Article

The International Journal of
Robotics Research
2015, Vol. 34(12) 1501–1514

DOI: 10.1177/0278364915576234
ijr.sagepub.com

Generalized reciprocal collision avoidance

Daman Bareiss[1] and Jur van den Berg[2]

Abstract

Reciprocal collision avoidance has become a popular area of research over recent years. Approaches have been developed for a variety of dynamic systems ranging from single integrators to car-like, differential-drive, and arbitrary, linear equations of motion. In this paper, we present two contributions. First, we provide a unification of these previous approaches under a single, generalized representation using control obstacles. In particular, we show how velocity obstacles, acceleration velocity obstacles, continuous control obstacles, and LQR-obstacles are special instances of our generalized framework. Secondly, we present an extension of control obstacles to general reciprocal collision avoidance for non-linear, non-homogeneous systems where the robots may have different state spaces and different non-linear equations of motion from one another. Previous approaches to reciprocal collision avoidance could not be applied to such systems, as they use a relative formulation of the equations of motion and can, therefore, only apply to homogeneous, linear systems where all robots have the same linear equations of motion. Our approach allows for general mobile robots to independently select new control inputs while avoiding collisions with each other. We implemented our approach in simulation for a variety of mobile robots with non-linear equations of motion: differential-drive, differential-drive with a trailer, car-like, and hovercrafts. We also performed physical experiments with a combination of differential-drive, differential-drive with a trailer, and car-like robots. Our results show that our approach is capable of letting a non-homogeneous group of robots with non-linear equations of motion safely avoid collisions at real-time computation rates.

Keywords

Collision avoidance, multi-robot system, decentralized control, mobile robot navigation, motion control

1. Introduction

Collision avoidance is a fundamental problem in robotics. The problem can generally be defined in the context of an autonomous mobile robot navigating in an environment with obstacles and/or other moving entities, where the robot employs a continuous sensing-control cycle. In each cycle, the robot must compute an action based on its local observations of the environment, such that it stays free of collisions with the moving obstacles and the other robots, and progresses towards a goal. Many works in robotics have addressed the problem of collision avoidance with moving obstacles (Fox et al., 1997; Fiorini and Shiller, 1998; Hsu et al., 2002; Petti and Fraichard, 2005). Typically, these approaches predict where the moving obstacles might be in the future by extrapolating their observed trajectories, and let the robot avoid collisions accordingly. Velocity obstacles (VO) (Fiorini and Shiller, 1998) formalize this principle by characterizing the set of velocities for the robot that result in a collision at some future time. Continually selecting a velocity outside of this set will then guarantee collision-free navigation for the robot.

However, such approaches are insufficient when the robot encounters other robots that also actively make decisions based on their surroundings: considering them as moving obstacles overlooks the fact that they react to the robot in the same way the robot reacts to them, and inherently causes suboptimal and oscillatory motion (Kluge and Prassler, 2004; Van den Berg et al., 2008).

This has lead to the development of *reciprocal collision avoidance* techniques, which specifically account for the reactive nature of the other robots without relying on coordination or communication among robots. The earliest approaches were direct extensions of VO, in which each robot is given half the responsibility of avoiding pairwise collisions (Van den Berg et al., 2008, 2009). Since this

[1]Department of Mechanical Engineering, University of Utah, UT, USA
[2]School of Computing, University of Utah, UT, USA

Corresponding author:
Daman Bareiss, Department of Mechanical Engineering, University of Utah, 50 S. Central Campus Dr., 2110 MEB, Salt Lake City, UT 841412, USA.
Email address: daman.bareiss@utah.edu

Table 1. Classification of reciprocal collision avoidance approaches.

	Homogeneous	Non-homogeneous
Linear	VO/ORCA AVO CCO LQR–obstacles	**Control obstacles**
Non-linear	**Control obstacles** **Control obstacles**	**Control obstacles**

approach only applies to robots with very simple dynamics that allow the robots to change their velocity instantaneously, most subsequent research on the topic has focused on extending the approach to robots with more complex dynamic constraints, such as differential-drive (Alonso-Mora et al., 2010; Snape et al., 2010), car-like (Alonso-Mora et al., 2012), double-integrator (Lalish and Morgansen, 2012; Van den Berg et al., 2012), arbitrary integrator (Ruffli et al., 2013), and robots with linear quadratic regulator (LQR) controllers (Bareiss and van den Berg, 2013). A major limitation of these approaches though is that all robots are assumed to have exactly the same equations of motion, in other words, they apply to *homogeneous* systems only. Moreover, in all these approaches the assumed equations of motion are *linear*, as these approaches rely on the ability to express the *relative* motion of pairs of robots in terms of the *relative* control input (i.e. the difference between the control inputs) of the robots. Hence, they do not apply to non-homogeneous or non-linear systems, where robots have different and/or non-linear equations of motion, which limits their applicability to real-world robots and in real-world applications.

In this paper, we address this shortcoming by presenting a new reciprocal collision avoidance method with two main contributions (see Table 1):

- First, we provide a unification of all previous approaches to reciprocal collision avoidance under a single, generalized representation using *control obstacles*. We will show specifically that approaches such as VO (Fiorini and Shiller, 1998), acceleration velocity obstacles (AVO) (Van den Berg et al., 2012), continuous control obstacles (CCO) (Ruffli et al., 2013), and LQR-obstacles (Bareiss and van den Berg, 2013) are each a special instance of our generalized framework. Moreover, we will show that our formulation is generally applicable to all homogeneous systems with linear equations of motion, and as such covers that entire class of systems.
- Second, we present an extension of control obstacles to reciprocal collision avoidance for general non-linear and/or non-homogeneous systems where the robots may have different state spaces and different non-linear equations of motion. No previous approaches to reciprocal collision avoidance could be applied to these categories of systems, even though some previous work has shown how specific instances of non-linear systems can be turned into a linear system formulation to which one of the previous approaches could be applied (see the next section for a more thorough discussion).

We implemented our approach in simulation for a variety of mobile robots with non-linear equations of motion: differential-drive, differential-drive with a trailer, car-like, and hovercrafts. We also performed physical experiments with a combination of differential-drive, differential-drive with a trailer, and car-like robots. Our results show that our approach is capable of letting a non-homogeneous group of robots with non-linear equations of motion safely avoid collisions at real-time computation rates.

The remainder of the paper is structured as follows. Section 2 reviews previous approaches to reciprocal collision avoidance. Section 3 formally defines the problem of reciprocal collision avoidance. Section 4 presents our generalized approach for homogeneous systems with linear equations of motion using control obstacles. Section 5 shows how the previous reciprocal collision avoidance approaches can be represented in our generalized approach. In Section 6 we explore the potential of our approach to non-homogeneous systems with non-linear equations of motion. Section 7 presents our results and Section 8 summarizes and concludes.

2. Previous work

One of the early developments in collision avoidance was the velocity obstacle (VO) (Fiorini and Shiller, 1998). The VO is defined as a cone in the velocity space based on relative positions and geometries which defines all relative velocities which will result in a collision. To avoid a collision, the robot needs to select a new velocity that lies outside the VO.

The approach of the VO was initially developed for a single active agent avoiding collisions with passive agents or moving obstacles. The approach was extended to perform reciprocal collision avoidance between two active agents in reciprocal velocity obstacles (RVO) (Van den Berg et al., 2008). In RVO, the concept of the VO is used but each robot must take half the responsibility to avoid collisions rather than the entire responsibility as in the VO algorithm. However, as the number of agents increases RVO tends to result in oscillatory motions. Optimal reciprocal velocity obstacles (ORCA) (Van den Berg et al., 2009) was developed to address this issue. In ORCA, the set of safe velocities is evenly divided between two robots by defining halfplanes of safe, possible velocities. These halfplanes are defined with respect to the VO and are the sets of individual velocities for two robots that result in *relative* velocities outside of the VO, thus avoiding collisions. Each robot selects a new velocity from the set of

safe, possible velocities that is as close as possible to a target velocity.

These algorithms all have the limitation that they are only guaranteed to provide safe, collision-free motion for robots with the linear equation of motion $\dot{\mathbf{p}} = \mathbf{v}$ where the position $\mathbf{p}$ defines the state and the velocity $\mathbf{v}$ defines the control input. This model is not practical for most real-world robots. Several extensions of these earlier approaches were introduced to address this issue and present reciprocal collision avoidance for more complicated dynamics. AVO (Van den Berg et al., 2012) were defined for robots with states consisting of position and velocity $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ with acceleration control inputs $\mathbf{u} = k(\mathbf{v}^{\star} - \mathbf{v})$ where k is some proportional gain, $\mathbf{v}^{\star}$ is some target velocity, and $\mathbf{v}$ is the current velocity. The approach in AVO was further generalized to apply to arbitrary-degree integrators through CCO by Ruffli et al. (2013). Further extension was provided by Bareiss and van den Berg (2013) for robots with arbitrary, homogeneous, linear equations of motion.

Table 1 summarizes the types of multi-robot system that have been considered in reciprocal collision avoidance based on a high-level categorization along two axes:

- *Homogeneous versus non-homogeneous systems:* *Homogeneous* teams of robots consist of robots that all have exactly the same state space and equations of motion, for example all robots are single-integrators with directly controllable velocity. *Non-homogeneous* teams of robots on the other hand may include robots with different state spaces and equations of motion, for example a single-integrator interacting with a double-integrator (a robot with directly controllable acceleration).
- *Linear versus non-linear equations of motion*: For systems with *linear* equations of motion, the derivative of the state is a linear function of the state and the control input, as is the case with single and double integrators for example. Differential-drive and car-like robots are examples of systems with *non-linear* equations of motion.

It turns out that all existing approaches to reciprocal collision avoidance are limited to *specific* instances of homogeneous systems with linear equations of motion, as these approaches are reliant upon the ability to express the equations of motion in terms of their current *relative* states and *relative* control inputs, which is generally not possible for non-homogeneous and/or non-linear systems. Our approach extends this previous work and is generally applicable across this two-dimensional spectrum.

There have been some developments for reciprocal collision avoidance for non-linear equations of motion. Reciprocal collision avoidance for differential-drive robots was performed by Snape et al. (2010) where the center of the robots was shifted and the bounding radius was increased in order to model the robots as holonomic with linear equations of motion. In Alonso-Mora et al. (2010), non-holonomic ORCA (NH-ORCA) was developed. NH-ORCA increases the radius of the robot based on the error in a tracking controller which allows non-holonomic robots to track holonomic trajectories as demonstrated for differential-drive robots. The NH-ORCA algorithm is applied to car-like robots by Alonso-Mora et al. (2012). In Van den Berg et al. (2012) it was shown how car-like robots can be represented as double integrators, to which AVO can be applied. These approaches all have in common that they transform specific instances of non-linear equations of motion into a linear formulation to which reciprocal collision avoidance can be applied. Our approach, in contrast, will apply directly to any general non-linear equations of motion.

Our work has some similarities to non-linear velocity obstacles (NLVO) (Shiller et al., 2001) and generalized velocity obstacles (GVO) (Wilkie et al., 2009). The NLVO algorithm expands the VO algorithm to allow for a robot with linear equations of motion to avoid collisions with passive obstacles moving with known, possibly non-linear trajectories. The self-motion velocity obstacle (SMVO) is another approach that utilizes the NLVO while considering more general robot trajectories (Shiller et al., 2008). The GVO algorithm does basically the opposite of the NLVO by defining a "control obstacle" for robots with non-linear equations of motion to avoid a passive obstacle moving along a linear trajectory. This approach samples the space of possible control inputs to determine if a collision will occur in the future. Neither of these approaches can be trivially extended to reciprocal collision avoidance.

3. Problem statement

3.1. Notation

We use the following notational conventions in this paper. Vector sets $\mathcal{A}$ are denoted using calligraphics, vectors $\mathbf{a}$ are denoted using boldface, matrices A are denoted using upper-case italics, and scalars a are denoted by lower-case italics. Scalar and matrix multiplication, and Minkowski sums of sets, are defined as

$$a\mathcal{X} = \{a\mathbf{x} | \mathbf{x} \in \mathcal{X}\}, \quad A\mathcal{X} = \{A\mathbf{x} | \mathbf{x} \in \mathcal{X}\}$$
$$\mathcal{X} \oplus \mathcal{Y} = \{\mathbf{x} + \mathbf{y} | \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}\}$$

It follows that $\mathcal{A} \oplus \{\mathbf{a}\}$ denotes a translation of a set $\mathcal{A}$ by a vector $\mathbf{a}$.

3.2. Problem setup

We consider multiple mobile robots sharing a common workspace where the robots have potentially different, non-linear equations of motion and state spaces. Let the state

space of robot i be $\mathcal{X}_i \subset \mathbb{R}^{n_i}$. Let $\mathbb{R}^d$ be the robots' physical workspace, where $d = 2$ or $d = 3$ typically. We assume the position $\mathbf{p}_i \in \mathbb{R}^d$ of robot i can be derived from its state $\mathbf{x}_i \in \mathcal{X}_i$ by some potentially non-linear projection function $\mathbf{q}_i \in \mathcal{X}_i \rightarrow \mathbb{R}^d$:

$$\mathbf{p}_i(t) = \mathbf{q}_i(\mathbf{x}_i(t)) \tag{1}$$

Let $\mathcal{O}_i \subset \mathbb{R}^d$ be the geometry of robot i relative to its position. We assume that the geometric shape of the robot is determined only by its position and not its orientation, in other words, it is rotationally invariant. More specifically, we consider the robot geometry as its bounding circle similar to the approach in the original VO (Fiorini and Shiller, 1998). This is a reasonable assumption for most mobile robots that greatly simplifies the development of our approach. See Giese et al. (2014) for work specifically including the orientation dimension in reciprocal collision avoidance.

We further assume that the dimension of the control input is equal to the dimension of the workspace, where $\mathcal{U}_i \subset \mathbb{R}^d$ is the valid control input space, which is assumed to be convex. Let the continuous-time equation of motion for robot i be given by a potentially non-linear function $\mathbf{f}_i \in \mathcal{X}_i \times \mathcal{U}_i \rightarrow \mathbb{R}^{n_i}$:

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t), \mathbf{u}_i(t)) \tag{2}$$

where $\mathbf{x}_i(t)$ is the state and $\mathbf{u}_i(t)$ is the control input at time t for robot i. It is important to remember that $\mathcal{X}_i$, $\mathbf{q}_i$, and $\mathbf{f}_i$ may be different for every robot i.

Given a current state $\mathbf{x}_i = \mathbf{x}_i(0)$ of robot i and some constant control input $\mathbf{u}_i = \mathbf{u}_i(0)$, the state of the robot at a given time $t > 0$ is given by

$$\mathbf{x}_i(t) = \mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i) \tag{3}$$

where $\mathbf{g}_i \in \mathbb{R} \times \mathcal{X}_i \times \mathcal{U}_i \rightarrow \mathcal{X}_i$ is the solution to the differential equation of equation (2), which can be obtained numerically, for example through a Runge–Kutta integration.

3.3. Problem statement

The problem of reciprocal collision avoidance we are addressing can now be defined as having each robot i independently compute a change $\Delta\mathbf{u}_i \in \mathcal{U}_i \oplus \{-\mathbf{u}_i\}$ of its current control input $\mathbf{u}_i$ given the current states $\mathbf{x}_j$ and control inputs $\mathbf{u}_j$ of all other robots $j \neq i$, such that the robots do not collide within a time horizon τ:

$$\begin{aligned} \forall (j \neq i, 0 \leq t < \tau) :: (\mathcal{O}_i \oplus \{\mathbf{q}_i(\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i + \Delta\mathbf{u}_i))\}) \cap \\ (\mathcal{O}_j \oplus \{\mathbf{q}_j(\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j + \Delta\mathbf{u}_j))\}) = \emptyset \end{aligned} \tag{4}$$

3.4. Challenges and assumptions

The challenge of reciprocal collision avoidance is that robot i does not know the change in control input $\Delta\mathbf{u}_j$ the other robots are going to choose. Therefore, we rely on the assumption that all robots use the same algorithm in order to select their change of control input. In this paper we discuss the design of an algorithm to compute changes in control inputs such that collision avoidance is achieved. In doing so, we make the following assumptions:

(I) When computing a control input, we assume that it remains constant over finite time τ into the future. The actual sensing-action cycle is much shorter than τ and a new control input is computed in every sensing-action cycle.

(II) We assume that the robots can fully observe each other's state and control input.

(III) We assume that the robots have the same type of control input, for example desired velocity, which is equal in dimension to the dimension of the workspace.

4. Generalized reciprocal collision avoidance for homogeneous, linear equations of motion

In this section we introduce the general concept of *control obstacles* that applies to general *linear* and *homogeneous* systems. The control obstacle generalizes all previous approaches on reciprocal collision avoidance, as we will show in Section 5. In Section 6 we present the extension to non-linear, non-homogeneous systems.

4.1. Control obstacles

In this section, we consider a system of robots that all have the same linear equations of motion, that is, a linear, homogeneous system. Equation (1) can be expressed for all robots i in a linear, homogeneous system as

$$\mathbf{p}_i(t) = \mathbf{q}(\mathbf{x}_i(t)) = C\mathbf{x}_i(t) + \mathbf{d} \tag{5}$$

where the matrix $C \in \mathbb{R}^{d \times n}$ and the vector $\mathbf{d} \in \mathbb{R}^d$ map a robot's state to its position and are identical for all robots.

Let the state space $\mathcal{X}_i = \mathcal{X} \subset \mathbb{R}^n$ be identical for all robots i. Equation (2) can be expressed for all robots i as

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}(\mathbf{x}_i(t), \mathbf{u}_i(t)) = A\mathbf{x}_i(t) + B\mathbf{u}_i(t) + \mathbf{c} \tag{6}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times d}$, and $\mathbf{c} \in \mathbb{R}^n$.

Given a current state $\mathbf{x}_i = \mathbf{x}_i(0)$ and a constant control input $\mathbf{u}_i = \mathbf{u}_i(0)$, solving the differential equation in equation (6) gives

$$\mathbf{x}_i(t) = \mathbf{g}(t, \mathbf{x}_i, \mathbf{u}_i) = F(t)\mathbf{x}_i + G(t)\mathbf{u}_i + \mathbf{h}(t) \tag{7}$$

where $F(t) \in \mathbb{R} \rightarrow \mathbb{R}^{n \times n}$, $G(t) \in \mathbb{R} \rightarrow \mathbb{R}^{n \times d}$, and $\mathbf{h}(t) \in \mathbb{R} \rightarrow \mathbb{R}^n$ are identical for every robot and are given as

$$\begin{bmatrix} F(t) & G(t) & \mathbf{h}(t) \\ 0 & I & 0 \\ 0 & 0 & 1 \end{bmatrix} = \exp\left(t \begin{bmatrix} A & B & \mathbf{c} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \tag{8}$$

Remember that for homogeneous systems with linear equations of motion, $\mathbf{q}$, $\mathbf{f}$, and $\mathbf{g}$ are the same for all robots. A unique property for these systems is that

$$\begin{aligned}\mathbf{g}(t,\mathbf{x}_i,\mathbf{u}_i)-\mathbf{g}(t,\mathbf{x}_j,\mathbf{u}_j) &= \mathbf{g}(t,\mathbf{x}_i-\mathbf{x}_j,\mathbf{u}_i-\mathbf{u}_j)\\ &= \mathbf{g}(t,\mathbf{x}_{ij},\mathbf{u}_{ij})\\ &= F(t)\mathbf{x}_{ij}+G(t)\mathbf{u}_{ij}+\mathbf{h}(t)\end{aligned} \tag{9}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ are the relative initial states and relative control inputs, respectively. This property has been exploited by all previous work on reciprocal collision avoidance. However, it does not hold for homogeneous systems with non-linear equations of motion, non-homogeneous systems with linear equations of motion, or non-homogeneous systems with non-linear equations of motion. We will discuss these cases in Section 6.

Substituting equation (7) into equation (5) for a given control input $\mathbf{u}_i + \Delta\mathbf{u}_i$, where $\mathbf{u}_i$ is the current control input and $\Delta\mathbf{u}_i$ is the change in control input, gives

$$\mathbf{p}_i(t) = \hat{\mathbf{p}}(t,\mathbf{x}_i,\mathbf{u}_i) + J(t)\Delta\mathbf{u}_i \tag{10}$$

where $\hat{\mathbf{p}}(t,\mathbf{x}_i,\mathbf{u}_i) \in \mathbb{R}^d$ and $J(t) \in \mathbb{R}^{d\times d}$ are

$$\hat{\mathbf{p}}(t,\mathbf{x}_i,\mathbf{u}_i) = C(F(t)\mathbf{x}_i + G(t)\mathbf{u}_i + \mathbf{h}(t)) + \mathbf{d} \tag{11}$$

$$J(t) = CG(t) \tag{12}$$

Given equation (10), we now define the control obstacle $\mathcal{UO}_{ij}$ for a robot i avoiding collisions with robot j. In order for the robots to avoid collision, their relative position $\mathbf{p}_{ij}(t) = \mathbf{p}_i(t) - \mathbf{p}_j(t)$ must remain outside of the Minkowski sum of the robots' geometries $\mathcal{O}_{ij} = \mathcal{O}_j \oplus -\mathcal{O}_i$:

$$\mathbf{p}_{ij}(t) \notin \mathcal{O}_{ij} \tag{13}$$

Substituting equation (10) into equation (13) and solving for the relative change in control input $\Delta\mathbf{u}_{ij}$ gives

$$\begin{aligned}&\hat{\mathbf{p}}(t,\mathbf{x}_{ij},\mathbf{u}_{ij}) + J(t)\Delta\mathbf{u}_{ij} \notin \mathcal{O}_{ij} \Rightarrow \Delta\mathbf{u}_{ij} \notin J(t)^{-1}\\ &\quad(\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}(t,\mathbf{x}_{ij},\mathbf{u}_{ij})\})\end{aligned} \tag{14}$$

Equation (14) represents a constraint on the change in relative control input $\Delta\mathbf{u}_{ij}$ such that robots i and j do not collide at time t. We define the control obstacle as the union of equation (14) for all time t less than the time horizon τ:

$$\mathcal{UO}_{ij} = \bigcup_{0\le t<\tau} J(t)^{-1}(\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}(t,\mathbf{x}_{ij},\mathbf{u}_{ij})\}) \tag{15}$$

In other words, a collision will not occur between robot i and robot j within τ time into the future when their relative change in control input $\Delta\mathbf{u}_{ij}$ lies outside the control obstacle:

$$\Delta\mathbf{u}_{ij} \notin \mathcal{UO}_{ij} \tag{16}$$

The geometry of $\mathcal{UO}_{ij}$ can be seen as a union of copies of the relative geometry $\mathcal{O}_{ij}$, each translated to $-\hat{\mathbf{p}}(t,\mathbf{x}_{ij},\mathbf{u}_{ij})$, that is, the nominal trajectory of robot j relative to robot i, and then transformed by J^{-1}. If the geometries of the robots are discs, $\mathcal{UO}_{ij}$ is hence a union of ellipsoids.

4.2. Avoiding collisions with passive robots

For a passive robot or environmental object, we can assume that $\Delta\mathbf{u}_j = \mathbf{0}$. That is, we assume the other robot does not change its control input. Avoiding collisions with that robot or object can then be performed simply by selecting a change in control input $\Delta\mathbf{u}_i$ outside the control obstacle:

$$\Delta\mathbf{u}_i \notin \mathcal{UO}_{ij} \tag{17}$$

For the case where it cannot be assumed that $\Delta\mathbf{u}_j = \mathbf{0}$, in other words, both robots are actively avoiding collisions, reciprocal collision avoidance must be performed, which we discuss next.

4.3. Reciprocal collision avoidance using control obstacles

Equation (16) gives the constraint on the relative change in control input $\Delta\mathbf{u}_{ij}$ for two robots to avoid collisions. When it cannot be assumed that $\Delta\mathbf{u}_j = \mathbf{0}$, robot i has to consider the change in control input $\Delta\mathbf{u}_j$ robot j is going to select in order for robot i to select a safe change in control input $\Delta\mathbf{u}_i$ for itself. The challenge is that $\Delta\mathbf{u}_j$ is unknown to robot i and the robots are not allowed to communicate. Hence, our approach is that robot i computes sets $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ of possible *safe* changes of control inputs for robot i and robot j, respectively, that satisfy the constraint

$$((\mathcal{RCA}_{ij} \cap \tilde{\mathcal{U}}_i) \oplus -(\mathcal{RCA}_{ji} \cap \tilde{\mathcal{U}}_j)) \cap \mathcal{UO}_{ij} = \emptyset \tag{18}$$

where $\tilde{\mathcal{U}}_i = \mathcal{U}_i \oplus \{-\mathbf{u}_i\}$ is the set of feasible changes in control input for robot i given the control input constraints. If robot i selects a change in control input $\Delta\mathbf{u}_i$ from $\mathcal{RCA}_{ij}$ and robot j selects a change in control input $\Delta\mathbf{u}_j$ from $\mathcal{RCA}_{ji}$, which each satisfy their respective control input constraints, then it is guaranteed that $\Delta\mathbf{u}_{ij} \notin \mathcal{UO}_{ij}$ and the robots will not collide within τ time in the future. We will let robot i compute $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ in such a way that if robot j were to apply the same algorithm to its situation, it would compute the same sets $\mathcal{RCA}_{ji}$ and $\mathcal{RCA}_{ij}$. Robot i is then free to choose any change in control input from the set $\mathcal{RCA}_{ij}$ to avoid collisions with robot j.

There are infinitely many pairs of sets of changes in control inputs $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ that satisfy equation (18). Therefore, we choose to find a pair of sets that divides the responsibility of avoiding collisions equally between both robots. Let us define a convex set $\mathcal{C}$ of safe *relative* changes in control inputs such that

$$(\mathcal{C} \cap \tilde{\mathcal{U}}_{ij}) \cap \mathcal{UO}_{ij} = \emptyset \tag{19}$$

where $\tilde{\mathcal{U}}_{ij} = \tilde{\mathcal{U}}_i \oplus -\tilde{\mathcal{U}}_j$ represents the feasible relative changes in control inputs. Any relative change in control

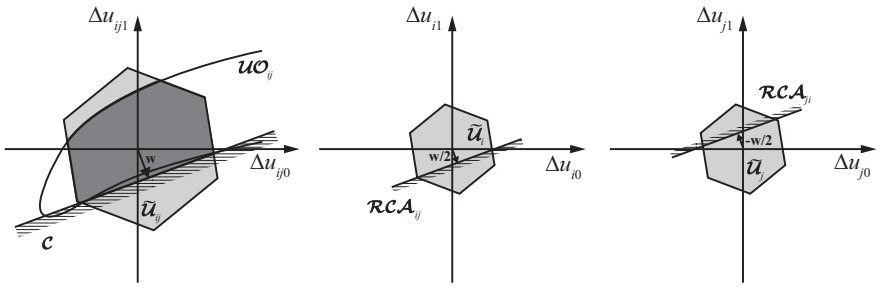

Fig. 1. On the left, a control obstacle $\mathcal{UO}_{ij}$ is given by its outline. The set of feasible relative changes in control input $\tilde{\mathcal{U}}_{ij}$ (i.e. those that adhere to the control input constraints) is shown as the light gray hexagon. The minimum change in relative control input required to avoid collision is shown as the vector $\mathbf{w}$ which defines the position of the halfspace $\mathcal{C}$. The vector $\mathbf{w}$ is defined as the closest point to the origin outside the *convex hull* (dark gray region) of the intersection of the control obstacle and the feasible changes in control input $\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}$. Each robot constructs a set of safe changes in control input, $\mathcal{RCA}_{ij}$ for robot i and $\mathcal{RCA}_{ji}$ for robot j, at $\mathbf{w}/2$ and $-\mathbf{w}/2$ respectively as shown in the middle and right images.

input $\Delta\mathbf{u}_{ij}$ that does not violate control input constraints and that is within $\mathcal{C}$, in other words any $\Delta\mathbf{u}_{ij} \in (\mathcal{C} \cap \tilde{\mathcal{U}}_{ij})$, will avoid collisions between robots i and j within τ time.

Since for convex sets it holds that $\mathcal{X} = \frac{1}{2}\mathcal{X} \oplus \frac{1}{2}\mathcal{X}$, the set $\mathcal{C}$ can be "halved" to determine sets of safe changes in control input for both robot i and robot j as

$$\mathcal{RCA}_{ij} = \frac{1}{2}\mathcal{C}, \qquad \mathcal{RCA}_{ji} = -\frac{1}{2}\mathcal{C} \tag{20}$$

where the desire to divide the responsibility for collision avoidance equally between the two robots motivates halving of the set $\mathcal{C}$ for each robot. We will define the convex set $\mathcal{C}$ more concretely below, but if it satisfies the condition of equation (19), we can prove that our definition of $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ in equation (20) satisfies our constraint on $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ of equation (18):

$$\begin{aligned}
&((\mathcal{RCA}_{ij} \cap \tilde{\mathcal{U}}_i) \oplus -(\mathcal{RCA}_{ji} \cap \tilde{\mathcal{U}}_j)) \cap \mathcal{UO}_{ij} \\
&\quad = \left(\left(\frac{1}{2}\mathcal{C} \cap \tilde{\mathcal{U}}_i\right) \oplus -\left(-\frac{1}{2}\mathcal{C} \cap \tilde{\mathcal{U}}_j\right)\right) \cap \mathcal{UO}_{ij} \\
&\quad \subseteq \left(\left(\frac{1}{2}\mathcal{C} \oplus \frac{1}{2}\mathcal{C}\right) \cap (\tilde{\mathcal{U}}_i \oplus -\tilde{\mathcal{U}}_j)\right) \cap \mathcal{UO}_{ij} \\
&\quad = (\mathcal{C} \cap \tilde{\mathcal{U}}_{ij}) \cap \mathcal{UO}_{ij} \\
&\quad = \emptyset
\end{aligned} \tag{21}$$

where we use the fact that

$$\begin{aligned}
(\mathcal{W} \cap \mathcal{X}) \oplus (\mathcal{Y} \cap \mathcal{Z}) = (\mathcal{W} \oplus \mathcal{Y}) \cap (\mathcal{W} \oplus \mathcal{Z}) \cap \\
(\mathcal{X} \oplus \mathcal{Y}) \cap (\mathcal{X} \oplus \mathcal{Z}) \\
\subseteq (\mathcal{W} \oplus \mathcal{Y}) \cap (\mathcal{X} \oplus \mathcal{Z})
\end{aligned}$$

What remains is choosing the convex set $\mathcal{C}$ of safe relative changes in control inputs. Ideally $\mathcal{C}$ should be the largest set of safe relative changes in control input, but such a set can be difficult to compute exactly. Therefore, we define $\mathcal{C}$ to be the halfspace tangent to the convex hull of the set of feasible relative control inputs that will result in a collision, that is, $\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})$, at the point $\mathbf{w}$ on the convex hull's boundary closest to the origin:

$$\mathbf{w} = \underset{\mathbf{u} \in \partial\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})}{\operatorname{argmin}} \|\mathbf{u}\| \tag{22}$$

$$\mathcal{C} = \begin{cases} \{\mathbf{u} | (\mathbf{u} - \mathbf{w}) \cdot \mathbf{w} \geq \mathbf{0}\} & \text{if } \mathbf{0} \in \mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}) \\ \{\mathbf{u} | (\mathbf{u} - \mathbf{w}) \cdot \mathbf{w} \leq \mathbf{0}\} & \text{if } \mathbf{0} \notin \mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}) \end{cases} \tag{23}$$

where ∂ refers to the boundary of a set. By construction, this definition of $\mathcal{C}$ satisfies equation (19).

This is illustrated in Figure 1. The set of feasible relative changes in control input $\tilde{\mathcal{U}}_{ij}$ (i.e. those that adhere to control input constraints) is represented by the light gray hexagon. The dark gray region represents the convex hull of the intersection of the feasible relative changes in control input and the control obstacle, that is, $\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})$. The set $\mathcal{C}$ is shown located tangent to this convex hull at the point closest to the origin. Placing the set $\mathcal{C}$ at the closest point to the origin represents the desire to keep the relative changes in control input as small as possible and, therefore, allow the robots to maintain their current, desired control input as closely as possible.

Since $\mathcal{C}$ is a halfspace, it follows that $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ are halfspaces as well. If the robots are currently on a collision course, that is, $\Delta\mathbf{u}_{ij} \in \mathcal{UO}_{ij}$, the vector $\mathbf{w}$ represents the smallest relative change in control input required to avoid a collision. Given that the two robots share the responsibility for avoiding collisions equally, the sets $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ are halfspaces located at $\frac{1}{2}\mathbf{w}$ and $-\frac{1}{2}\mathbf{w}$ from the origin of their respective control input spaces, as shown in Figure 1.

It is important to note that each robot i and j can independently compute their halfspaces $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ since the construction of the control obstacle from robot j's perspective $\mathcal{UO}_{ji}$ results in the same sets $\mathcal{RCA}_{ji}$ and $\mathcal{RCA}_{ij}$ since $\mathcal{UO}_{ij} = -\mathcal{UO}_{ji}$.

If both robots desire to keep their changes in control inputs as small as possible while ensuring they avoid collisions, each robot selects a change in control input as

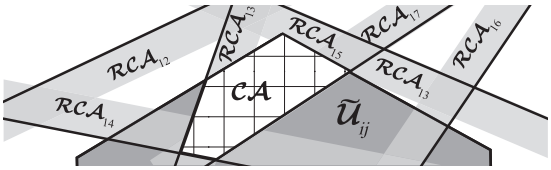

Fig. 2. A scenario for a group of seven robots avoiding collision. Robot 1 creates a safe set of changes in control inputs $\mathcal{RCA}_{1j}$ for every other robot j. The intersection of the union of these planes and the space of possible changes of control input $\tilde{\mathcal{U}}_{ij}$ is shown as $\mathcal{CA}$, which is the set of changes in control input that avoid collisions with every other robot while adhering to control input constraints.

$$\Delta\mathbf{u}_i^{\min} = \underset{\Delta\mathbf{u}_i \in \mathcal{RCA}_{ij}}{\operatorname{argmin}} \|\Delta\mathbf{u}_i\| \tag{24}$$

Given the symmetry of $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$, it follows that

$$\Delta\mathbf{u}_j^{\min} = -\Delta\mathbf{u}_i^{\min} \tag{25}$$

We use this observation later in the extension of the reciprocal collision avoidance approach to non-linear systems.

4.4. Avoiding collisions with multiple robots

A control obstacle is defined for pairwise collision avoidance but can easily be extended to more than two robots with an approach similar to Van den Berg et al. (2009). Each robot i creates a control obstacle and determines its set of safe potential changes in control input $\mathcal{RCA}_{ij}$ with respect to *every* other robot j, as in Figure 2. After considering every robot j, the change in control input for robot i is selected that is safe from all collisions:

$$\Delta\mathbf{u}_i \in \tilde{\mathcal{U}}_i \cap \bigcap_{j \neq i} \mathcal{RCA}_{ij} \tag{26}$$

where $\Delta\mathbf{u}_i$ can be found with a convex optimization method in the d-dimensional space of control inputs similar to the method by Van den Berg et al. (2009).

Given a preferred change in control input, the convex optimization will result in a change in control input that is as close as possible to some preferred input while not violating equation (26). However, there can be cases in which the set of safe changes in control input may be empty, that is, $\tilde{\mathcal{U}}_i \cap \bigcap_{j \neq i} \mathcal{RCA}_{ij} = \emptyset$. When this occurs, a convex optimization in a $(d + 1)$-dimensional space will select the change of control input that will *least* violate the constraints. See Van den Berg et al. (2009) for full details. This potentially results in a collision within τ time if the control inputs truly remain constant, but given that a new control input is selected in each sensing-action cycle, in practice this turns out to typically result in safe motion. However, the fact that collision avoidance can only be theoretically guaranteed in some cases remains a limitation of our approach.

5. Generalization of previous reciprocal collision avoidance approaches

Above, we have developed a method for reciprocal collision avoidance for a homogeneous system of multiple robots with general, linear equations of motion. We did so through a new method of control obstacles. Previous approaches of reciprocal collision avoidance can be shown to be special cases of control obstacles. As shown in the previous sections, the control obstacle is fully defined for a system if given A, B, and $\mathbf{c}$ from equation (6) and C and $\mathbf{d}$ from equation (5). We will now show how previous methods of reciprocal collision avoidance can be represented as control obstacles using these terms.

5.1. VO

The VO algorithm assumes the robot's equations of motion are a single integrator kinematic model:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{27}$$

where $\mathbf{x} = \mathbf{p}$ and $\mathcal{X} \subset \mathbb{R}^2$ is the space of positions, $\mathbf{u} = \mathbf{v}$ and $\mathcal{U} \subset \mathbb{R}^2$ is the space of velocities. For equation 27 we find

$$A = 0, \qquad B = I, \qquad \mathbf{c} = \mathbf{0}, \qquad C = I, \qquad \mathbf{d} = \mathbf{0}$$

Solving equation (7) for these, we find

$$F(t) = I, \qquad G(t) = tI, \qquad \mathbf{h}(t) = \mathbf{0}$$

When $\mathcal{O}_{ij}$ is a circle or sphere, the control obstacle is equivalent to the VO translated by the negative of the current relative input $-\mathbf{v}_{ij}$ as in Figure 3. This discrepancy arises from control obstacles being developed based on *changes* in control input rather than the absolute control input.

5.2. AVO

The AVO algorithm is presented by Van den Berg et al. (2012) as an alternative to the VO. One of the major problems with the VO is the assumption that instantaneous changes in velocity are possible. However, as this is not the case for physical systems, the AVO algorithm was developed.

In AVO, the robots have four state variables (the two-dimensional position and velocity) $\mathbf{x} \in \mathbb{R}^4$ and two control inputs (the two-dimensional acceleration) $\mathbf{u} \in \mathbb{R}^2$. The control inputs are driven by a proportional controller:

$$\dot{\mathbf{v}} = \frac{1}{\delta}(\mathbf{v}^\star - \mathbf{v}) \tag{28}$$

where $\mathbf{v}^\star$ is the desired velocity, $\mathbf{v}$ is the current velocity, and δ is a controller parameter.

Integrating equation (28) twice to obtain the system's trajectory gives

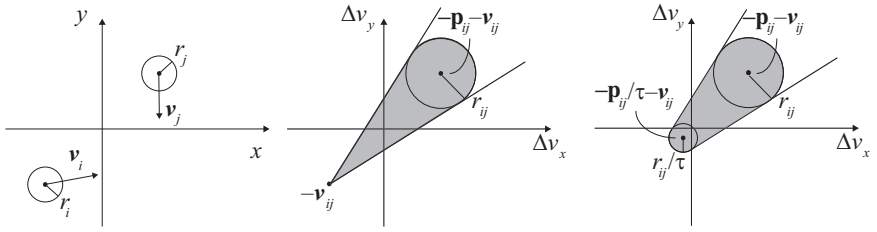

Fig. 3. Left: a pair of robots with equation of motion $\dot{\mathbf{p}} = \mathbf{v}$ where their current velocities will lead to a collision course with each other. Middle: an infinite time horizon control obstacle is given for the robot configuration on the left. As can be seen, the control obstacle contains the origin, meaning that the robots will indeed collide if they continue with their current control input. Right: the same control obstacle is shown, except now it is bounded by a finite time horizon τ. This control obstacle is equivalent to the VO for the single-integrator dynamics except it is shifted by the negative of the current relative velocity $-\mathbf{v}_{ij}$. This discrepancy between the control obstacle and VO arises because the control obstacle is defined in terms of the change in velocity rather than the absolute velocity.

$$\mathbf{v}(t) = \mathbf{v}^{\star} - e^{-t/\delta}\left(\mathbf{v}^{\star} - \mathbf{v}(0)\right) \tag{29}$$

$$\mathbf{p}(t) = \mathbf{p}(0) - \delta(e^{-t/\delta} - 1)\mathbf{v}(0) + \left(t + \delta(e^{-t/\delta} - 1)\right)\mathbf{v}^{\star} \tag{30}$$

Hence, we can represent the AVO with a control obstacle by choosing the state $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, the control input $\mathbf{u} = \mathbf{v}^{\star}$, and

$$A = \begin{bmatrix} 0 & I \\ 0 & -\frac{1}{\delta}I \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{1}{\delta}I \end{bmatrix}, \qquad \mathbf{c} = \mathbf{0}$$

$$C = [I \quad 0], \qquad \mathbf{d} = \mathbf{0}$$

Solving equation (7), we find

$$F(t) = \begin{bmatrix} 1 & -\delta(e^{-t/\delta} - 1) \\ 0 & e^{-t/\delta} \end{bmatrix}, \quad G(t) = \begin{bmatrix} t + \delta(e^{-t/\delta} - 1) \\ 1 - e^{-t/\delta} \end{bmatrix}, \quad \mathbf{h}(t) = \mathbf{0}$$

5.3. CCO

CCO (Ruffli et al., 2013) generalizes the previously mentioned AVO algorithm by defining the $(n + 1)$th derivative of position as the low-level control input where

$$\mathbf{p}^{(n+1)} = -c_n\mathbf{p}^{(n)} - \ldots - c_2\ddot{\mathbf{p}} + c_1(\mathbf{v}^{\star} - \dot{\mathbf{p}}) \tag{31}$$

where the superscript (n) represents the nth derivative of the term and $\mathbf{v}^{\star}$ is the target velocity that is a high-level control input.

The low-level control input is an input given directly to the robot which determines the next state through the equations of motion. The high-level control input abstracts the low-level control input to velocity through a controller. This abstraction to velocity as a control input is common in reciprocal collision avoidance and we discuss its use in our method in Section 6.1.

For example, controlling the jerk of a robot $\mathbf{p}^{(3)}$ is shown in full in Ruffli et al. (2013).

The state is $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\dot{\mathbf{p}}^{\mathrm{T}} \ldots (\mathbf{p}^{(n)})^{\mathrm{T}}]^{\mathrm{T}}$ and the control input is $\mathbf{u} = \mathbf{v}^{\star}$. Solving for the state equation gives

$$A = \begin{bmatrix} 0 & I & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & I \\ 0 & -c_1 I & \ldots & -c_n I \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ c_1 \end{bmatrix} \quad \mathbf{c} = \mathbf{0}$$

$$C = [I \quad 0 \quad \ldots \quad 0], \qquad \mathbf{d} = \mathbf{0}$$

5.4. LQR-obstacles

LQR-obstacles by Bareiss and van den Berg (2013) provide a method for reciprocal collision avoidance for homogeneous systems of robots with the same arbitrary linear equations of motion. The equations of motion of each robot are

$$\dot{\mathbf{x}} = \tilde{A}\mathbf{x} + \tilde{B}\mathbf{u} + \tilde{\mathbf{c}} \tag{32}$$

An LQR controller is used to obtain the low-level input from the high-level input $\mathbf{v}^{\star}$ using the control law

$$\mathbf{u} = -L\mathbf{x} + E\mathbf{v}^{\star} + \ell \tag{33}$$

By substituting equation (33) into equation (32), the closed-loop equations of motion are given as

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{v}^{\star} + \mathbf{c} \tag{34}$$

where

$$A = \tilde{A} - \tilde{B}L, B = \tilde{B}E, \mathbf{c} = \tilde{B}\ell + \tilde{\mathbf{c}} \tag{35}$$

Along with equation (34), the control obstacle is fully defined for a given C and $\mathbf{d}$ that extract the position from the state, similar to equation (5).

6. Non-homogeneous, non-linear equations of motion

The previous discussion defined a generalized method for reciprocal collision avoidance using control obstacles $\mathcal{UO}_{ij}$ for sets of robots with the same linear equations of motion. We present the extension of these methods for robots in

non-homogeneous systems, that is, different types, with possibly non-linear equations of motion.

Given the equations of motion of robots i and j (see equation (2)), we can approximate the relative position $\mathbf{p}_{ij}(t)$ given each robot's current state $\mathbf{x}_i$ and $\mathbf{x}_j$ as well as constant control inputs $\mathbf{u}_i$ and $\mathbf{u}_j$ through a first-order Taylor approximation about the current control input:

$$\mathbf{p}_{ij}(t) \approx \mathbf{q}_i[\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i)] - \mathbf{q}_j[\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j)] + \frac{\partial(\mathbf{q}_i \circ \mathbf{g}_i)}{\partial \mathbf{u}_i}(t, \mathbf{x}_i, \mathbf{u}_i)\Delta\mathbf{u}_i - \frac{\partial(\mathbf{q}_j \circ \mathbf{g}_j)}{\partial \mathbf{u}_j}(t, \mathbf{x}_j, \mathbf{u}_j)\Delta\mathbf{u}_j \tag{36}$$

In the spirit of equation (25), we make the assumption that $\Delta\mathbf{u}_i \approx \Delta\mathbf{u}_{ij}/2$ and $\Delta\mathbf{u}_j \approx -\Delta\mathbf{u}_{ij}/2$ such that each robot is required to take half the responsibility for avoiding pairwise collisions. For this assumption to be realistic, we require that the control input of both robots be of the same "type", for example a desired velocity, which we will discuss in Section 6.1. We can then re-write equation (36) as

$$\mathbf{p}_{ij}(t) \approx \hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) + J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)\Delta\mathbf{u}_{ij} \tag{37}$$

where

$$\hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) = \mathbf{q}_i[\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i)] - \mathbf{q}_j[\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j)] \tag{38}$$

$$J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) = \frac{1}{2}\left(\frac{\partial(\mathbf{q}_i \circ \mathbf{g}_i)}{\partial \mathbf{u}_i}(t, \mathbf{x}_i, \mathbf{u}_i) + \frac{\partial(\mathbf{q}_j \circ \mathbf{g}_j)}{\partial \mathbf{u}_j}(t, \mathbf{x}_j, \mathbf{u}_j)\right) \tag{39}$$

Given the definitions of equations (38) and (39), the control obstacle can be given similar to before as

$$\mathcal{UO}_{ij} = \bigcup_{0<t<\tau} J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)^{-1}(\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)\}) \tag{40}$$

The methods for performing reciprocal collision avoidance with this new control obstacle formulation (equation (40)) are identical to those defined in Sections 4.3 and 4.4.

6.1. Higher-level control input

We have presented a method for reciprocal collision avoidance for robots in non-homogeneous systems with general, non-linear equations of motion. In doing so, we have made three key assumptions, which are:

(I) The control input remains constant over finite time τ;
(II) The robots observe each other's state and control input;
(III) The robots have the same type of input, equal in dimension to the workspace.

Many robots have control inputs which violate some or all of these assumptions. Let us consider a car-like robot with control inputs of acceleration at the rear axle and the steering angle. It is not reasonable to assume that these remain constant for long periods of time as required by (I). Of course, it will not remain constant because it changes every sensing-action cycle, but at least we want the constant assumption to give a reasonable estimate of the future motion of the other robots. For low-level control inputs that can change quickly (unlike a goal velocity), it cannot be assumed that a constant control input gives a reasonable estimate. It is also unreasonable to assume that these low-level control inputs can be observed by the other robots, violating (II). Performing reciprocal collision avoidance between a car-like robot and a differential-drive robot would violate (III).

For these reasons, we implement a controller which abstracts the low-level control inputs to a high-level control input, such as a target velocity, similar to Van den Berg et al. (2012), Bareiss and van den Berg (2013), and Ruffli et al. (2013). Abstracting to a high-level input makes the assumptions reasonable for most mobile robots. A target velocity typically remains approximately constant over long periods of time. A velocity is inherently equal to the dimension of the workspace. Lastly, it is reasonable to assume the current velocity of other robots can be observed, and it can be assumed the target velocity is approximately equal to the current velocity.

7. Results

We performed both simulations and physical experiments to verify the performance of the algorithm. In this section, we present the equations of motion for the robots used in the simulations and physical experiments as well as the result from those experiments. Each robot presented uses a controller to define a target velocity as a higher-level control input.

7.1. Robot dynamics

7.1.1. Differential-drive robot. We implemented a differential-drive robot in both simulation and experiments. We used the kinematic model with a three-dimensional state consisting of the two-dimensional position and the orientation (x, y, θ). The low-level control inputs are the left and right wheel velocities (v_r, v_l). The equations of motion are given as

$$\dot{x} = (v_r + v_l)\cos(\theta)/2 \tag{41}$$

$$\dot{y} = (v_r + v_l)\sin(\theta)/2 \tag{42}$$

$$\dot{\theta} = (v_r - v_l)/\ell \tag{43}$$

where ℓ is the distance between the wheels.

The low-level control inputs v_r and v_l are abstracted to a high-level input $\mathbf{v}^\star$ through a controller where

$$v_r = \|\mathbf{v}^\star\| + \ell k(\angle\mathbf{v}^\star - \theta)/2 \tag{44}$$

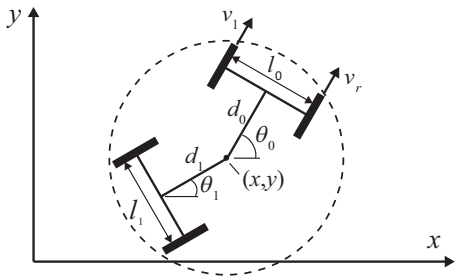

Fig. 4. The configuration of a differential-drive robot pulling a trailer with the origin (x, y) considered to be the point of connection between the robot and the trailer.

$$v_l = \left\|\mathbf{v}^\star\right\| - \ell k(\angle\mathbf{v}^\star - \theta)/2 \tag{45}$$

where $\angle\mathbf{v}^\star$ is the angle the target velocity makes with the positive x-axis and k is a controller gain. Substituting the new high-level control inputs from
'equations (44) and (45) into equations (41) to (43) gives

$$\dot{x} = \left\|\mathbf{v}^\star\right\| \cos(\theta), g t \dot{y} = \left\|\mathbf{v}^\star\right\| \sin(\theta), g t \dot{\theta} = k(\angle\mathbf{v}^\star - \theta) \tag{46}$$

7.1.2. Differential-drive with off-axle trailer. We implemented a differential-drive robot pulling a trailer in both simulation and experiments. The equations of motion were adapted from Lee et al. (2004) which uses a car with an off-axle trailer. For the configuration given in Figure 4, the state is given as $(x, y, \theta_0, \theta_1)$ and the equations of motion are given as

$$\dot{x} = (v_r + v_l)\cos(\theta_0)/2 + \dot{\theta}_0 d_0 \sin(\theta_0) \tag{47}$$

$$\dot{y} = (v_r + v_l)\sin(\theta_0)/2 - \dot{\theta}_0 d_0 \cos(\theta_0) \tag{48}$$

$$\dot{\theta}_0 = (v_r - v_l)/\ell_0 \tag{49}$$

$$\dot{\theta}_1 = \left(\frac{v_r + v_l}{2}\sin(\theta_0 - \theta_1) - \dot{\theta}_0 d_0 \cos(\theta_0 - \theta_1)\right)/d_1 \tag{50}$$

where d_0, d_1, and ℓ_0 are the parameters shown in Figure 4.

We implemented a controller such that

$$\frac{v_r + v_l}{2} = \left\|\mathbf{v}^\star\right\|, \qquad \dot{\theta}_0 d_0 = k(\angle\mathbf{v}^\star - \theta_0) \tag{51}$$

which when substituting equation 51 into equations (47) to (50) gives

$$\dot{x} = \left\|\mathbf{v}^\star\right\| \cos(\theta_0) + k(\angle\mathbf{v}^\star - \theta_0)\sin(\theta_0) \tag{52}$$

$$\dot{y} = \left\|\mathbf{v}^\star\right\| \sin(\theta_1) - k(\angle\mathbf{v}^\star - \theta_0)\cos(\theta_0) \tag{53}$$

$$\dot{\theta}_0 = k(\angle\mathbf{v}^\star - \theta_0)/d_0 \tag{54}$$

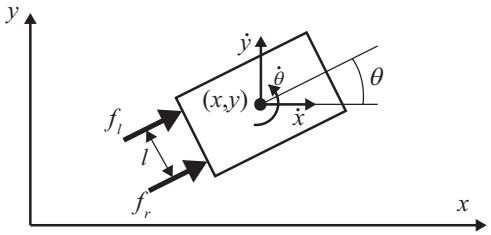

Fig. 5. The model of the hovercraft-like robot implemented in simulation. The two thrusters are shown as f_r and f_l with the distance between them as ℓ. The center position and orientation are shown.

$$\dot{\theta}_1 = (\left\|\mathbf{v}^\star\right\| \sin(\theta_0 - \theta_1) - k(\angle\mathbf{v}^\star - \theta_0)\cos(\theta_0 - \theta_1))/d_1 \tag{55}$$

7.1.3. Car-like robot. We implemented a car-like robot in simulation and physical experiments. We used a four-dimensional state consisting of the two-dimensional position, the orientation, and the speed (x, y, θ, v). The low-level control inputs are the acceleration at the rear axle and the steering curvature (a, κ) where equations of motion are defined at the midpoint by

$$\dot{x} = v\cos(\theta) - \ell v\kappa \sin(\theta)/2 \tag{56}$$

$$\dot{y} = v\sin(\theta) + \ell v\kappa \cos(\theta)/2 \tag{57}$$

$$\dot{\theta} = v\kappa \tag{58}$$

$$\dot{v} = a \tag{59}$$

where ℓ is the distance between the front and rear wheels. Deriving the equations of motion in terms of the midpoint of the robot, rather than at the midpoint along the rear axle keeps the enclosing disc as small as possible.

We implemented a proportional controller to determine the low-level control inputs in terms of the target velocity $\mathbf{v}^\star$:

$$a = k_0(\left\|\mathbf{v}^\star\right\| - v), \qquad \kappa = k_1\ell(\angle\mathbf{v}^\star - \theta)/v \tag{60}$$

where k_0 and k_1 are proportional controller gains. Substituting equation 60 into equations (56) to (59) gives

$$\dot{x} = v\cos(\theta) - k_1(\angle\mathbf{v}^\star - \theta)\sin(\theta)/2 \tag{61}$$

$$\dot{y} = v\sin(\theta) + k_1(\angle\mathbf{v}^\star - \theta)\cos(\theta)/2 \tag{62}$$

$$\dot{\theta} = k_1(\angle\mathbf{v}^\star - \theta) \tag{63}$$

$$\dot{v} = k_0(\left\|\mathbf{v}^\star\right\| - v) \tag{64}$$

7.1.4. Hovercraft. We implemented a simulated hovercraft-style robot with two thrusters as seen in Figure 5. The hovercraft's state is given as the two-dimensional position, the heading, the two-dimensional velocity, and the rate of change of the heading $(x, y, \dot{x}, \dot{y}, \theta, \dot{\theta})$. Given the forces

provided by thrusters (f_r, f_l), the midpoint equations of motion are given as

$$\ddot{x} = ((f_r + f_l)\cos(\theta) - b_t\dot{x})/m \tag{65}$$

$$\ddot{y} = ((f_r + f_l)\sin(\theta) - b_t\dot{y})/m \tag{66}$$

$$\ddot{\theta} = \left((f_r - f_l)\ell/2 - b_r\dot{\theta}\right)/\iota \tag{67}$$

where m is the mass of the robot, ι is the robot's rotational inertia, b_t is the translational friction coefficient, and b_r is the rotational friction coefficient.

Including a proportional-derivative controller to solve for control inputs as a function of the target velocity $\mathbf{v}^\star$ gives

$$(f_r + f_l) = k_0 m(\left\|\mathbf{v}^\star\right\| - \|\mathbf{v}\|) \tag{68}$$

$$(f_r - f_l)\ell/2 = k_1\iota(\angle\mathbf{v}^\star - \theta) - k_2\dot{\theta} \tag{69}$$

where k_0 and k_1 are controller gains, $\mathbf{v} = (\dot{x}, \dot{y})$ is the velocity, and $\angle\mathbf{v}^\star$ is the angle the target velocity makes with the x-axis. Substituting equations (68) and (69) into equations (65) to (67) gives

$$\ddot{x} = k_0(\left\|\mathbf{v}^\star\right\| - \|\mathbf{v}\|)\cos(\theta) - b_t\dot{x}/m \tag{70}$$

$$\ddot{y} = k_0(\left\|\mathbf{v}^\star\right\| - \|\mathbf{v}\|)\sin(\theta) - b_t\dot{y}/m \tag{71}$$

$$\ddot{\theta} = k_1(\angle\mathbf{v}^\star - \theta) - (k_2 + b_r/\iota)\dot{\theta} \tag{72}$$

7.2. Simulation setup and implementation details

We performed simulations on a desktop machine running Windows 7 Professional 64-bit with an Intel i7-2600 CPU (3.40 GHz) and 8 GB RAM. The simulations were developed in a Visual Studio C++ environment. The frequency of the sensing-action control sequence was 10 Hz. The equations of motion were discretized using Runge–Kutta integration at 0.1 s time-steps.

The relative geometry $\mathcal{O}_{ij}$ was approximated using a set of 16 points uniformly sampled around a circle of the robots' combined radii. The control obstacle can then be approximated by performing the operations in equation (14) on the generated set of points for each time-step up to the time horizon τ. The convex hull of the control obstacle was computed using the Boost library (Dawes et al., 2009). Upon determining the halfplanes, the RVO2-2D library (Van den Berg et al., 2009) was used to compute the new control input through a convex optimization method.

The differential-drive robots had bounding circle radii of 0.3 m. The car-like robots had radii of 0.45 m. The hovercraft robots had radii of 0.47 m. The differential-drive robots with trailers had radii of 0.45 m. The desired speed of the robots during the simulations was 0.3 m/s.

Unless otherwise noted, we used a time horizon of $\tau = 7$ s during simulations. This value was determined experimentally. We found the selection of the time horizon to have a significant impact on the performance of our algorithm. Too short a time horizon can lead to a "late" reaction from the robots. This can lead to situations where a rather large change in control input is necessary to avoid collisions. If this large input violates the control input constraints on the robot, the collision-avoiding input is not obtainable and a collision can occur. On the other hand, selecting τ to be too large has a negative effect as well. The halfplanes from equation (18) become more restrictive as τ increases, possibly leading to no solution for equation (26). The time horizon is an empirical term that is situation-dependent, which is a limitation of our algorithm. We note that for the specific case of single integrator dynamics, Gal et al. (2009) have performed systematic analysis on the optimal value of the time horizon.

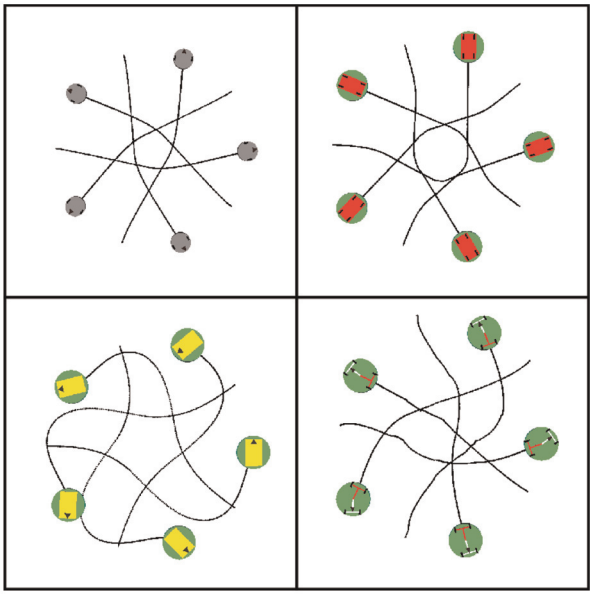

Fig. 6. Simulations where five robots avoid collisions while crossing the workspace. Top left: differential-drive robots. Top right: car-like robots. Bottom left: hovercraft-like robots. Bottom right: differential-drive robots with off-axis trailers.

7.3. Simulation results

We performed a variety of simulations to validate our approach. One set of simulations consisted of groups of five robots, where each robot type was simulated separately as shown in Figure 6. We ran simulations with all the robot types included. One such simulation included two of each type, eight in total. A selection of screenshots is shown in Figure 7. We included a simulation where two groups of four tried to cross the workspace as shown in Figure 8. We also performed a simulation where a group of four passive robots cross the workspace in a vertical line while a group of four active robots cross in the opposite direction. The four passive robots do not update their control input based on the positions of the other robots. Selected screenshots of this are shown in Figure 9. We performed a simulation

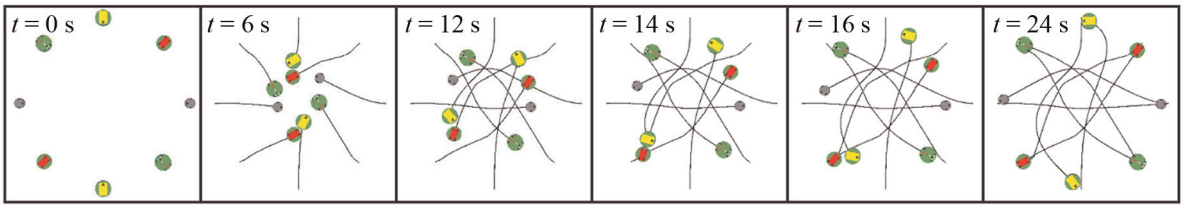

Fig. 7. A simulation that contains eight robots: two differential-drive (red discs), two differential-drives with trailers (red and white), two car-like robots (red rectangles), and two hovercrafts (yellow rectangles). They begin on a circle and cross the circle to finish on the side opposite their starting positions. Six screen shots from the simulation with the individual robot paths are shown.

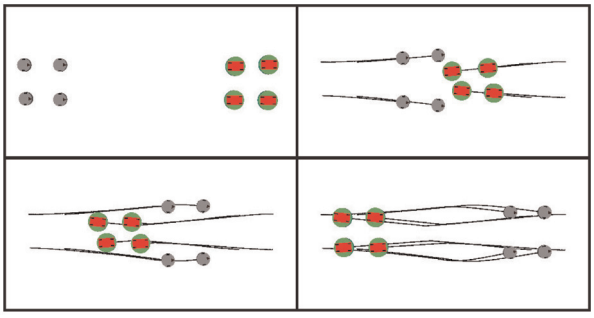

Fig. 8. Two groups of four robots crossing the workspace while avoiding collisions with each other.

with 100 robots moving from random starting positions to random goal positions. This simulation can be seen in the video found at http://arl.cs.utah.edu/research/grca/.

In physical experiments, it cannot be assumed that the robots can have perfect state estimation of the other robots. A series of simulations were performed to demonstrate our algorithm's robustness in the presence of noise. A group of eight differential-drive robots were initialized in a circle and their goal was to cross to the antipodal position. In these simulations, we introduced artificial noise by adding a normal random variable to the position components of $\mathbf{x}_j$ in equation (14). Increasing the robot's radius is a common practice in reciprocal collision avoidance methods, and therefore we increased our robots' radii by 10% (0.03 m). Using a time horizon of $\tau = 5$ s, no collisions were observed until the standard deviation of the sensed position was 0.07 m. We repeated the experiments with the radii increased by 25% (0.075 m) and observed collisions at a standard deviation of 0.15 m. Similar simulations were run for the same scenario with two of each robot type (differential-drive, differential-drive with a trailer, car-like, and hovercraft). In this case, the algorithm was less robust to noise with collisions resulting from standard deviations of 0.02 m and 0.04 m for bounding radii increases of 10% and 25%, respectively. This is likely due to the more constrained input space $\widetilde{\mathcal{U}}_{ij}$ for the more complicated dynamics as well as our use of very simple controllers for complex equations of motion. These experiments, as we expected, suggest that for larger noise values a larger increase in the bounding circle can be used. However, too large a bounding circle makes for extremely conservative actions which may be too limiting for a given robot's control input constraints.

In order to quantify the speed of our algorithm, we calculated the per-time-step-per-robot average computation time, that is, the time it takes for one robot to determine its set of safe changes in velocity with respect to every other robot in a single sensing-action cycle. As can be seen in Figure 10, this quantity is linear with respect to the number of robots, as expected. In simulation, we found that it is possible for over 100 non-homogeneous, non-linear robots to perform reciprocal collision avoidance at real-time computational rates for a time horizon of $\tau = 20$ s with a simulation frequency of 10 Hz. At higher frequencies, for example, 20 Hz, the trends in Figure 10 would have a slope of twice that for 10 Hz, due to the doubled frequency during the integration in equation (15). As can be seen, the computation time also shows a linear trend with the value

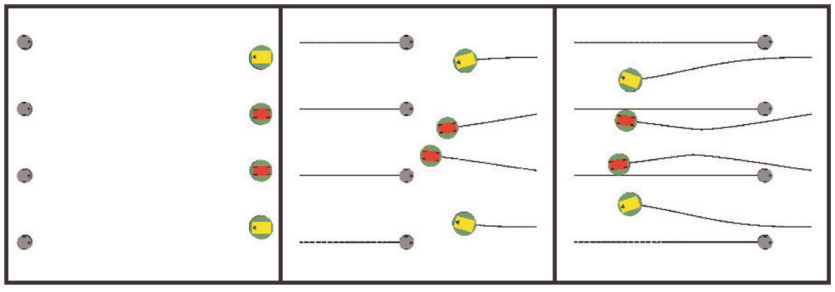

Fig. 9. A case where two *active* car-like robots and two *active* hovercraft robots cross the workspace from right to left as four *passive* differential-drive robots cross from left to right. The paths the robots take are drawn and it can be seen that the car-like and hovercraft robots make the necessary adjustments to avoid collisions with each other and the differential-drive robots.

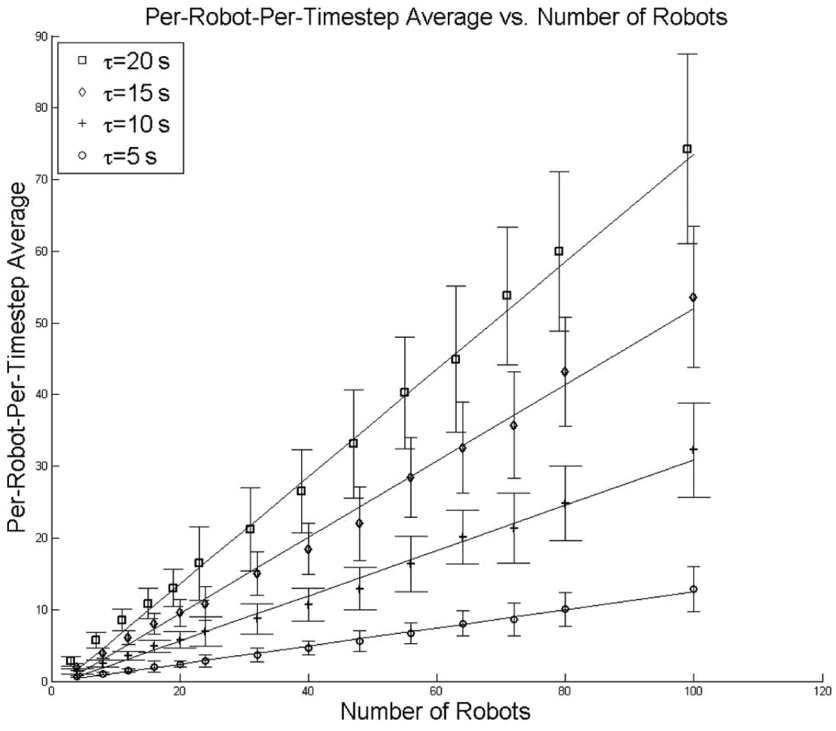

Fig. 10. Timing calculations were made for a number of experiments shown above. The data and the first-order fit are shown. For a sensing-action cycle frequency of 10 Hz, a single time-step is 0.1 s and our algorithm can produce real-time results for over 100 robots.

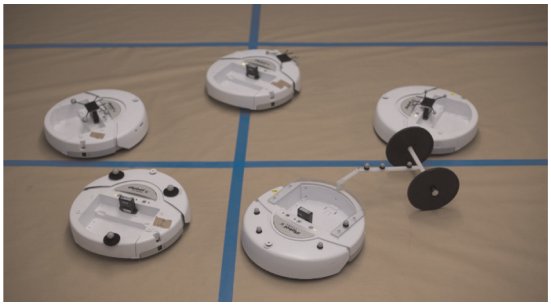

Fig. 11. An image taken while performing the physical experiments of our reciprocal collision avoidance algorithm.

of the time horizon, as twice as large a time horizon results in twice as complex a control obstacle (at least in the way we implemented its construction).

7.4. Experiments

The physical experiments were performed using the Robot Operating System (ROS) software platform. A motion-capture environment was used to estimate the positions and orientations of the robots. Similar to the simulations, the control obstacles were approximated using sets of 16 points uniformly sampled around the robots' bounding circles. While the state estimates of the robots are performed by the motion capture system, the algorithm is developed as a decentralized method where on-board state estimation could be implemented in the future.

For the experiments, we used six iRobot Creates. Three of the Creates were used as differential-drive robots, two were used to simulate car-like motion by restricting their minimum turning radius, and the sixth Create had a custom trailer mounted to it. The Creates have a radius of 0.335 m. The trailer axle is located 0.25 m behind the center of the Create. The robots were driven with a desired speed of 0.2 m/s. Their maximum speed possible is 0.5 m/s. The experiments were performed with a frequency of 50 Hz and a time horizon of 3.5 s.

Due to the stochastic nature of the experiments from modeling and sensor error, the robots' bounding circle was increased by 25%. Less accurate models or less accurate sensors could require a further increase in the radius. At times the robots can be seen moving back and forth between each other in a form of "reciprocal dance" due to sensing noise. This phenomenon has been more thoroughly studied by Conroy et al. (2014).

During the experiments, we recorded the desired velocity before the control obstacles algorithm was performed as well as the collision-free target velocity resulting from the control obstacles. To further quantify the experimental results, we determined the Euclidean norm between the desired and the calculated target velocities, representing the change in input from the algorithm. From approximately 5500 data points the mean and standard deviation of the set were found to be 0.0794 m/s and 0.128 m/s, respectively.

We ran experiments similar to the simulations with the robots crossing through the center of the workspace and avoiding collisions as shown in Figure 11. We also

performed experiments of situations more applicable to real-world scenarios where the robots were divided into two groups crossing the workspace. Videos of the experiments and simulations can be found at http://arl.cs.utah.edu/research/grca/.

8. Conclusions

Previously, reciprocal collision avoidance has been applied to a variety of robotic systems with both linear and special cases of non-linear equations of motion. In these approaches, the fact that all the robots were the same type allowed for a relative state formulation to be used. However, for non-homogeneous systems, that is, systems of robots that are different types, this is not possible.

In this paper, we presented a unified method for reciprocal collision avoidance of non-homogeneous systems of robots with non-linear equations of motion. In order to do so, we presented the control obstacle for homogeneous systems of robots with linear equations of motion. We then showed how the control obstacle generalizes previous reciprocal collision avoidance methods and provided examples of how previous methods fit into our framework. More specifically, we showed VO (Fiorini and Shiller, 1998), AVO (Van den Berg et al., 2012), CCO (Ruffli et al., 2013), and LQR-obstacles (Bareiss and van den Berg, 2013). Finally, we extended control obstacles for use with non-linear equations of motion and/or non-homogeneous systems. In our simulations and physical experiments, we saw that our algorithm was able to provide smooth, collision-free motion for all robots in the environment.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

Alonso-Mora J, Breitenmoser A, Beardsley P, et al. (2012) Reciprocal collision avoidance for multiple car-like robots. In: *IEEE international conference on robotics and automation*.

Alonso-Mora J, Breitenmoser A, Rufli M, et al. (2010) Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: *Proceedings of the 10th international symposium on distributed autonomous robotic systems*.

Bareiss D and van den Berg J (2013) Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles. In: *IEEE international conference on robotics and automation*.

Conroy P, Bareiss D and Beall M (2014) 3-D reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning. Available at: http://arxiv.org/abs/1411.3794.

Dawes B, Abrahams D and Rivera R (2009) Boost C++ libraries. Available at: http://www.boost.org.

Fiorini P and Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research* 17: 760–772.

Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1): 23–33.

Gal O, Shiller Z and Rimon E (2009) Efficient and safe on-line motion planning in dynamic environments. In: *IEEE international conference on robotics and automation*.

Giese A, Latypov D and Amato N (2014) Reciprocally-rotating velocity obstacles. In: *IEEE international conference on robotics and automation*.

Hsu D, Kindel R, Latombe J, et al. (2002) Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research* 21(3): 233–255.

Kluge B and Prassler E (2004) Reflective navigation: Individual behaviors and group behaviors. In: *IEEE international conference on robotics and automation*.

Lalish E and Morgansen K (2012) Distributed reactive collision avoidance. *Autonomous Robots* 32(3): 207–226.

Lee JH, Chung W, Kim M, et al. (2004) A passive multiple trailer system with off-axle hitching. *International Journal of Control, Automation, and Systems* 2(3): 289–297.

Petti S and Fraichard T (2005) Safe motion planning in dynamic environments. In: *IEEE RSJ international conference on intelligent robots and systems*.

Ruffli M, Alonso-Mora J and Siegwart R (2013) Reciprocal collision avoidance with motion continuity constraints. *IEEE Transactions on Robotics* 29(4): 899–911.

Shiller Z, Large F and Sekhavat S (2001) Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In: *IEEE international conference on robotics and automation*.

Shiller Z, Prasanna R and Salinger J (2008) *A unified approach to forward and lane-change collision warning for driver assistance and situational awareness*. Technical Report 2008-01-0204, Society of Automotive Engineers, Warrendale, PA.

Snape J, van den Berg J, Guy S, et al. (2010) Smooth and collision-free navigation for multiple robots under differential-drive constraints. In: *IEEE international conference on intelligent robots and systems*.

Van den Berg J, Guy S, Lin M, et al. (2009) Reciprocal *n*-body collision avoidance. In: *Proceedings of the international symposium of robotics research*.

Van den Berg J, Ling M and Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: *IEEE international conference on robotics and automation*.

Van den Berg J, Snape J, Guy S, et al. (2012) Reciprocal collision avoidance with acceleration-velocity obstacles. In: *IEEE international conference on robotics and automation*.

Wilkie D, van den Berg J and Manocha D (2009) Generalized velocity obstacles. In: *IEEE international conference on robotics and systems*.