2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Reciprocal Collision Avoidance for Robots with Linear Dynamics using LQR-Obstacles

Daman Bareiss Jur van den Berg

Abstract— **In this paper we present a formal approach to reciprocal collision avoidance for multiple mobile robots sharing a common 2-D or 3-D workspace whose dynamics are subject to linear differential constraints. Our approach defines a protocol for robots to select their control input independently (i.e. without coordination with other robots) while guaranteeing collision-free motion for all robots, assuming the robots can perfectly observe each other's state. To this end, we extend the concept of LQR-Obstacles (which is a generalization of Velocity Obstacles to robots with dynamics for collision avoidance among static obstacles) for reciprocal collision avoidance among multiple robots. We implemented and tested our approach in 3-D simulation environments for reciprocal collision avoidance of quadrotor helicopters, which have complex dynamics in 16-D state spaces. Our results show that our approach enables collision avoidance among over a hundred quadrotors in tight workspaces at real-time computation rates.**

I. Introduction

Collision avoidance is a fundamental problem in (mobile) 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 [5], [6], [8], [15]. However, such approaches are insufficient for multi-robot settings, where the robot encounters other robots that also make decisions based on their surroundings: considering them as moving obstacles overlooks the fact that they react to the robot in the same way as the robot reacts to them, and inherently causes undesirable oscillations in the motion of the robots [19]. Specifically accounting for the reactive nature of the other robots while not relying on coordination among robots is called *reciprocal* collision avoidance, in which robots are typically given half the responsibility of avoiding pairwise collisions. However, approaches that in fact guarantee collision avoidance have so far been limited to robots with specific and simple dynamics, such as holonomic [20], differential-drive [1], [16], car-like [2], and double-integrator [10], [21] robots.

In this paper, we present an approach for reciprocal collision avoidance for multiple robots with arbitrary linear dynamics. Our approach takes an adapted, relative formulation of the concept of *LQR-Obstacles* [22] (which is a generalization of Velocity Obstacles [5] to robots with linear dynamics) and extends it to reciprocal collision avoidance among multiple robots, by following a generalization of the approach of [21] (which only applied to robots with double-integrator dynamics). Our approach works for any number of robots with linear or linearizable dynamics and state spaces of any dimension in both 2-D and 3-D environments, and allows robots to fully navigate independently (without explicit coordination with other robots, unlike [18]) while guaranteeing collision-free motion for all robots, assuming the robots can perfectly observe each other's state.

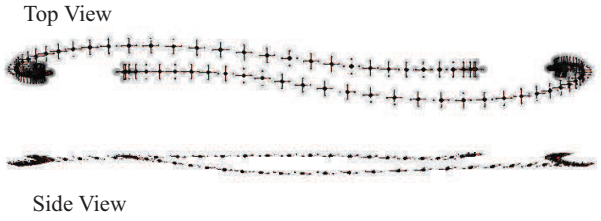


Fig. 1. Simulation results of two quadrotors reciprocally avoiding collisions while exchanging positions with each other. The quadrotors choose to avoid collisions by passing beside each other.

While our approach is designed for linear dynamics systems in general, we will demonstrate the potential of our approach on quadrotor helicopters, which operate in 3-D environments and have complex underactuated dynamics in 16-D state spaces with a 4-D control input. Algorithms and controllers have been developed allowing quadrotors to fly aerobatic maneuvers, perching and landing, avoid collisions with static and moving obstacles, fly in formation, and collaboratively manipulate objects [4], [11]–[14], [17]. While some of these results involved multiple quadrotors flying in a common workspace, their motions were typically centrally coordinated to make sure that collisions among quadrotors (or situations in which quadrotors fly in each other's downwash [9]) are avoided. Purely independent navigation, where each quadrotor observes its environment by itself and makes autonomous control decisions that guarantee collision avoidance with other quadrotors without mutual communication or coordination (much like humans do while walking on campus), has to date not been achieved. Our simulation results indicate that our approach can successfully compute smooth, collision-avoiding, and goal-directed motions for more than one hundred quadrotors in tight environments. Also, as each quadrotor computes its control inputs independently, the computations can be performed in

Daman Bareiss is with the Department of Mechanical Engineering at the University of Utah. E-mail: daman.bareiss@utah.edu.

Jur van den Berg is with the School of Computing at the University of Utah. E-mail: berg@cs.utah.edu.

parallel and at real-time rates.

The rest of this paper is organized as follows. In Section III we review LQR-Obstacles, and in Section IV this concept is extended for reciprocal collision avoidance among multiple robots. We discuss implementation details and simulation results in Section V, and conclude in Section VI.

II. Notation and Definitions

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 in lower-case italics. Scalar and matrix multiplication, and Minkowski sums of sets are defined as:

$$a\mathcal{X} = \{a\mathbf{x} \mid \mathbf{x} \in \mathcal{X}\}, \qquad A\mathcal{X} = \{A\mathbf{x} \mid \mathbf{x} \in \mathcal{X}\}, \quad (1)$$

$$\mathcal{X} \oplus \mathcal{Y} = \{\mathbf{x} + \mathbf{y} \mid \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}\}. \quad (2)$$

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

We define the following concepts. Let $\mathcal{X} \subset \mathbb{R}^n$ be the *state space* of the robots, and let $\mathbb{R}^d$, where typically $d = 2$ or $d = 3$, be the physical workspace the robots operate in. Let us assume, with some loss of generality, that the position $\mathbf{p}_i \in \mathbb{R}^d$ and the velocity $\mathbf{v}_i \in \mathbb{R}^d$ of robot i are part of its state $\mathbf{x}_i \in \mathcal{X}$, and that they are obtained by $\mathbf{p}_i = C\mathbf{x}_i$ and $\mathbf{v}_i = V\mathbf{x}_i$, respectively, for given projective matrices $C \in \mathbb{R}^{d\times n}$ and $V \in \mathbb{R}^{d\times n}$. Further, let us assume that only the position of the robot determines its geometric appearance in the workspace (and not its orientation, for instance), and let $\mathcal{O}_i \subset \mathbb{R}^d$ be the geometry of robot i relative to its reference point (its position). Let $\mathcal{U} \subset \mathbb{R}^m$ be the valid control input space of the robots, which we assume is convex and defined by the intersection of a set of linear constraints:

$$\mathcal{U} = \textstyle\bigcap_k \{\mathbf{u} \mid \mathbf{a}_k^T\mathbf{u} < b_k\}, \quad (3)$$

where $\mathbf{a}_k \in \mathbb{R}^m$ and $b_k \in \mathbb{R}$. Lastly, let the dynamics of the robots be given by a deterministic discrete-time linear model:

$$\mathbf{x}_i[t+1] = A\mathbf{x}_i[t] + B\mathbf{u}_i[t], \quad (4)$$

where $\mathbf{x}_i[t] \in \mathcal{X}$ is the state and $\mathbf{u}_i[t] \in \mathcal{U}$ is the control input of robot i at time t. Matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are constant and the same for all robots.

These assumptions are reasonable for mobile robots, which we focus on this paper. For a quadrotor helicopter, for instance, the state may consist of its position, velocity, orientation, and angular velocity, its control input may be the thrust of each of the four rotors whose constraints are defined by a maximum and minimum thrust for each of the rotors, and its geometry can be described by an enclosing sphere, such that its orientation is irrelevant for its geometric appearance. A linear dynamics model can reasonably be obtained for a quadrotor by linearizing its true dynamics about the hover point.

III. Collision Avoidance with LQR-Obstacles

The concept of LQR-Obstacle was introduced in [22] for collision-avoidance of robots with linear dynamics with static obstacles. It was defined as the set of target positions for the robot that will result in a collision with the obstacles if the robot is LQR-controlled towards that target. In this section, we derive *relative* LQR-Obstacles that will be used for reciprocal collision avoidance. It is defined here in terms of the set of target *velocities* that will result in collision, rather than the set of target positions.

A. LQR Feedback Control

Let $\mathbf{v}_i^\star \in \mathbb{R}^d$ denote a target velocity robot i wishes to reach. For systems with linear dynamics of Eq. (4), an infinite-horizon LQR feedback controller can optimally control the robot towards this target velocity given a quadratic cost function that trades-off reaching the target quickly versus not applying extreme control inputs:

$$\textstyle\sum_{t=0}^{\infty}((V\mathbf{x}_i[t] - \mathbf{v}_i^\star)^T Q(V\mathbf{x}_i[t] - \mathbf{v}_i^\star) + \mathbf{u}_i[t]^T R\mathbf{u}_i[t]), \quad (5)$$

where V maps a robot's state to its velocity, and $Q \in \mathbb{R}^{d\times d}$ and $R \in \mathbb{R}^{m\times m}$ are given constant weight matrices, for which $Q = Q^T \geq 0$ and $R = R^T > 0$.

The feedback control policy that minimizes Eq. (5) can be derived as a function of the target velocity:

$$\mathbf{u}_i[t] = -L\mathbf{x}_i[t] + E\mathbf{v}_i^\star, \quad (6)$$

where

$$L = (R + B^TSB)^{-1}B^TSA, \quad E = (R + B^TSB)^{-1}B^TT,$$

and $S = S^T \geq 0$ and T are solutions to the equations:

$$S = V^TQV + A^TSA - A^TSB(R + B^TSB)^{-1}B^TSA,$$
$$T = V^TQ + A^TT - A^TSB(R + B^TSB)^{-1}B^TT.$$

We can construct the *closed-loop* dynamics of the robots in terms of its target velocity rather than its low-level control input, by substituting Eq. (6) into (4):

$$\mathbf{x}_i[t+1] = \tilde{A}\mathbf{x}_i[t] + \tilde{B}\mathbf{v}_i^\star, \quad (7)$$

where

$$\tilde{A} = A - BL, \qquad \tilde{B} = BE.$$

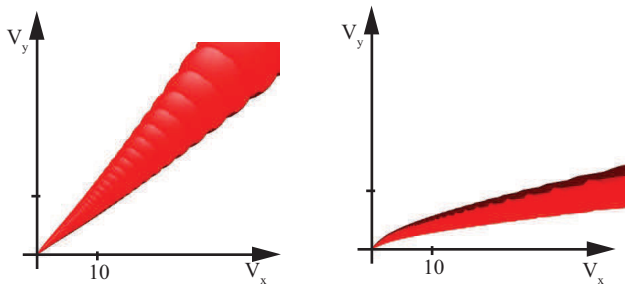
Given a current state $\mathbf{x}_i = \mathbf{x}_i[0]$ of robot i and a constant target velocity $\mathbf{v}_i^\star$, the state of the robot at a given time $t > 0$ is then given by solving the difference equation that defines the closed-loop dynamics (Eq. (7)):

$$\mathbf{x}_i[t] = F[t]\mathbf{x}_i + G[t]\mathbf{v}_i^\star, \quad (8)$$

where

$$F[t] = \tilde{A}^t, \qquad G[t] = \textstyle\sum_{k=0}^{t-1}\tilde{A}^k\tilde{B}.$$

With the closed-loop dynamics, the target velocity $\mathbf{v}_i^\star$ can be seen as a higher-level form of control input. Since it is more reasonable to assume that $\mathbf{v}_i^\star$ will stay constant over a short period of time than the low-level control input $\mathbf{u}_i$, we use the closed-loop dynamics to define (relative) LQR-Obstacles below and use it for collision-avoidance.



(a) xz-projection of *LQR-Obstacle* (b) xy-projection of *LQR-Obstacle*

Fig. 2. The LQR-Obstacle $\mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_{ij})$ for two quadrotors i and j. Robot i was given a current position of $\mathbf{p}_i = (0,0,0)^T$, and a current velocity of $\mathbf{v}_i = (-2,0,0)^T$. Robot j was given a current position of $\mathbf{p}_j = (4,6,-4)^T$, and a current velocity of $\mathbf{v}_j = (-1,-2,2)^T$. The time horizon τ in this example is 6 seconds.

B. Relative LQR-Obstacles

Let us look at a pair of robots i and j. The state of robot i *relative* to the state of robot j is denoted $\mathbf{x}_{ij}[t]$, and defined as $\mathbf{x}_{ij}[t] = \mathbf{x}_i[t] - \mathbf{x}_j[t]$. The robots i and j *collide* if their relative position $C\mathbf{x}_{ij}[t]$ (recall that C maps a robot's state to its position) is contained within the Minkowski difference $\mathcal{O}_{ij} = \mathcal{O}_j \oplus -\mathcal{O}_i$ of the robot's geometries, i.e.:

$$C\mathbf{x}_{ij}[t] \in \mathcal{O}_{ij}. \tag{9}$$

Let the relative target velocity of robots i and j be defined as $\mathbf{v}^{\star}_{ij} = \mathbf{v}^{\star}_i - \mathbf{v}^{\star}_j$. Given a current relative state $\mathbf{x}_{ij}$ of the robots and a constant relative target velocity $\mathbf{v}^{\star}_{ij}$, the relative state of the robots at a given time $t > 0$ is similar to Eq. (8) and given by $\mathbf{x}_{ij}[t] = F[t]\mathbf{x}_{ij} + G[t]\mathbf{v}^{\star}_{ij}$, which follows from the fact that the dynamics are linear and the same for all robots. Substituting this into Eq. (9), robots i and j collide at time t if:

$$\begin{aligned} & CF[t]\mathbf{x}_{ij} + CG[t]\mathbf{v}^{\star}_{ij} \in \mathcal{O}_{ij} \\ \iff \quad & \mathbf{v}^{\star}_{ij} \in (CG[t])^{-1}(\mathcal{O}_{ij} \oplus \{-CF[t]\mathbf{x}_{ij}\}), \end{aligned} \tag{10}$$

with the assumption that $CG[t]$ is an invertible matrix. Now, the *relative LQR-Obstacle* $\mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_{ij})$ of robots i and j with current relative state $\mathbf{x}_{ij}$ is defined as the set of all relative target velocities $\mathbf{v}^{\star}_{ij}$ that result in a collision within τ time into the future:

$$\mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_{ij}) = \bigcup_{t=1}^{\tau} (CG[t])^{-1}(\mathcal{O}_{ij} \oplus \{-CF[t]\mathbf{x}_{ij}\}). \tag{11}$$

Hence, if the geometry of each of the robots is defined by a sphere (or an ellipsoid), the relative LQR-obstacle is a union of ellipsoids (see Fig. 2).

C. Avoiding Collisions with Passive Robots

The relative LQR-obstacle as defined above can be used by robot i to avoid collisions with a robot j as follows: assuming that the current state of robot i is $\mathbf{x}_i$ and that robot i can observe the current state $\mathbf{x}_j$ of robot j and knows robot j's target velocity $\mathbf{v}^{\star}_j$, robot i must choose its target velocity $\mathbf{v}^{\star}_i$ *outside* the relative LQR-obstacle translated by j's target velocity in order to avoid collisions (within τ time):

$$\mathbf{v}^{\star}_i \notin \mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_i - \mathbf{x}_j) \oplus \{\mathbf{v}^{\star}_j\}. \tag{12}$$

While it may be reasonable to assume that robot i can estimate robot j's current state (e.g., using a Kalman filter) it is not reasonable to assume that robot i knows robot j's intent, i.e. its target velocity. In this case, robot i can make the assumption that robot j's current velocity is its target velocity, i.e. $\mathbf{v}^{\star}_j \approx V\mathbf{x}_j$, and choose its own target velocity accordingly. This is a reasonable approach if robot i recomputes its target velocity every sensing-control cycle.

Given the current state $\mathbf{x}_i$ of robot i and a target velocity $\mathbf{v}^{\star}_i$, the corresponding low-level control input for the current control cycle is given by the control policy of Eq. (6). Hence, the constraints on the control input of Eq. (3) transform into constraints on the target velocity $\mathbf{v}^{\star}_i$ of robot i. The resulting set of valid target velocities is denoted $\mathcal{V}^{\star}(\mathbf{x}_i) \subset \mathbb{R}^d$, and defined by:

$$\mathcal{V}^{\star}(\mathbf{x}_i) = \bigcap_k \{\mathbf{v}^{\star}_i \mid \mathbf{a}^T_k E\mathbf{v}^{\star}_i < b_k + \mathbf{a}^T_k L\mathbf{x}_i\}. \tag{13}$$

Hence, robot i can avoid collisions with *multiple* other robots while making sure that the control-input constraints are satisfied by selecting in each sensing-control cycle a valid target velocity outside the union of LQR-obstacles with respect to each other robot:

$$\mathbf{v}^{\star}_i \in \mathcal{V}^{\star}(\mathbf{x}_i) \setminus \bigcup_{j \neq i} (\mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_i - \mathbf{x}_j) \oplus \{V\mathbf{x}_j\}). \tag{14}$$

This approach works well if robot i is the only robot that observes the other robots and takes appropriate action to avoid them, i.e. robot i is *active* and the others are *passive*. If other robots j would be active as well, and simultaneously use the same approach to avoid collisions with robot i, *oscillations* in their motions occur [19]. This is because both robots determine their target velocity based on the (constantly violated) assumption that the other robot continues moving as it currently does. In other words, both robots take 100% of the responsibility to avoid collisions. To prevent these oscillations from occurring, the robots have to adopt an approach that takes into account that other robots react on them as well as they react on others, and, informally speaking, only take 50% of the responsibility of avoiding pairwise collisions. This is known as reciprocal collision avoidance [19], and will be discussed in the next section in the context of LQR-Obstacles.

IV. Reciprocal Collision Avoidance

In this section we extend the concept of LQR-Obstacles for reciprocal collision avoidance. We first discuss how a pair of robots reciprocally avoid collisions, and then extend the analysis to multiple robots and static obstacles.

A. A Pair of Robots

Let us consider a pair of robots i and j. We assume both robots can observe their own and each other's current states $\mathbf{x}_i$ and $\mathbf{x}_j$, and hence the current relative state $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. In order for robots i and j to reciprocally avoid collisions, target velocities $\mathbf{v}^{\star}_i$ and $\mathbf{v}^{\star}_j$ must be chosen for both robots simultaneously such that the relative target velocity $\mathbf{v}^{\star}_{ij} = \mathbf{v}^{\star}_i - \mathbf{v}^{\star}_j$ is outside the relative LQR-Obstacle, i.e. $\mathbf{v}^{\star}_{ij} \notin \mathcal{LQR}^{\tau}_{ij}(\mathbf{x}_{ij})$. The general approach is to assign i and j a set

of potential target velocities $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$, respectively, such that:

$$\big((\mathcal{RCA}^\tau_{ij} \cap \mathcal{V}^\star(\mathbf{x}_i)) \oplus -(\mathcal{RCA}^\tau_{ji} \cap \mathcal{V}^\star(\mathbf{x}_j))\big) \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) = \emptyset \quad (15)$$

Hence, if robot i selects a target velocity $\mathbf{v}^\star_i$ from $\mathcal{RCA}^\tau_{ij}$ such the control input constraints are satisfied, and robot j selects a target velocity $\mathbf{v}^\star_j$ from $\mathcal{RCA}^\tau_{ji}$ such the control input constraints are satisfied, it is guaranteed that $\mathbf{v}^\star_{ij} \notin \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$ and that the robots will not collide.

There are infinitely many pairs of sets of potential target velocities $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ for which these requirements hold. We want to select a pair of sets that is *fair*, i.e. both robots share the responsibility of avoiding collisions equally, *large*, i.e. both robots have a large amount of target velocities they can choose from, and defined such that both i and j can determine their own set of potential target velocities without mutual coordination. The sets $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ are constructed as follows.

Let $\mathcal{V}^\star_{ij}$ be the set of potential relative target velocities satisfying the control input constraints:

$$\mathcal{V}^\star_{ij} = \mathcal{V}^\star(\mathbf{x}_i) \oplus -\mathcal{V}^\star(\mathbf{x}_j) \quad (16)$$

This set is not disjoint with $\mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$ in general, so some relative target velocities will result in collision. Now, let us find a *convex* set $\mathcal{C}$ that is chosen such that:

$$(\mathcal{C} \cap \mathcal{V}^\star_{ij}) \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) = \emptyset \quad (17)$$

Hence, any relative target velocity in $\mathcal{C}$ that obeys the control input constraints will not result in a collision.

Since for convex sets $\mathcal{X}$ holds that $\frac{1}{2}\mathcal{X} \oplus \frac{1}{2}\mathcal{X} = \mathcal{X}$, we "divide" $\mathcal{C}$ by two to determine the sets of potential target velocities $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ for robot i and robot j, respectively:

$$\mathcal{RCA}^\tau_{ij} = \tfrac{1}{2}(\mathcal{C} \oplus \{-\mathbf{v}_{ij}\}) \oplus \{\mathbf{v}_i\}, \quad (18)$$

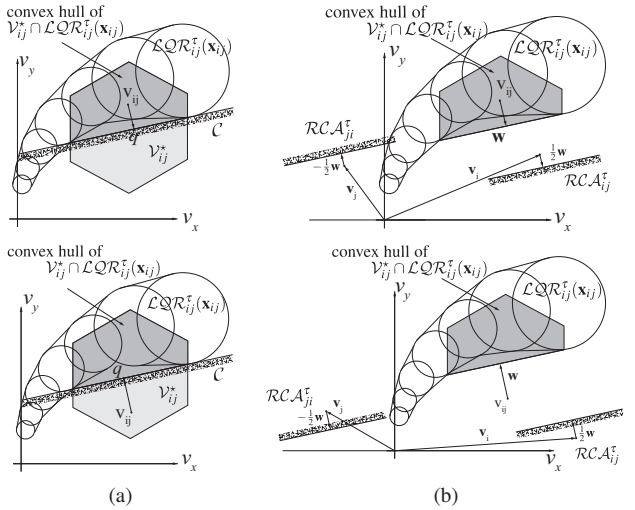
$$\mathcal{RCA}^\tau_{ji} = -\tfrac{1}{2}(\mathcal{C} \oplus \{-\mathbf{v}_{ij}\}) \oplus \{\mathbf{v}_j\}, \quad (19)$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are robot i's and robot j's current velocity, and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the current relative velocity of robot i and j. Here, we have "divided" the convex set of safe relative target velocities $\mathcal{C}$ about the robots' current relative velocity, and centered each of the parts about the robots' current velocities. This is to make sure that a maximal amount of potential target velocities are available around each of the robot's current velocity, from which (it is assumed) the robots want to deviate as little as possible (for the lack of knowledge of the robots about each other's true intent).

Let us prove that the definition of $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ of Eqs. (18) and (19) satisfy the requirement of Eq. (15):

$$\begin{aligned} & (\mathcal{RCA}^\tau_{ij} \cap \mathcal{V}^\star(\mathbf{x}_i)) \oplus -(\mathcal{RCA}^\tau_{ji} \cap \mathcal{V}^\star(\mathbf{x}_j)) \\ \subseteq\ & (\mathcal{RCA}^\tau_{ij} \oplus -\mathcal{RCA}^\tau_{ji}) \cap (\mathcal{V}^\star(\mathbf{x}_i) \oplus -\mathcal{V}^\star(\mathbf{x}_j)) \\ =\ & \big(\tfrac{1}{2}(\mathcal{C} \oplus \{-\mathbf{v}_{ij}\}) \oplus \{\mathbf{v}_i\} \oplus \tfrac{1}{2}(\mathcal{C} \oplus \{-\mathbf{v}_{ij}\}) \oplus \{-\mathbf{v}_j\}\big) \cap \mathcal{V}^\star_{ij} \\ =\ & \mathcal{C} \cap \mathcal{V}^\star_{ij}. \end{aligned} \quad (20)$$

Since $(\mathcal{C} \cap \mathcal{V}^\star_{ij}) \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) = \emptyset$ by definition (see Eq. (17)), this implies Eq. (15).



(a) (b)

Fig. 3. (a) Examples of the set $\mathcal{C}$ in case $\mathbf{v}_{ij}$ is inside (top) and outside (bottom) $\mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) \cap \mathcal{V}^\star_{ij}$. (b) The corresponding halfspaces $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$.

The question that remains is how to construct the convex set $\mathcal{C}$ such that $(\mathcal{C} \cap \mathcal{V}^\star_{ij}) \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) = \emptyset$. Ideally, we want $\mathcal{C}$ to be the largest such set, but this is difficult to achieve. Instead, we take the following approach. We define $\mathcal{C}$ to be the *halfspace* that is tangent to the convex hull of $\mathcal{V}^\star_{ij} \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$ at the closest point $\mathbf{q}$ on the boundary of the convex hull to the current relative velocity $\mathbf{v}_{ij}$ of the robots (see Fig. 3(a)). It then follows from the construction that $(\mathcal{C} \cap \mathcal{V}^\star_{ij}) \cap \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij}) = \emptyset$. We chose to take the tangent point closest to the current relative velocity $\mathbf{v}_{ij}$ to ensure that a maximal possible amount of potential target velocities are available close to the current velocities of the robots (assuming that the robots prefer to deviate as little as possible from their current velocity).

Since $\mathcal{C}$ is a halfspace, it follows from Eq. (18) that $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ are halfspaces too. Let $\mathbf{w} = \mathbf{q} - \mathbf{v}_{ij}$, i.e. the vector from the current relative velocity to the tangent point of the halfspace $\mathcal{C}$. If the robots are currently on a collision course, i.e. $\mathbf{v}_{ij} \in \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$, the vector $\mathbf{w}$ can be seen as the smallest change required to the relative velocity $\mathbf{v}_{ij}$ to escape an imminent collision. Knowing that the responsibility of avoiding the collision is to be shared by both robots, robot i must change its velocity by at least $\frac{1}{2}\mathbf{w}$, and robot j by at least $-\frac{1}{2}\mathbf{w}$. If the robots are not on a collision course, i.e. $\mathbf{v}_{ij} \notin \mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$, the vector $\mathbf{w}$ is the maximal change allowed to the relative velocity $\mathbf{v}_{ij}$ that still prevents a collision, and robot i may change its velocity by at most $\frac{1}{2}\mathbf{w}$, and robot j by at most $-\frac{1}{2}\mathbf{w}$. Indeed, the sets $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$ of safe target velocities for robot i and robot j, respectively, are halfspaces located at $\mathbf{v}_i + \frac{1}{2}\mathbf{w}$ and $\mathbf{v}_j - \frac{1}{2}\mathbf{w}$, respectively, that require the robots to choose a target velocity that make sure that collisions are avoided (see Fig. 3(b)).

We note that robot i and j can independently construct their sets of potential target velocities $\mathcal{RCA}^\tau_{ij}$ and $\mathcal{RCA}^\tau_{ji}$, respectively, since the construction from j's perspective, based on $\mathcal{LQR}^\tau_{ji}(\mathbf{x}_{ji})$, results in exactly the same sets $\mathcal{RCA}^\tau_{ij}$ and

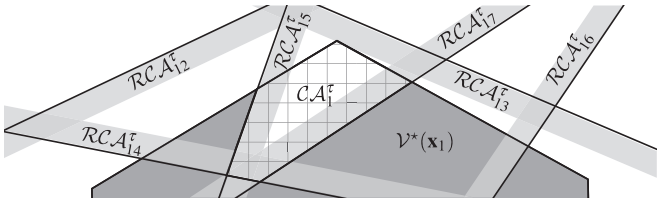


Fig. 4. Example of the set $\mathcal{CA}^\tau_1$ of safe target velocities for robot 1 among six other robots.

$\mathcal{RCA}^\tau_{ji}$ (note that $\mathcal{LQR}^\tau_{ji}(\mathbf{x}_{ji}) = -\mathcal{LQR}^\tau_{ij}(\mathbf{x}_{ij})$).

B. Multiple Robots

The set of safe target velocities $\mathcal{RCA}^\tau_{ij}$ for a robot i to reciprocally avoid collisions with a robot j as defined above can be used for independent navigation of multiple robots sharing a common workspace as follows. Each robot i independently performs a continual sensing-control cycle, in which each cycle consists of the following steps.

First, the robot estimates its own current state $\mathbf{x}_i$ and the current states $\mathbf{x}_j$ of the other robots j. Based on this information, robot i determines the set of safe target velocities $\mathcal{RCA}^\tau_{ij}$ with respect to each other robot j. The set of target velocities for i that are safe with respect to all other robots is the intersection of these sets. Accounting for static obstacles and the control constraints of i, we get (see Fig. 4):

$$\mathcal{CA}^\tau_i = \mathcal{V}^\star(\mathbf{x}_i) \cap \bigcap_{j \neq i} \mathcal{RCA}^\tau_{ij}. \tag{21}$$

Next the robot i determines its *preferred* velocity $\mathbf{v}^{\text{pref}}_i$ (i.e. the velocity it would have chosen if no other robots were around; we discuss below how such a velocity may be determined from a given target *position*), and chooses a target velocity $\mathbf{v}^\star_i$ from the set $\mathcal{CA}_i$ of safe target velocities that is closest to its preferred velocity:

$$\mathbf{v}^\star_i = \text{argmin}\{\mathbf{v} \in \mathcal{CA}^\tau_i\} \, \|\mathbf{v} - \mathbf{v}^{\text{pref}}_i\|. \tag{22}$$

Finally, the robot applies the low-level control input $\mathbf{u}_i$ as determined by Eq. (6), given $\mathbf{x}_i$ and $\mathbf{v}^\star_i$, and the process continues with the next sensing-control cycle.

It should be noted that in situations where the robots very densely occupy space it is possible that the set $\mathcal{CA}^\tau_i$ becomes empty. In this case, our implementation selects the "safest possible" target velocity $\mathbf{v}^\star_i$, i.e. one that minimally violates constraints [20]. However, this never occured in our simulations even with 100 quadrotors in a tight space.

C. Determining the Preferred Velocity from a Target Position

Above, the preferred velocity $\mathbf{v}^{\text{pref}}_i$ was defined as the velocity the the robot would have chosen as its target velocity $\mathbf{v}^\star_i$ if no other robots would be around. If the robot is given a specific target *position* $\mathbf{p}^\star_i$, we can infer the target velocity $\mathbf{v}^\star_i$ that leads the robot to its target position using an LQR-controller. From Eq. (7) we have the closed-loop dynamics of the robot, where $\mathbf{v}^\star_i$ (here endowed with a time subscript) can be seen as a control input: $\mathbf{x}_i[t+1] = \tilde{A}\mathbf{x}_i[t] + \tilde{B}\mathbf{v}^\star_i[t]$. The infinite-horizon LQR controller provides an optimal control policy to determine $\mathbf{v}^\star_i[t]$ that lets the robot reach the specified target position, given a quadratic cost function:

$$\textstyle\sum_{t=0}^{\infty}((C\mathbf{x}_i[t]-\mathbf{p}^\star_i)^T P(C\mathbf{x}_i[t]-\mathbf{p}^\star_i) + \mathbf{u}_i[t]^T R\mathbf{u}_i[t]), \tag{23}$$

where $\mathbf{u}_i[t]$ is the lower-level control input penalized by R as given before, and $P = P^T \geq 0$ specifies the weight of the cost of not being at the target position. Substituting the low-level control policy of Eq. (6) into Eq. (23), we get:

$$\begin{aligned}\textstyle\sum_{t=0}^{\infty}(&\mathbf{x}_i[t]^T\tilde{Q}\mathbf{x}_i[t] + \mathbf{v}^\star_i[t]^T\tilde{R}\mathbf{v}^\star_i[t] + \\ &2\mathbf{v}^\star_i[t]^T\tilde{P}\mathbf{x}_i[t] + 2\mathbf{x}_i[t]^T\tilde{\mathbf{q}}),\end{aligned} \tag{24}$$

where:

$$\tilde{Q} = C^T PC + L^T RL, \qquad \tilde{R} = E^T RE, \tag{25}$$

$$\tilde{P} = E^T RL, \qquad \tilde{\mathbf{q}} = -C^T P\mathbf{p}^\star_i. \tag{26}$$

Using this cost function and the closed-loop dynamics of Eq. (7) in a derivation similar as in Section III-A, we get a control policy for the target velocity $\mathbf{v}^*_i$ in terms of the target position $\mathbf{p}^\star_i$:

$$\mathbf{v}^\star_i[t] = \tilde{L}\mathbf{x}_i[t] + \tilde{E}\mathbf{p}^\star_i. \tag{27}$$

We use this policy to determine the preferred velocity $\mathbf{v}^{\text{pref}}_i = \tilde{L}\mathbf{x}_i + \tilde{E}\mathbf{p}^\star_i$ in each sensing-control cycle given the current state $\mathbf{x}_i$ and the target position $\mathbf{p}^\star_i$ of robot i.

V. Simulation Results

In this section we will discuss how the algorithm was implemented in a quadrotor simulation environment and the results from testing different scenarios with the developed reciprocal collision avoidance method.

A. Quadrotor Dynamics

The quadrotor helicopters used in our simulations were modeled after the Ascending Technologies' ResearchPilot. Its state $\mathbf{x} = (\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}^T, \mathbf{f}^T)^T$ is 16-dimensional, consisting of three-dimensional position $\mathbf{p}$, velocity $\mathbf{v}$, orientation $\mathbf{r}$ (rotation about axis $\mathbf{r}$ by angle $||\mathbf{r}||$), and angular velocity $\mathbf{w}$, and the current thrusts $\mathbf{f} = (f_1, f_2, f_3, f_4)^T$ of each of the rotors. The control input $\mathbf{f}^\star$ is 4-dimensional and consists of the desired thrusts for each of the rotors, constrained by a minimum and maximum for each rotor. Its dynamics are non-linear [14], and given by:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{28}$$

$$\dot{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \exp([\mathbf{r}]) \begin{bmatrix} 0 \\ 0 \\ f_1+f_2+f_3+f_4 \end{bmatrix} / m, \tag{29}$$

$$\dot{\mathbf{r}} = \mathbf{w} + \frac{1}{2}[\mathbf{r}]\mathbf{w} + (1 - \frac{\|\mathbf{r}\|}{2\tan(\frac{1}{2}\|\mathbf{r}\|)}) \frac{[\mathbf{r}]^2}{\|\mathbf{r}\|^2}\mathbf{w}, \tag{30}$$

$$\dot{\mathbf{w}} = J^{-1}(\begin{bmatrix} \ell(f_2-f_4) \\ \ell(f_3-f_1) \\ k_m(f_1-f_2+f_3-f_4) \end{bmatrix} - [\mathbf{w}]J\mathbf{w}), \tag{31}$$

$$\dot{\mathbf{f}} = k_f(\mathbf{f}^\star - \mathbf{f}), \tag{32}$$

where g is the gravity, m the mass of the robot, J the robot's moment of inertia matrix, ℓ the length of the beams, and k_m and k_f scaling constants. The notation $[\mathbf{a}]$ for a vector $\mathbf{a} \in \mathbb{R}^3$ refers to its skew-symmetric cross-product matrix.

The matrices A and B of the dynamics model of Eq. (4) are obtained by linearizing the above non-linear dynamics about the quadrotor's hover point. The projective matrices are given by $C = [I \;\; 0 \;\; \cdots \;\; 0]$, and $V = [0 \;\; I \;\; 0 \;\; \cdots \;\; 0]$. The geometry $\mathcal{O}_i$ of each quadrotor was defined as a bouding ellipse elongated along the z-axis (See Fig. 5) to prevent quadrotors from entering each other's immediate downwash [9].

Fig. 5. A bounding ellipse used as the geometry $\mathcal{O}_i$ of each quadrotor.

We note that the linearized model is only used to construct the LQR-Obstacles; our simulator uses the non-linear dynamics. Also, realistic amounts of artificial noise are injected in the motion and sensing of the robots in the simulator, while the LQR-Obstacles are constructed assuming perfect dynamics and sensing.

B. Implementation Details

Our algorithm was implemented in a simulator in C++. Since the geometry of a quadrotor is an ellipsoid, each LQR-Obstacle consists of a set of (transformed) ellipsoids. For each pair of quadrotors i and j, we modeled their LQR-Obstacle $\mathcal{LQR}^{\tau}_{ij}$ approximately using a set of points sampled from the boundaries of the constituting ellipsoids. A convex hull was computed (using the Qhull library [3]) of the points that are within $\mathcal{V}^{\star}_{ij}$ to compute the vector $\mathbf{w}$ (using an implementation of the GJK-method [7]). Based on this vector $\mathbf{w}$, the halfspace $\mathcal{RCA}^{\tau}_{ij}$ was computed. The set of safe target velocities $\mathcal{CA}^{\tau}_{i}$ for each quadrotor i is as a result the intersection of a set of 3-D halfspaces. The RVO2-3D library [20] was used to determine the velocity $\mathbf{v}^{\star}_i$ in $\mathcal{CA}^{\tau}_{i}$ that is closest to $\mathbf{v}^{\text{pref}}_i$.

C. Simulation Results

Simulations were run for a variety of configurations to demonstrate the performance of our algorithm (in all scenarios, a time-horizon of $\tau = 1.5$ seconds was used in a simulated sensing-control cycle of 30Hz). First, we show the situation of the two quadrotors flying directly towards each other for purpose of illustration. Two quadrotors begin some distance apart along the x-axis and pass each other in the xy-plane (Fig. 1).

Next, a scenario with 24 quadrotors was performed in which the quadrotors were all given initial and target positions that would force them to move through the center of the space, where a dense grouping of quadrotors is expected to occur (Fig. 6). In addition, a simulation with 100 quadrotors with random initial and target positions within a 10x10x10 meter space was run (Fig. 7). The results show that the algorithm directed the quadrotors successfully towards their individual goal positions, smoothly, while avoiding collisions with one another. In none of the simulations we performed collisions were observed (see also the attached video or videos at `http://arl.cs.utah.edu/research/rca/`).

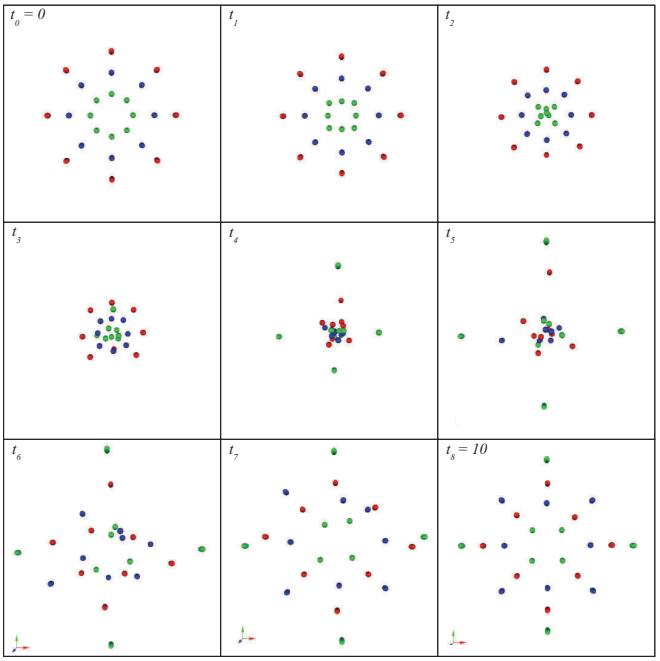


Fig. 6. The simulation with 24 quadrotors was run in such a way that a dense packing of quadrotors occurs in the center. Nine different steps through the simulation are shown.

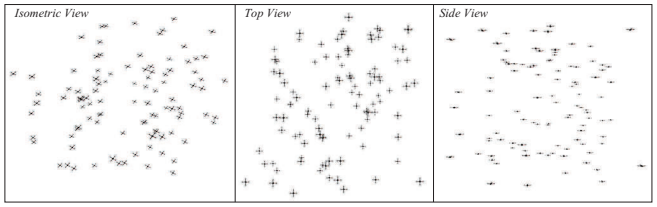


Fig. 7. Simulation of 100 quadrotors avoiding collisions with each other in a 10x10x10 meter space. See also the attached video or videos at `http://arl.cs.utah.edu/research/rca/`.

Performance: Our simulations were run on a desktop machine running Windows 7 Professional 64-bit with an *Intel i7-2600 CPU and 8GB-RAM*. To quantify the performance of our approach, we report the average computation time of each quadrotor to select a safe target velocity for itself in each sensing-control cycle (recall that in our approach the calculations are fully independent for each quadrotor). The results can be seen in Fig. 8 for experiments involving up to 128 quadrotors with random initial and target positions in a 10x10x10 meter space. As expected, the average computation time for each quadrotor is approximately linear in the number of other quadrotors it has to avoid collisions with. As seen in Fig. 8, the limit on the number of quadrotors that can be avoided with the computations being performed in real-time (33ms at a sensing-control cycle of 30Hz) is approximately 75. We note however, that for safe navigation it is not necessary to consider that many quadrotors in the collision avoidance. It would suffice to consider only a fixed number of nearest neighbors, or only quadrotors within a certain range. We also expect that the performance can be further improved by optimizing the implementation.

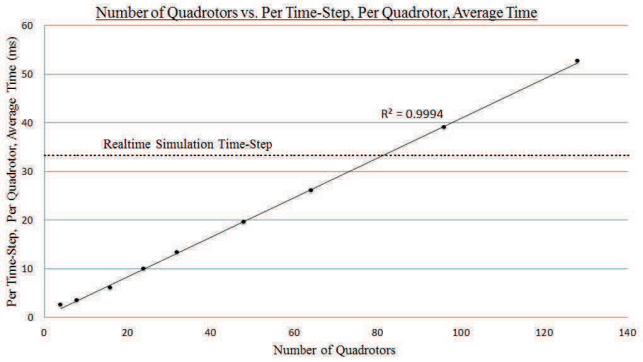


Fig. 8. The average computation time per quadrotor in each sensing-control cycle for an increasing number of quadrotors in the workspace. The horizontal, dashed line represents the amount of time per simulated sensing-control cycle, and is the upper-bound on the required computation time for real-time performance. It can be seen that when each quadrotor is considering all other quadrotors for collision avoidance, up to approximately 80 quadrotors can be avoided in real-time.

VI. Conclusions and Future Work

In this paper we have presented a method for reciprocal collision avoidance for multiple robots subject to linear differential constraints. In our approach each robot acts fully independently (there is no coordination among robots) and only need to continually observe each other's current state. Our quadrotor simulation results displayed that our approach is able to let a group of robots reach their target position from an initial position while smoothly avoiding collision with the other robots. Even though our approach is based on assumptions of perfect linear dynamics and perfect sensing, it still provides reliable collision avoidance in situations where these assumptions do not hold (our simulator used non-linear dynamics with aritificial motion and sensing noise). This is because our approach computes a new control for the robots in each sensing-acting cycle, and as such adapts in real-time to unpredicted situations. As an alternative, one can explicitly take into account motion and sensing noise by basing the approach on *LQG*-Obstacles [22] rather than LQR-Obstacles as was done in this paper. Motivated by the promising simulation results, we are currently implementing our approach on real-world quadrotors.

Our approach has a number of limitations. First, our approach requires that the state of the robot contains its position and its velocity, and that the geometry of the robot is fixed and only translates (i.e. it does not change with rotation). This limits the applicability of our approach to mobile robots, whose geometry can be approximated using a bounding disk or sphere. Second, while our approach can be extended for avoiding collisions with any static and moving obstacles, the other robots with which collisions are reciprocally avoided must in our current formulation have exactly the same dynamics, in order to be able to formulate the dynamics model of the robots' *relative* motion. Robots of different dynamics could be handled by using an abstraction of their dynamics model. For instance, it is known that quadrotors are dynamically capable of following any trajectory that is created by controlling snap (the second derivative of acceleration) within certain bounds. Since our approach does allow for each robot having its individual control constraints, collisions could be avoided among, for instance, quadrotors of different sizes and thrust capabilities by using such an abstract dynamics model.

References

[1] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. *Proc. Int. Symp. on Distributed Autonomous Robotic Systems*, 2010.

[2] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, R. Siegwart. Reciprocal collision avoidance for multiple car-like robots. *IEEE Int. Conf. on Robotics and Automation*, 2012.

[3] C. Barber, D. Dobkin, T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software* 22(4):469–483, 1996.

[4] G. Ducard, R. d'Andrea. Autonomous quadrotor flight using a vision system and accomodating frames misalignment. *IEEE Int. Symposium on Industrial Embedded Systems*, 2009.

[5] P. Fiorini, Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. J. of Robotics Research* 17(7):760–772, 1998.

[6] D. Fox, W. Burgard, S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine* 4:22–23, 1997.

[7] E. Gilbert, D. Johnson, S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation* 4(2):193–203, 1988.

[8] D. Hsu, R. Kindel, J. Latombe, S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robotics Research* 21(3):233–255, 2002.

[9] H. Huang, G. Hoffmann, S. Waslander, C. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. *IEEE Int. Conf. on Robotics and Automation*, 2009.

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

[11] S. Lupashin, A. Schöllig, M. Sherback, R. d'Andrea. A simple learning strategy for high-speed quadrotor multi-flips. *IEEE Int. Conf. on Robotics and Automation*, 2010.

[12] D. Mellinger, M. Shomin, V. Kumar. Control of quadrotors for robust perching and landing. *Int. Powered Lift Conference*, 2010.

[13] D. Mellinger, N. Michael, M. Shomin. V. Kumar. Recent advances in quadrotor capabilities. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[14] N. Michael, D. Mellinger, Q. Lindsey, V. Kumar. The grasp multiple micro-uav test bed: Experimental evaluation of multirobot aerial control algorithms. *IEEE Robotics & Automation Magazine* 17(3):56–65, 2010.

[15] S. Petti, T. Fraichard. Safe Motion Planning in Dynamic Environments. *Proc. IEEE RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

[16] J. Snape, J. van den Berg, S. Guy, D. Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010.

[17] M. Turpin, N. Michael, V. Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Proc. Int. Symp. of Robotics Research*, 2011.

[18] M. Turpin, N. Michael, V. Kumar. Trajectory planning and assignment in multirobot systems. *Proc. Workshop on Algorithmic Foundations of Robotics*, 2012.

[19] J. van den Berg, M. Ling, D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE Int. Conf. on Robotics and Automation*, 2008.

[20] J. van den Berg, S. Guy, M. Lin, D. Manocha. Reciprocal *n*-body collision avoidance. *Proc. Int. Symp. of Robotics Research*, 2009.

[21] J. van den Berg, J. Snape, S. Guy, D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[22] J. van den Berg, D. Wilkie, S. Guy, M. Niethammer, D. Manocha. LQG-Obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty. *IEEE Int. Conf. on Robotics and Automation*, 2012.