.e., $\mathbf{u}_i - \mathbf{u}_j \notin \mathrm{VO}_{ij}^\tau$. In the distributed case, the second inequality holds from Constraint 2 and (14), i.e., $\mathbf{u}_i \in \mathscr{F}_{i|j}$ and $\mathbf{u}_j \in \mathscr{F}_{j|i}$.

Avoidance of an obstacle j that moves with constant velocity $\mathbf{v}_j$ holds analogously

$$\begin{aligned} ||\mathbf{p}_i(t) - \mathbf{p}_j(t)|| &= ||f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t}) - (\mathbf{p}_j^0 + \mathbf{v}_j\tilde{t})|| \underset{\mathbf{u}_i \in R_i}{\geq} \\ &||(\mathbf{p}_i^0 + \mathbf{u}_i\tilde{t}) - (\mathbf{p}_j^0 + \mathbf{v}_j\tilde{t})|| - \varepsilon_i \underset{\mathbf{u}_i - \mathbf{v}_j \notin \mathrm{VO}_{ij}^\tau}{\geq} \\ &r_i + \varepsilon_i + r_j - \varepsilon_i = r_i + r_j. \end{aligned} \tag{25}$$

In the case where multiple robots and moving obstacles are present, (25) and (26) hold for each one of them from the pairwise avoidance constraints of the optimization problem. Additionally, it is required that $||\mathbf{p}_i^0 - \mathbf{p}_j^0|| \geq r_i + \varepsilon_i + r_j + \varepsilon_j$. This can be guaranteed by setting $0 \leq \varepsilon_i \leq (||\mathbf{p}_i^0 - \mathbf{p}_j^0|| - r_i - r_j)/2$.

Recalling Constraint 4, we can show that static obstacles are avoided within the time horizon

$$\begin{aligned} \mathbf{u}_i \notin \bar{\mathscr{Q}}_i \quad &\Rightarrow \quad (\mathbf{p}_i^0 + \mathbf{u}_i\tilde{t}) \notin \bar{\mathscr{O}}_{r_i+\varepsilon} \ \forall \tilde{t} \in [0, \tau] \\ &\underset{\mathbf{u}_i \in R_i}{\Rightarrow} \quad f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t}) \notin \bar{\mathscr{O}}_r \quad \forall \tilde{t} \in [0, \tau]. \end{aligned}$$

After each time-step a new collision-free trajectory is computed. The trajectories of all robots, given as concatenation of segments, are therefore collision-free. ■

As will be discussed in Remark 4, it may happen that the optimization is *infeasible*. If so, no collision-free solution exists that respects all the constraints. If the time horizon is longer than the required time to stop, safety is preserved if all involved vehicles drive their last feasible trajectory with a time re-parameterization to reach stop before a collision arises, similar to [31]. This implies a slowdown of the robot, which in turn typically renders the optimization feasible in future time steps. Since this computation is performed at a high frequency, each individual robot is able to adapt to changing situations.

Remark 2 (Dynamic Obstacles): The feasibility of the optimization indicates if moving obstacles can be avoided, assuming that they adhere to their predicted velocity, or a collision is imminent. A fast control loop is able to handle small deviations in the prediction.

Remark 3 (Heterogeneous Robot Teams): In (25), we observe that the derivation does not depend on the kinodynamic model of the robot, thanks to the abstraction provided by $f(\mathbf{z}_i^0, \mathbf{u}_i, \tilde{t})$, ε and the set R_i. The size of the robots can also be different, since they appear directly in the proof. Therefore, the proposed method applies to heterogeneous teams of robots as long as their respective constraints R_i and R_j are computed for their kinodynamic models. We note also that robot i does not require any information about the kinodynamics of other robots as long as they all respect that ε_i is less than half the clearance between robots.

Remark 4 (Infeasibility): Under some circumstances the optimization problem can be infeasible, i.e., not all constraints can be satisfied. In practice, this happens rarely and it is quickly resolved in subsequent iterations of the method as the robot slows down.

Infeasibility can happen for example due to the following.

1) Not enough time to find the solution within the allocated time.
2) Differences between the model and the real vehicle.
3) Large uncertainty in the localization and estimation of vehicles' state.
4) Limited local planning horizon and extreme restriction on motion capabilities to the set of motion primitives described in Section III.

If the method is *distributed*, infeasibility can also arise if a robot has conflicting partitions with respect to different neighbors, static obstacles, or kinematics. This is due to the use of pairwise partitions of velocity space. In our experience, slowing down is a good strategy when the problem is infeasible and these situations are resolved quickly as the robot slows down and the problem becomes feasible again.

An alternative is to relax the constraints by adding slack variables in the optimization problem. In this case, the optimization would always be feasible but safety could be endangered. We chose not to add slack variables and instead decelerate when the problem becomes infeasible.

Remark 5 (Motion Continuity): The method guarantees by construction, via Constraint 1, that the local trajectories respect the kinematic constraints of the robot and its limits in actuators, velocities, and accelerations, as long as the individual trajectory tracking controllers do so. For details see Section III.

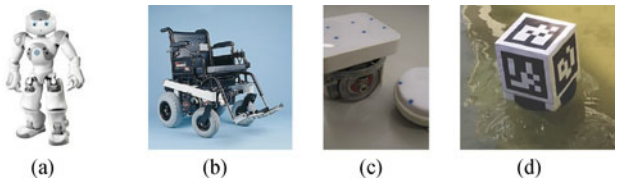

Fig. 10. Robots employed in the experiments. (a) Humanoid. (b) Wheelchair. (c) Unicycle. (d) Boat.

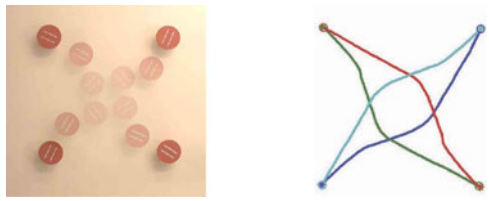

Fig. 11. Experiment with four e-puck robots exchanging antipodal positions. Time-lapse and trajectories. Trajectories are smooth and collision-free.

Remark 6 (Deadlocks): Since the proposed method is for collision avoidance and only considers local information deadlocks where the robot cannot make progress toward its goal can occur. For example, when the robot encounters a large obstacle between its position and the goal. For each robot, and with respect to static obstacles, we avoid deadlocks by employing a, globally computed, cost to go, which for every point in the map provides both the distance to the goal and the desired direction of motion. This does not avoid deadlocks between two or more robots. Multirobot deadlock situations can be resolved by employing a global path planner that guides the robot toward the goal [45] or a mission planner for global coordination and mission satisfaction [46].

VI. Experimental Results

We present experimental results with various robotic platforms, ranging from wheelchairs to boats. First, we describe the experimental setups, followed by results on trajectory smoothness and experimental comparisons of the proposed algorithms. For additional results on shared-control of semiautonomous wheelchairs, we refer the reader to [38]. A video with representative experiments accompanies this paper. The breadth of these experiments is to provide validation of and exemplify the generality of the proposed approach.

A. Experimental Setups

We have tested our method, εCCA, with several platforms, see Fig. 10, of different motion characteristics: Two types of small differentially driven robots [28], [47], simulated carlike robots with bicycle kinematics [29], [37], robotic wheelchairs [38], NAO humanoid robots [46], and omnidirectional boats with slow dynamics. The time-horizon τ of εCCA was in the range 5–8 s and the maximum enlargement ε was in the range 10%–25% of the robot radii. The simulated cars, the wheelchairs, and the boats where subject to maximum acceleration limits of 1–2m/s. Additional limits include the following: For the simulated car, 30° maximum steering angle and 30°/s maximum steering velocity; for the wheelchairs, 2 rad/s maximum angular velocity and 1.5 m/s maximum sideways acceleration. In our simulations, we introduce small measurement noise in the robot positions.

In the experiments, unless noted differently, we linearized the VO constraint with respect to the current relative velocity $\mathbf{v}_i$, see Section IV. This prioritizes feasibility and gives good results in general, but ignores symmetries. To avoid reciprocal dances and minimize deadlocks, a small preference for left-side (+5%) and right-side (+7%) avoidance was typically added. In all cases, if the optimization becomes unfeasible, the robot decelerates at maximum deceleration rate, until it reaches a rest state or the optimization becomes feasible again.

Tracking was performed with over-head cameras and computation took place in distributed threads (one per robot) in a central computer communicating with the robots. The update frequency of the collision avoidance was typically 10 Hz, except for the wheelchairs (30 Hz) and simulated cars (5 Hz). The convex optimization was solved using OOQP [47] and the mixed integer optimization using IBM CPLEX [48]. Computations were performed in a standard 2.66 GHz quadcore PC.

B. Quality of Trajectories

In all the experiments of this section, we employ a distributed version of εCCA. The convex version of εCCA, described in Section V-B1, is well suited for obstacle-free environments. On the other hand, for complex environments and robot dynamics the approach of Section V-B2 is better suited. In the following, we present results with both approaches.

1) Obstacle-Free Environments: In Fig. 11, we show a representative experiment of εCCA with four e-puck robots. We observe that the trajectories are smooth and collision-free. The scalability of the method is shown in the accompanying video in an experiment with 14 e-pucks. Furthermore, in [47], we applied it to control 50 pixelbots. The method can be applied in scenarios with varying number of robots without changes in the parameters. We observe that the robots can successfully solve crowded scenarios while avoiding collisions, yet a slowdown can be noticed in areas of increased robot density. The method applies to other robot physiologies. Fig. 12 shows the trajectories of ten simulated carlike robots for three representative experiments: antipodal position exchange, antipodal position exchange with one nonreacting robot, i.e., dynamic obstacle, and transition to randomized goal positions. Again, the method achieves smooth and collision-free trajectories, even in this scenario, where robots have a limited turning radius.

2) Complex Environments and Robot Dynamics: The proposed method, in its distributed cooperative nonconvex form of Section V-B2, is well suited for navigation in arbitrarily complex environments and for robots of arbitrary kinodynamic constraints. In Fig. 13, we present results of the method for simulated wheelchairs navigating at high speeds in a complex environment. Here, a 20 Hz control rate was maintained and we observe that the paths are smooth and the distance between

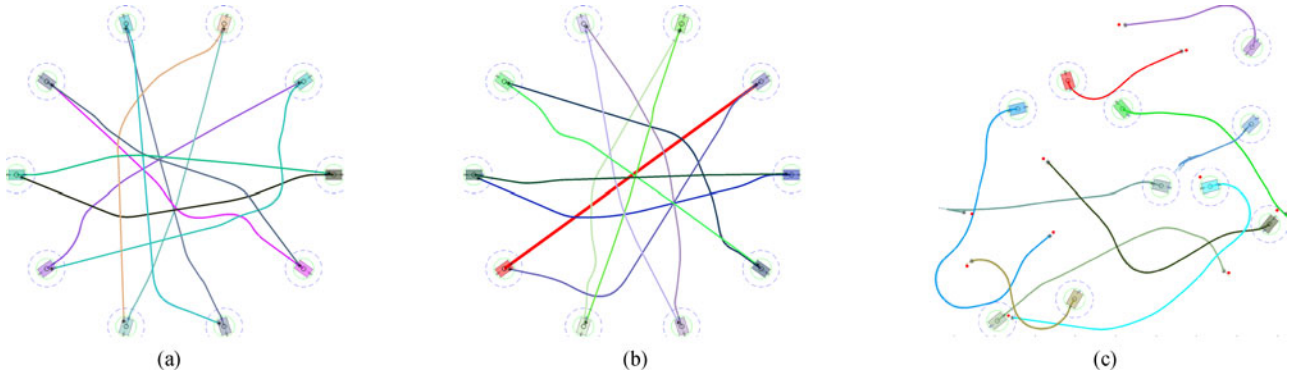

Fig. 12. Trajectories of ten carlike robots moving from their initial to a goal (red circles) configuration. (a) Antipodal positions, cooperative. (b) With a dynamic obstacle (red). (c) Randomized configurations.

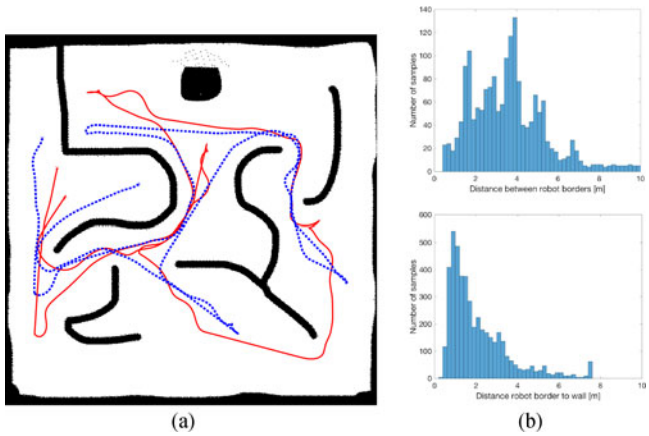

Fig. 13. Two simulated wheelchairs navigate, at a preferred speed of 5 m/s, between several goal positions in a complex environment and successfully avoid collisions via cooperative εCCA. (a) Traces of the path of both robots and obstacle map. (b) Histograms of border to border inter-robot distance (top) and robot border to wall (below). No collisions are observed.

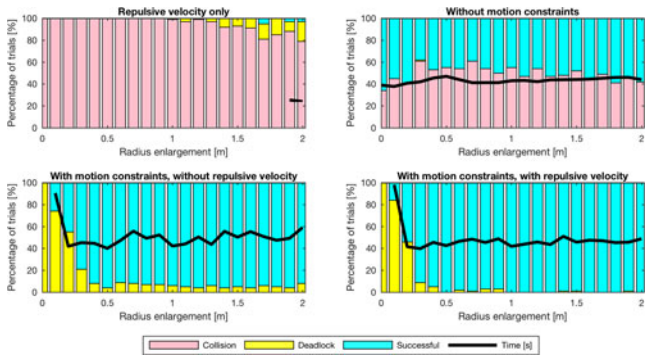

Fig. 14. Performance comparison for four instances of the method in the antipodal exchange scenario of Fig. 12 with ten simulated robots. We display the percentage of trials that ended in collision, deadlock or succeeded, over 100 simulations for each radius enlargement ε. Top left: The preferred velocity corrected with the repulsive velocity is applied directly to the robots. Top right: The optimization method without motion constraints (ORCA). Bottom left: The method with motion constraints (εCCA), but without correcting the preferred velocity with the repulsive velocity. Bottom right: The method with motion constraints and repulsive velocity (εCCA). The same parameters are used in all simulations.

robots and between a robot and a static obstacle is never zero, i.e., no collisions were observed.

C. Method Comparison

In the following, we provide a quantitative analysis of the method. For clarity of explanation, we compare several aspects and instances of the method independently. We first show the value of motion constraints. This is followed by a comparison of the centralized, convex and nonconvex, instantiations of the method, and their scalability with the number of robots. Finally, we provide a comparative analysis of using a cooperative partition of velocity space.

1) Value of Motion Constraints: In Fig. 14, we compare the traditional ORCA [23] method for holonomic robots (top-right) with the distributed version of εCCA (bottom-right), both applied to simulated robots with carlike dynamics, identical time horizon, and robot radii enlarged by ε. We present results for different values of ε, ranging from zero to the robot radius. The test scenario is the antipodal position exchange with ten carlike robots shown in Fig. 12. For each value of ε, the simulation is repeated 100 times with uniform noise in position. Each simulation is classified as: *collision* (if two or more robots are closer than the sum of radii without considering the ε enlargement), *deadlock* (one or more vehicles stop before reaching the goal and do not make progress anymore), and *convergence* (all vehicles reach their goals).

If the preferred velocity, corrected with the repulsive velocity, is directly applied to the robots, then most of the simulations ended in a collision (top-left). If collision avoidance constraints are added but the motion constraints of the robots are ignored, i.e., using ORCA directly, then collisions appear independently of the value of ε and, in this particular scenario, in about 50% of the simulations (top-right). When including the motion constraints, i.e., using εCCA (bottom figures), zero collisions appear in the simulations. Deadlock situations are observed for low values of ε. These deadlocks appear because lower values of ε imply higher restrictions in the motion of the vehicle in order to guarantee safe motion, see Section III. We further observe that, when converged, convergence time was similar with and without motion constraints. If we compare the bottom-right and bottom-left figures, we observe that the addition of a small

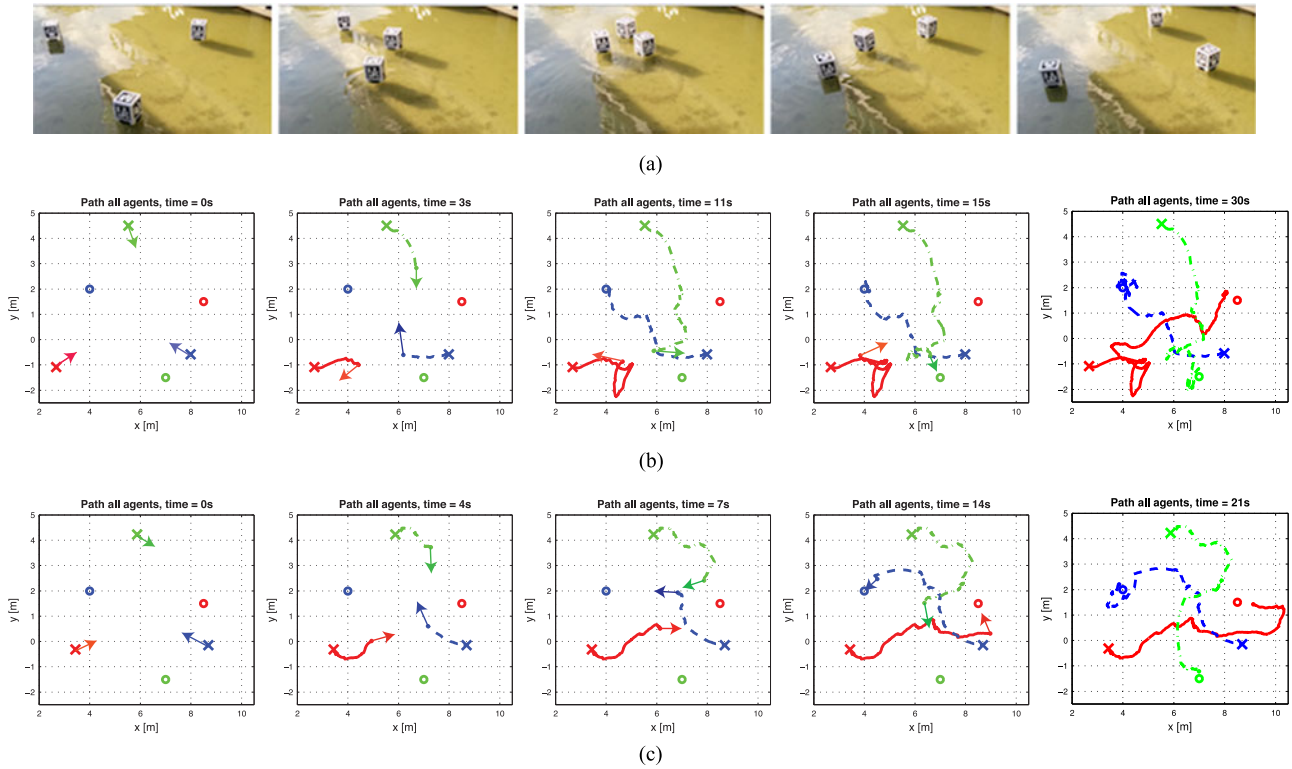

Fig. 15. Representative position exchange with three boats, where paths are crossing. Start positions are displayed with X and goal positions with O. The arrows indicate the commanded velocity at that time instance. (a) Three boats transition to antipodal positions. Representative frames equispaced in time. (b) Collision avoidance with repulsive velocities. The paths of the boats are displayed at representative intervals. Total time is 30 s. (c) Collision avoidance with distributed εCCA. Total time is 20 s.

repulsive velocity to the preferred velocity does indeed improve convergence, lowering the number of deadlock scenarios.

In Fig. 15, we compare distributed εCCA (lower row) and repulsive forces (middle row) when three robotic boats navigate to goal positions. Both methods are tuned for best performance with the system. First, this shows the applicability of εCCA to a system with slow dynamics. Thanks to the identified model of the boats, εCCA was able to successfully avoid collisions, except in rare cases with strong unmodeled wind disturbances. We observe that the resulting path was less smooth than for the other platforms. This was due to wind disturbances, intermittent detection failures, and slower dynamics.

Second, this representative example shows the superior performance of εCCA over repulsive velocities—as also hinted in Fig. 14 and in past experiments with small differential-drive robots [49]. For the purely repulsive approach, oscillations and "bouncing" behaviors are observed, since the velocities of other robots are not taken into account, nor the dynamics of the ago-robot. For instance, see the capture at $t = 3$ s, where the red robot virtually bounces on the blue, followed by another virtual bounce against the green robot at $t = 11$ s. The optimization of εCCA with VOs successfully considers both the topology of the avoidance and the velocity of other robots, leading to smoother behaviors. Convergence to the goal positions was also faster.

In general, another disadvantage of a potential field approach with respect to εCCA is that, in order to guarantee collision-free motion at high speeds (and with dynamic constraints), large potentials may be required, which can result problematic when passing through narrow doors or navigating in close proximity to walls.

2) Convex Versus Nonconvex Optimization and Robustness: Using the distributed version of εCCA, see Section V-B1, as reference, the centralized convex (QP) optimization, see Section V-A1, is compared to the centralized nonconvex (MIQP) optimization, see Section V-A2. In Fig. 16, comparative results of the three methods are shown for varying number of robots. In all experiments, the same parameters are used.

In Fig. 16(a), we compare the computational time for all methods and up to 50 robots, for detailed timings of the distributed approach refer to Table I. The centralized convex quadratic program (QP) algorithm shows real-time performance (below 0.05 s for 90% of the sample points) even for large teams of robots. On the other hand, the computational cost of the nonconvex optimization quickly grows with the number of robots, especially in the worst case. Note that computational time strongly depends on the number of neighbors considered in the collision avoidance, thus, the difference between the worst case and the 90% bar. Furthermore, timings of the MIQP approach are bounded by the maximum number of explored nodes, which we set to 200 nodes.

In Fig. 16(b), we compare the time to convergence in an antipodal position exchange, such as the one shown in Fig. 12. The distributed approach performs the worst, with a deadlock

TABLE I
COMPUTATIONAL TIME [MEAN ± STANDARD DEVIATION (MINIMUM, MAXIMUM)], PER AGENT, FOR EACH STEP OF THE DISTRIBUTED COLLISION AVOIDANCE ALGORITHM

	Noncooperative [ms]	Cooperative [ms]
Distributed velocity prediction	–	4.946 ± 1.380 [1.597, 10.868]
Pairwise VO linearization	0.061 ± 0.021 [0.010, 0.225]	2.476 ± 0.886 [0.028, 6.906]
Distributed convex QP solver (1)	0.978 ± 0.414 [0.350, 6.235]	0.512 ± 0.097 [0.320, 2.010]
Distributed grid search within convex region	0.387 ± 0.519 [0.081, 6.282]	0.396 ± 0.654 [0.050, 5.888]
Total distributed collision avoidance (2)	1.661 ± 0.841 [0.541, 7.322]	5.260 ± 0.854 [3.757, 10.043]

(1) Displayed for three neighbors, slowly increases with the number of robots. (2) Displayed for three neighbors, the number of pairwise VO linearizations equals the number of neighbors.

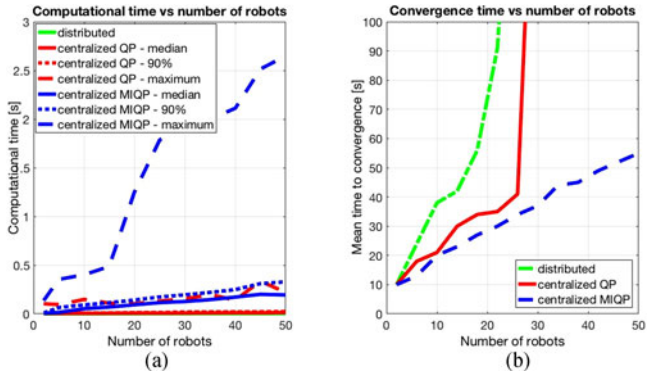

Fig. 16. Antipodal position exchange in a circle, for varying number of robots with bicycle kinematics. Comparison of distributed εCCA (light green), centralized convex QP εCCA (dark red), and centralized nonconvex MIQP εCCA (dark blue). (a) Computational time of the approach, where the green line is below the red line. (b) Mean time to convergence to the antipodal positions in the circle.

situation appearing for about 20 robots in our scenario. Both centralized approaches show similar performance for low number of robots. Nevertheless, for a large number of robots, the convex QP approach exhibits a deadlock behavior. The deadlock is due to the pairwise convexification of the collision avoidance constraints, which does not impose a global coordination to rotate clockwise or anticlockwise. On the other hand, the deadlock-resolution behavior is achieved in the nonconvex MIQP optimization thanks to the global exploration of pairwise avoidance topologies.

3) Cooperative Versus Reciprocal εCCA: Our εCCA method can be applied for both reciprocal, see Section IV-B, and cooperative, see Section IV-C, avoidance. Although in open spaces their performance is identical, the more general cooperative approach outperforms the reciprocal counterpart in situations where the "equal avoidance effort" assumption of the former does not hold. A representative situation is shown in Fig. 17, where the path and velocities of two robots navigating toward their respective goals (gray circles) are depicted. In the reciprocal case, the green robot blocks the red one, which requires an abrupt slowdown and change in orientation to avoid the collision. On the other hand, in the cooperative case, the green robot reasons about the static obstacle blocking the red robot and slightly slows down to let it pass in front in a more cooperative manner. The velocity partitions for this example are shown in Fig. 6.

In Table I, we show the computational times for the various steps of the distributed collision avoidance algorithm for the scenario of Fig. 17 with four robots. Both methods present low computational cost, enabling high-frequency loops. This is below 2 ms in mean for the noncooperative (constant velocity or reciprocal assumption) approach and about 5 ms in mean for the cooperative approach, due to the higher cost to compute the optimal pairwise distributed avoidance constraint described in Section IV-C. In the computation of the velocity prediction, obtaining the cost to go takes 13 ms in this map and is only executed, and stored for all start positions, when a new goal arrives. Although more general, the cooperative approach presents the challenge of computing a utility over the possible velocities that the other robot may take.

D. Lessons Learned

In our deployments, the distributed and reciprocal εCCA approach was mostly employed, since it provides good performance at a lower computational expense, as it scales better with the number of robots. The centralized and, especially, the nonconvex optimization can be beneficial in some particular situations where coordination is required, such as the antipodal position exchange of Fig. 12. Yet, after the optimal avoidance topology is chosen, both approaches perform similarly. In our experience, the convex and reciprocal approach performed well with relatively slow moving robots or in obstacle-free scenarios. Yet, for the robotic wheelchairs, which move at higher speed and in complex environments, the complete method of Section V-B2 is preferred, since it can take into account the nonconvexity in both the environment and the motion constraints.

In summary, for high-performing systems, such as the fast robotic wheelchairs, our recommended implementation is the distributed search within a convex region of Section V-B2, with cooperative avoidance. On the other hand, for large teams of relatively slow robots, such as the e-pucks, our recommended implementation is the distributed convex optimization of Section V-B1, with reciprocal avoidance, due to its computational simplicity.

In some cases, especially with a large number of robots and complex scenarios such as the antipodal position exchange, a fair amount of suboptimal small movements can be required

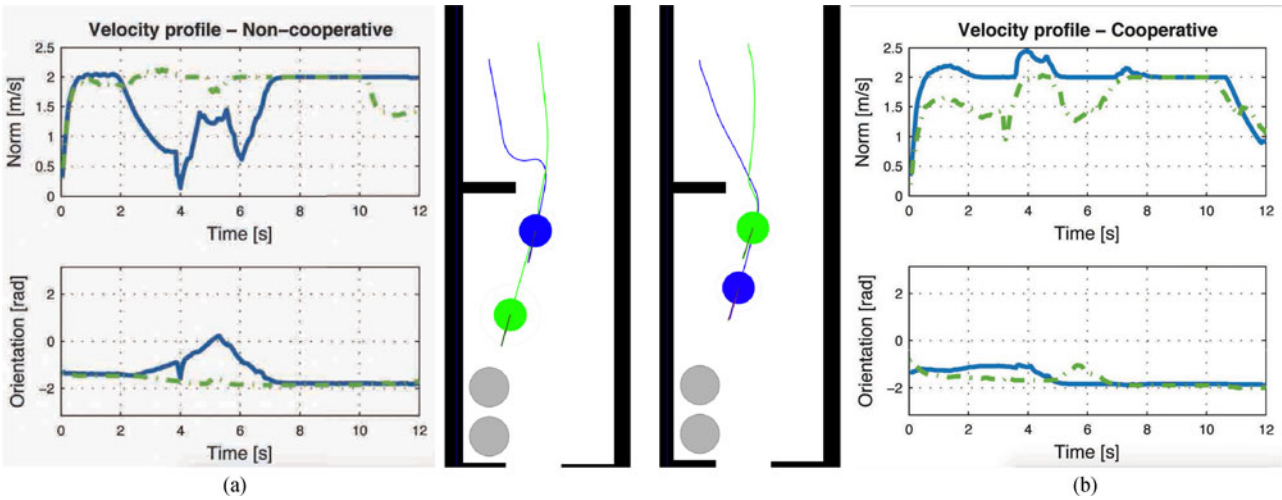

Fig. 17. Comparison of the εCCA reciprocal and cooperative approaches for computing the pairwise distributed avoidance constraints. Two robots (blue/green circles) navigate in a corridor (white = free space) toward their goals (gray circles). In the middle, path of the robots. In the sides, velocity profile and orientation over time for both robots during the interaction (dashed green and solid blue). (a) Reciprocal collision avoidance. (b) Cooperative collision avoidance.

before the robots reach a configuration where the goal can be reached relatively quickly. This is due to the localness of the collision avoidance method and the kinematic constraints of the robots, and can be observed in the accompanying experiment with 12 robots. A solution to improve performance could be to employ higher level reasoning or social norms.

VII. Conclusion

In this paper, we have described a method, namely εCCA, for collision avoidance among multiple robots in planar environments, which models inter-robot interaction and decision making. In particular, we extended the traditional VOs method to robots with nonholonomic kinematics and subject to arbitrary dynamic constraints. The idea was to reduce the set of local motions to those generated with an adequate controller toward a constant velocity reference trajectory. We discussed centralized and distributed, convex and nonconvex, implementations of the method and showed their tradeoffs in extensive experimental evaluations. The presented methods allow for smooth and safe navigation and good performance was observed in extensive experimental tests with various robot types. Further, the low computational cost of the algorithm allows for real-time control of hundreds of robots, or a fast control loop for single-robot navigation in dynamic environments.

Many challenges and avenues for future development still remain. Future work should aim at further improving the motion planning toward global reasoning, adaptation, and uncertain environments. In this paper, robots of arbitrary shape can be considered, but with the assumption of locally nonrotating during the time horizon of the local planner. Seamless integration of arbitrary shape, as well as on-board computation and sensing remain as future works. We believe that εCCA is well suited for onboard computation and sensing thanks to its low computational complexity, which allows for fast planning loops. The effect of social rules could also be explored.

Acknowledgment

The authors would like to thank A. Breitenmoser and M. Rufli for fruitful discussions and M. Del'Ambrogio, P. Gohl, L. Limacher, and C. Gwerder for their help with experiments.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Boston, MA, USA: Kluver, 1991.
[2] H. Choset *et al.*, *Principles of Robot Motion: Theory, Algorithms, and Implementations* (Intelligent Robotics and Autonomous Agents Series). Cambridge, MA, USA: MIT Press, 2005.
[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[4] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proc. 2002 IEEE Int. Conf. Robot. Autom.*, 2002, pp. 968–975.
[5] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, pp. 308–333, Mar. 2009.
[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, pp. 846–894, Jun. 2011.
[8] D. Hsu, R. Kindel, J.-C. Latombe, and S. M. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.
[9] J. P. Gonzalez and M. Likhachev, "Search-based planning with provable suboptimality bounds for continuous state spaces," *Proc. 4th Annu. Symp. Combinatorial Search*, May 2011, pp. 60–67.
[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
[11] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. 1999 IEEE Int. Conf. Robot. Autom.*, 1999, pp. 341–346.
[12] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 2210–2215.
[13] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, pp. 1251–1270, Aug. 2014.
[14] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *J. Guid., Control, Dyn.*, vol. 37, pp. 1725–1740, Nov. 2014.
[15] M. Cap, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[16] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1688–1694.
[17] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1047–1054, Apr. 2017.
[18] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017.
[19] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 797–803.
[20] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds—An experimental comparison," *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1341–1346.
[21] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Proc. Robot.: Sci. Syst. (RSS)*, Jul. 2012.
[22] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.
[23] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. Int. Symp. Robot. Res.*, 2009, pp. 3–19.
[24] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 5917–5922.
[25] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 5573–5578.
[26] E. Lalish and K. A. Morgansen, "Decentralized reactive collision avoidance for multivehicle systems," in *Proc. 47th IEEE Conf. Decis. Control*, Dec. 2008, pp. 1218–1224.
[27] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4584–4589.
[28] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," *Proc. Int. Symp. Distrib. Auton. Robot. Syst.*, 2010, pp. 203–216.
[29] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 360–366.
[30] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," *IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3475–3482.
[31] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal collision avoidance with motion continuity constraints," *IEEE Trans. Robot.*, Dec. 2013, pp. 1–14.
[32] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *Int. J. Robot. Res.*, vol. 34, pp. 1501–1514, Oct. 2015.
[33] A. Giese, D. Latypov, and N. M. Amato, "Reciprocally-rotating velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3234–3241.
[34] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1192–1198.
[35] J. van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "LQG-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 346–353.
[36] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Auton. Robots*, vol. 39, pp. 101–121, Jun. 2015.
[37] J. Alonso-Mora, M. Rufli, R. Siegwart, and P. Beardsley, "Collision avoidance for multiple agents with joint utility maximization," *IEEE Int. Conf. Robot. Autom.*, Mar. 2013, pp. 1–6.
[38] J. Alonso-Mora, P. Gohl, S. Watson, R. Siegwart, and P. Beardsley, "Shared control of autonomous vehicles based on velocity space optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1639–1645.
[39] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 42–49.
[40] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," *AIAA Guid., Navig., Control Conf. Exhibit.*, 2008, pp. 1–14.
[41] P. Fiorini and Z. Shillert, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.
[42] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*. Berlin, Germany: Springer-Verlag, 1998, pp. 171–253.
[43] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "Pledestrians: A least-effort approach to crowd simulation," *Proc. 2010 ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2010, pp. 119–128.
[44] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Proc. Neural Inf. Process. Syst.*, 2013, pp. 521–529.
[45] S. J. Guy *et al.*, "ClearPath: Highly parallel collision avoidance for multi-agent simulation," in *Proc. 2009 ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, New York, NY, USA, ACM, 2009, pp. 177–187.
[46] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple robots," *Int. J. Robot. Res.*, vol. 31, pp. 753–773, May 2012.
[47] OOQP, "Ooqp solver." [Online]. Available: https://github.com/emgertz/OOQP
[48] IBM CPLEX Optimizer, "IBM CPLEX optimizer." [Online]. Available: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
[49] S. Hauri, J. Alonso-Mora, A. Breitenmoser, R. Siegwart, and P. Beardsley, "Multi-robot formation control via a real-time drawing interface," in *Proc. Int. Conf. Field Serv. Robot.*, 2012, pp. 175–189.

Javier Alonso-Mora (M'17) received the M.Sc. and Ph.D. degrees in robotics from ETH Zurich, Zürich, Switzerland, in 2010 and 2014, respectively.

He is currently an Assistant Professor with Delft University of Technology, Delft, The Netherlands. Until October 2016, he was a PostDoctoral Associate with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. He was also a Member of Disney Research Zurich. His main research interests include autonomous navigation of mobile robots, with a special emphasis in multirobot systems and robots that interact with other robots and humans. Toward the smart cities of the future, he applies these techniques in various fields, including self-driving cars, automated factories, aerial vehicles, and intelligent transportation systems.

Dr. Alonso-Mora is the recipient of an NWO Veni Grant from The Netherlands Organisation for Scientific Research (2017), a Best Video Award at IEEE/ACM HRI 2014, and a nomination for Best Student Paper Award at DARS 2010.

Paul Beardsley received the Ph.D. degree in computer vision from University of Oxford, Oxford, U.K., in 1992.

He produced foundational work in three-dimensional computer vision with University of Oxford. Armed with formal training and industry experience in software engineering and an interest in robust real-time vision systems, he has crafted a research agenda that ranges from applied to fundamental. On the applied side, he uses vision for human interaction and for human logistics such as crowd counting. His more fundamental projects are in robotic vision, including the use of sensor swarms of heterogeneous mobile sensors to recover models of complex indoor and outdoor environments. His research work focuses on computer vision.

Dr. Beardsley was a corecipient of one of the R&D 100 Awards for the most significant inventions of 2006 in connection with his work on live registration of helicopter-mounted video with street maps, for use by emergency services.

Roland Siegwart (F'08) received the Doctoral degree in mechanical engineering or mechatronics from ETH Zurich, Zürich, Switzerland, in 1989.

He has been a Full Professor in autonomous systems with ETH Zurich since July 2006 and is currently a Founding Co-Director with Wyss Zurich, Zürich, Switzerland. From January 2010 to December 2014, he was the Vice-President Research and Corporate Relations in the Executive Board. He then spent one year as a Postdoctoral Fellow with Stanford University, Stanford, CA, USA. Back in Switzerland, he worked part time from 1991 to 1996 as an R&D Director with MECOS Traxler AG and as a Lecturer and a Deputy Head with the Institute of Robotics, ETH Zurich. In 1996, he was appointed as a Professor in autonomous microsystems and robots with the Ecole Polytechnique Fédérale de Lausanne. His research interests include design and control of systems operating in complex and highly dynamical environments.