Reciprocal *n*-body Collision Avoidance

Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha

Abstract In this paper, we present a formal approach to *reciprocal n-body collision avoidance*, where multiple mobile robots need to avoid collisions with each other while moving in a common workspace. In our formulation, each robot acts fully independently, and does not communicate with other robots. Based on the definition of velocity obstacles [5], we derive sufficient conditions for collision-free motion by reducing the problem to solving a low-dimensional linear program. We test our approach on several dense and complex simulation scenarios involving thousands of robots and compute collision-free actions for all of them in only a few milliseconds. To the best of our knowledge, this method is the first that can guarantee local collision-free motion for a large number of robots in a cluttered workspace.

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 cycle of sensing and acting. In each cycle, an action for the robot must be computed based on local observations of the environment, such that the robot stays free of collisions with the obstacles and the other moving entities, while making progress towards a goal.[1]

The authors are with the Department of Computer Science, University of North Carolina at Chapel Hill, USA. E-mail: {berg, sjguy, lin, dm}@cs.unc.edu.
This research is supported in part by ARO Contracts W911NF-04-1-0088, NSF award 0636208, DARPA/RDECOM Contracts N61339-04-C-0043 and WR91CRB-08-C-0137, Intel, and Microsoft.

[1] Note that the problem of (local) collision-avoidance differs from *motion planning*, where the global environment of the robot is considered to be known and a complete path towards a goal configuration is planned at once [18], and *collision detection*, which simply determines if two geometric objects intersect or not (see e.g. [17]).

The problem of collision avoidance has been well studied for one robot avoiding static or moving obstacles. In this paper, we address the more involved and less studied problem of *reciprocal n-body collision avoidance*, where collisions need to be avoided among multiple robots (or any decision-making entities). This problem has important applications in many areas in robotics, such as multi-robot navigation and coordination among swarms of robots [20]. It is also an key component in crowd simulation for computer graphics and VR [11, 21], modeling of non-player characters in AI, studying flocks of birds and fish in biology [23], and real-time (air) traffic control [16]. In this paper, we propose a fast and novel method that simultaneously determines actions for many (possibly thousands of) robots that each may have different objectives. The actions are computed for each robot independently, without communication among the robots or central coordination. Yet, we prove that our method guarantees collision-free motion for each of the robots.

We use a simplified robot model, where each robot is assumed to have a simple shape (circular or convex polygon) moving in a two-dimensional workspace. Furthermore, we assume that the robot is *holonomic*, i.e. it can move in any direction, such that the control input of each robot is simply given by a two-dimensional velocity vector. Also, we assume that each robot has *perfect* sensing, and is able to infer the exact shape, position and velocity of obstacles and other robots in the environment.

Main results: We present a rigorous approach for reciprocal n-body collision avoidance that provides a *sufficient* condition for each robot to be collision-free for at least a fixed amount of time into the future, only assuming that the other robots use the same collision-avoidance protocol. Our approach is *velocity-based*. That implies that each robot takes into account the observed velocity of other robots in order to avoid collisions with them, and also that the robot selects its own velocity from its *velocity space* in which certain regions are marked as 'forbidden' because of the presence of other robots. Our formulation, "optimal reciprocal collision avoidance", infers for each other robot a half-plane (in velocity-space) of velocities that are allowed to be selected in order to guarantee collision avoidance. The robot then selects its optimal velocity from the intersection of all permitted half-planes, which can be done efficiently using *linear programming*. Under certain conditions with densely packed robots, the resulting linear program may be infeasible, in which case we select the 'safest possible' velocity using a three-dimensional linear program.

We experimented with our approach on several complex simulation scenarios containing thousands of robots. As each robot is independent, we can fully *parallellize* the computation of the actions for each robot and report very fast real-time running times. Furthermore, our experiments show that our approach achieves convincing motions that are smooth and collision-free.

The rest of this paper is organized as follows. We start by discussing previous work in Section 2. In Section 3, we formally define the problem we address in this paper. We derive the half-plane of permitted velocities for optimal reciprocal collision avoidance of a robot with respect to another robot in Section 4, and show how this approach is used to navigate among multiple robots in Section 5. We report experimental results in Section 6 and conclude in Section 7.

2 Previous Work

The problem of collision avoidance has been extensively studied. Many approaches assume the observed obstacles to be static (i.e. non-moving) [2, 4, 6, 7, 13, 14, 24], and compute an immediate action for the robot that would avert collisions with the obstacle, in many cases taking into account the robot's kinematics and dynamics. If the obstacles are also moving, such approaches typically repeatedly "replan" based on new readings of the positions of the obstacles. This may work well if the obstacles move slower than the robot, but among fast obstacles (such as crossing a highway), the velocity of the obstacles need to be specifically taken into account. This problem is generally referred to as "asteroid avoidance", and approaches typically extrapolate the observed velocities in order to estimate the future positions of obstacles [8, 9, 12, 19, 22, 28].

The problem of collision avoidance becomes harder when the obstacles are not simply moving without considering their environment, but are also intelligent decision-making entities that try to avoid collisions as well. Simply considering them as moving obstacles may lead to *oscillations* if the other entity considers all other robots as moving obstacles as well [15, 26]. Therefore, the reactive nature of the other entities must be specifically taken into account in order to guarantee that collisions are avoided. Yet, the robot may not be able to communicate with other entities and may not know their intents. We call this problem *reciprocal collision avoidance*, and is the focus of this paper.

Velocity obstacles (VO) [5] have been a successful velocity-based approach to avoid collisions with moving obstacles; they provide a *sufficient* and *necessary* condition for a robot to select velocity that avoids collisions with an obstacle moving at a known velocity. This approach was extended for robot-robot collision avoidance with the definition of Reciprocal Velocity Obstacles (RVO) [10, 26], where both robots were assumed to select a velocity outside the RVO induced by the other robot. However, this formulation only guarantees collision-avoidance under specific conditions, and does not provide a sufficient condition for collision-avoidance in general.[2] In this paper, we present the principle of *optimal reciprocal collision avoidance* (ORCA) that overcomes this limitation; ORCA provides a sufficient condition for multiple robots to avoid collisions among one another, and thus can guarantee collision-free navigation.

We note that it is possible to provide a sufficient and necessary condition for multiple (say n) robots to select collision-avoiding velocities, by defining a *composite* velocity obstacle in the $2n$-dimensional space of velocities of all n robots [1]. However, this is not only computationally impractical, it also requires central coordination among robots. This is incompatible with the type of distributed multi-robot navigation we focus on in this paper, in which each robot independently and simultaneously selects its velocity from its own 2-D velocity space.

[2] In fact, both robots selecting a velocity *inside* each other's RVO is a sufficient condition to end up in a collision.

Problem assumptions: robot is a disk in $\mathbb{R}^2$; each robot has p, r, v that can be observed by other robots and V_{max} and V_{pref} which cannot be observed; all robots use same strategy to select new velocities

3 Problem Definition

Task: given a duration of time τ, select a velocity v_{new} such that the robot is guranteed collision-free for τ for all robots

Secondary task: pick v_{new} to be as close to the prefered velocity as possible

The problem we discuss in this paper is formally defined as follows. Let there be a set of n robots sharing an environment. For simplicity we assume the robots are disc-shaped and move in the plane $\mathbb{R}^2$ (the definitions and algorithms we present in this paper can easily be extended to translating polygons, and also to higher dimensions). Each robot A has a current position $\mathbf{p}_A$ (the center of its disc), a current velocity $\mathbf{v}_A$ and a radius r_A. These parameters are part of the robot's external state, i.e. we assume that they can be observed by other robots. Furthermore, each robot has a maximum speed $v_A^{\max}$ and a preferred velocity $\mathbf{v}_A^{\text{pref}}$, which is the velocity the robot would assume had no other robots been in its way (for instance a velocity directed towards the robot's goal with a magnitude equal to the robot's preferred speed). We consider these parameters part of the internal state of the robot, and can therefore not be observed by other robots.

The task is for each robot A to *independently* (and simultaneously) select a new velocity $\mathbf{v}_A^{\text{new}}$ for itself such that *all* robots are *guaranteed* to be collision-free for at least a preset amount of time τ when they would continue to move at their new velocity. As a secondary objective, the robots should select their new velocity as close as possible to their preferred velocity. The robots are not allowed to communicate with each other, and can only use observations of the other robot's *current* position and velocity. However, each of the robots may assume that the other robots use the same strategy as itself to select a new velocity.

We name this problem "reciprocal n-body collision avoidance". Note that this problem cannot be solved using central coordination, as the preferred velocity of each robot is only known to the robot itself. In Section 4, we present a *sufficient* condition for each robot to select a velocity that is collision-free for (at least) τ time. In Section 5 we show how we use this principle in a continuous cycle for multi-robot navigation.

$D(p,r) = \{q \mid \|q-p\| < r\}$ => All points q where the distance between q and p are less than the radius of the disk

4 Reciprocal Collision Avoidance

4.1 Preliminaries

For two robots A and B, the *velocity obstacle* $VO^{\tau}_{A|B}$ (read: the velocity obstacle for A induced by B for time window τ) is the set of all *relative* velocities of A with respect to B that will result in a collision between A and B at some moment before time τ [5]. It is formally defined as follows. Let $D(\mathbf{p}, r)$ denote an open disc of radius r centered at $\mathbf{p}$;

$$D(\mathbf{p}, r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\| < r\}, \tag{1}$$

then:

$$VO^{\tau}_{A|B} = \{\mathbf{v} \mid \exists t \in [0, \tau] :: t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\} \tag{2}$$

$VO^{\tau}_{A|B} = \{v \mid \exists t \in [0,\tau] :: tv \in D(p_B - p_A, r_A + r_B)\}$ => All v for t in the range $[0, \tau]$ where $t \cdot v$ falls into the bounds of the disk $D(p_B - p_A, r_A + r_B)$

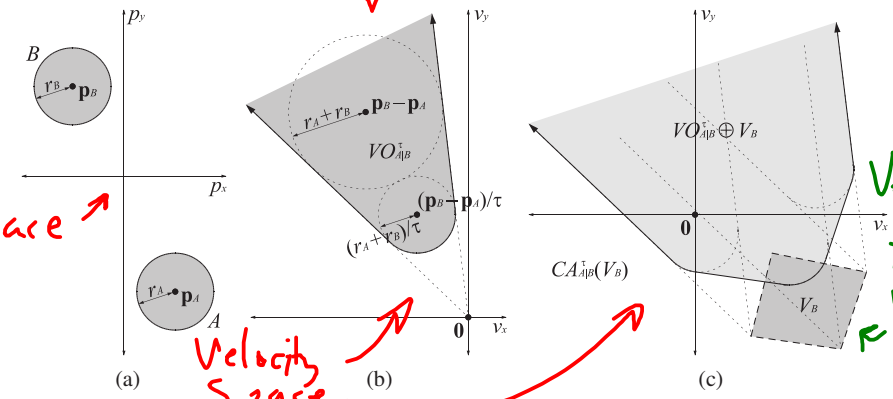


Fig. 1 **(a)** A configuration of two robots A and B. **(b)** The velocity obstacle $VO^{\tau}_{A|B}$ (gray) can geometrically be interpreted as a truncated cone with its apex at the origin (in velocity space) and its legs tangent to the disc of radius $r_A + r_B$ centered at $\mathbf{p}_B - \mathbf{p}_A$. The amount of truncation depends on the value of τ; the cone is truncated by an arc of a disc of radius $(r_A + r_B)/\tau$ centered at $(\mathbf{p}_B - \mathbf{p}_A)/\tau$. The velocity obstacle shown here is for $\tau = 2$. **(c)** The set of collision-avoiding velocities $CA^{\tau}_{A|B}(V_B)$ for robot A given that robot B selects its velocity from some set V_B (dark gray) is the complement of the Minkowski sum (light gray) of $VO^{\tau}_{A|B}$ and V_B.

The geometric interpretation of velocity obstacles is shown in Fig. 1(b). Note that $VO^{\tau}_{A|B}$ and $VO^{\tau}_{B|A}$ are *symmetric* in the origin.

Let $\mathbf{v}_A$ and $\mathbf{v}_B$ be current the velocities of robots A and B, respectively. The definition of the velocity obstacle implies that if $\mathbf{v}_A - \mathbf{v}_B \in VO^{\tau}_{A|B}$, or equivalently if $\mathbf{v}_B - \mathbf{v}_A \in VO^{\tau}_{B|A}$, then A and B will collide at some moment before time τ if they continue moving at their current velocity. Conversely, if $\mathbf{v}_A - \mathbf{v}_B \notin VO^{\tau}_{A|B}$, robot A and B are guaranteed to be collision-free for at least τ time.

More generally, let $X \oplus Y$ denote the Minkowski sum of sets X and Y;

$$X \oplus Y = \{\mathbf{x} + \mathbf{y} \,|\, \mathbf{x} \in X, \mathbf{y} \in Y\}, \tag{3}$$

then for any set V_B, if $\mathbf{v}_B \in V_B$ and $\mathbf{v}_A \notin VO^{\tau}_{A|B} \oplus V_B$, then A and B are guaranteed to be collision-free at their current velocities for at least τ time. This leads to the definition of the set of *collision-avoiding* velocities $CA^{\tau}_{A|B}(V_B)$ for A given that B selects its velocity from V_B (see Fig. 1(c)):

$$CA^{\tau}_{A|B}(V_B) = \{\mathbf{v} \,|\, \mathbf{v} \notin VO^{\tau}_{A|B} \oplus V_B\} \tag{4}$$

We call a pair of sets V_A and V_B of velocities for A and B *reciprocally collision-avoiding* if $V_A \subseteq CA^{\tau}_{A|B}(V_B)$ and $V_B \subseteq CA^{\tau}_{B|A}(V_A)$. If $V_A = CA^{\tau}_{A|B}(V_B)$ and $V_B = CA^{\tau}_{B|A}(V_A)$, we call V_A and V_B *reciprocally maximal*.

4.2 Optimal Reciprocal Collision Avoidance

Given the definitions above, we would like to choose sets of *permitted* velocities V_A for A and V_B for B such that $CA^\tau_{A|B}(V_B) = V_A$ and $CA^\tau_{B|A}(V_A) = V_B$, i.e. they are reciprocally collision-avoiding and maximal and guarantee that A and B are collision-free for at least τ time. Also, because A and B are individual robots, they should be able to infer their set of permitted velocities without communication with the other robot. There are infinitely many pairs of sets V_A and V_B that obey these requirements, but among those we select the pair that maximizes the amount of permitted velocities "close" to *optimization velocities* $\mathbf{v}^{\text{opt}}_A$ for A and $\mathbf{v}^{\text{opt}}_B$ for B.[3] We denote these sets $ORCA^\tau_{A|B}$ for A and $ORCA^\tau_{B|A}$ for B, and formally define them as follows. Let $|V|$ denote the measure (i.e. area in $\mathbb{R}^2$) of set V;

Definition 1 (Optimal Reciprocal Collision Avoidance). $ORCA^\tau_{A|B}$ and $ORCA^\tau_{B|A}$ are defined such that they are reciprocally collision-avoiding and maximal, i.e. $CA^\tau_{A|B}(ORCA^\tau_{B|A}) = ORCA^\tau_{A|B}$ and $CA^\tau_{B|A}(ORCA^\tau_{A|B}) = ORCA^\tau_{B|A}$, and such that for *all* other pairs of sets of reciprocally collision-avoiding velocities V_A and V_B (i.e. $V_A \subseteq CA^\tau_{A|B}(V_B)$ and $V_B \subseteq CA^\tau_{B|A}(V_A)$), and for *all* radii $r > 0$,

$$|ORCA^\tau_{A|B} \cap D(\mathbf{v}^{\text{opt}}_A, r)| = |ORCA^\tau_{B|A} \cap D(\mathbf{v}^{\text{opt}}_B, r)| \geq \min(|V_A \cap D(\mathbf{v}^{\text{opt}}_A, r)|, |V_B \cap D(\mathbf{v}^{\text{opt}}_B, r)|).$$

This means that $ORCA^\tau_{A|B}$ and $ORCA^\tau_{B|A}$ contain more velocities close to $\mathbf{v}^{\text{opt}}_A$ and $\mathbf{v}^{\text{opt}}_B$, respectively, than any other pair of sets of reciprocally collision-avoiding velocities. Also, the distribution of permitted velocities is "fair", as the amount of velocities close to the optimization velocity is equal for A and B.

We can geometrically construct $ORCA^\tau_{A|B}$ and $ORCA^\tau_{B|A}$ as follows. Let us assume that A and B adopt velocities $\mathbf{v}^{\text{opt}}_A$ and $\mathbf{v}^{\text{opt}}_B$, respectively, and let us assume that that causes A and B to be on collision course, i.e. $\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B \in VO^\tau_{A|B}$. Let $\mathbf{u}$ be the vector from $\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B$ to the closest point on the boundary of the velocity obstacle (see Fig. 2):

$$\mathbf{u} = \left(\operatorname*{arg\,min}_{\mathbf{v} \in \partial VO^\tau_{A|B}} \|\mathbf{v} - (\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B)\|\right) - (\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B), \quad (5)$$

and let $\mathbf{n}$ be the outward normal of the boundary of $VO^\tau_{A|B}$ at point $(\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B) + \mathbf{u}$. Then, $\mathbf{u}$ is the smallest change required to the relative velocity of A and B to avert collision within τ time. To "share the responsibility" of avoiding collisions among the robots in a fair way, robot A adapts its velocity by (at least) $\frac{1}{2}\mathbf{u}$ and assumes that

[3] We introduce these optimization velocities to generalize the definition of ORCA. Nominally, the optimization velocities are equal to the current velocities, such that the robots have to deviate as little as possible from their current trajectories to avoid collisions. Other choices are discussed in detail in Section 5.2.

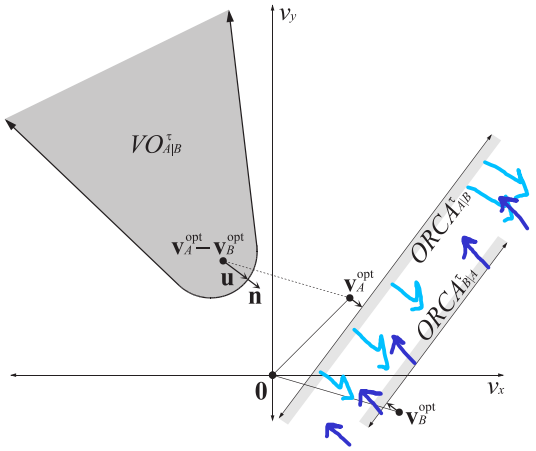


Fig. 2 The set $ORCA^{\tau}_{A|B}$ of permitted velocities for A for optimal reciprocal collision avoidance with B is a half-plane delimited by the line perpendicular to $\mathbf{u}$ through the point $\mathbf{v}_A^{\text{opt}} + \frac{1}{2}\mathbf{u}$, where $\mathbf{u}$ is the vector from $\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}}$ to the closest point on the boundary of $VO^{\tau}_{A|B}$.

B takes care of the other half. Hence, the set $ORCA^{\tau}_{A|B}$ of permitted velocities for A is the *half-plane* pointing in the direction of $\mathbf{n}$ starting at the point $\mathbf{v}_A^{\text{opt}} + \frac{1}{2}\mathbf{u}$. More formally:

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} \,|\, (\mathbf{v} - (\mathbf{v}_A^{\text{opt}} + \frac{1}{2}\mathbf{u})) \cdot \mathbf{n} \geq 0\}. \tag{6}$$

The set $ORCA^{\tau}_{B|A}$ for B is defined symmetrically (see Fig. 2). The above equations also apply if A and B are *not* on a collision course when adopting their optimization velocities, i.e. $\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}} \notin VO^{\tau}_{A|B}$. In this case, both robots each will take half of the responsibility to remain on a collision-free trajectory.

It can be seen that $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$ as constructed above are in fact optimal according to the criterion of Definition 1. Agents A and B can infer $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$, respectively, without communicating with each other, as long the robots can *observe* each other's position, radius, and optimization velocity. In Section 5.2, we discuss reasonable choices for the optimization velocity of the robots.

5 n-Body Collision Avoidance

In this section we show how to apply the ORCA principle as defined above to perform *n-body collision avoidance* among multiple moving robots, and discuss how we can incorporate static obstacles in this framework.

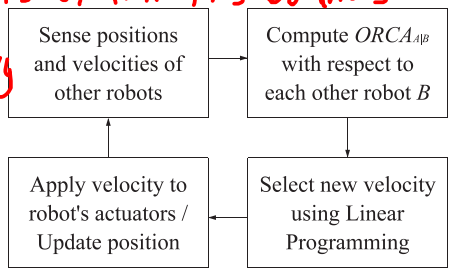


Fig. 3 A schematic overview of the continuous cycle of sensing and acting that is independently executed by each robot.

5.1 Basic Approach

The overall approach is as follows. Each robot A performs a continuous cycle of sensing and acting with time step Δt. In each iteration, the robot acquires the radius, the current position and the current optimization velocity of the other robots (and of itself). Based on this information, the robot infers the permitted half-plane of velocities $ORCA^{\tau}_{A|B}$ with respect to each other robot B. The set of velocities that are permitted for A with respect to *all* robots is the intersection of the half-planes of permitted velocities induced by each other robot, and we denote this set $ORCA^{\tau}_{A}$ (see Fig. 4):

$$ORCA^{\tau}_{A} = D(\mathbf{0}, v^{\max}_{A}) \cap \bigcap_{B \neq A} ORCA^{\tau}_{A|B}. \quad (7)$$

Note that this definition also includes the maximum speed constraint on robot A.

Next, the robot selects a new velocity $\mathbf{v}^{\text{new}}_{A}$ for itself that is closest to its preferred velocity $\mathbf{v}^{\text{pref}}_{A}$ amongst all velocities inside the region of permitted velocities:

$$\mathbf{v}^{\text{new}}_{A} = \underset{\mathbf{v} \in ORCA^{\tau}_{A}}{\arg\min} \|\mathbf{v} - \mathbf{v}^{\text{pref}}_{A}\|. \quad (8)$$

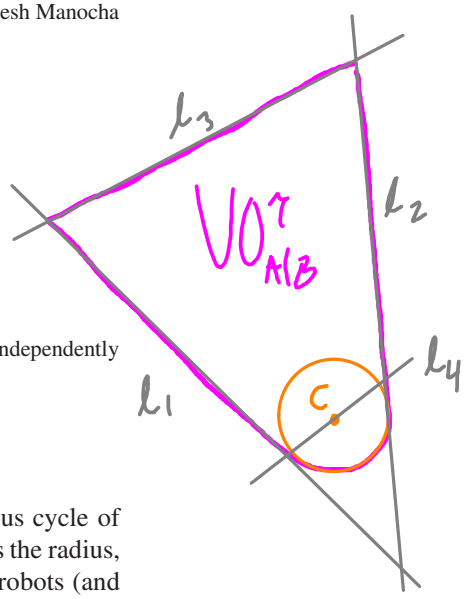
We will show below how to compute this velocity efficiently. Finally, the robot reaches its new position;

$$\mathbf{p}^{\text{new}}_{A} = \mathbf{p}_{A} + \mathbf{v}^{\text{new}}_{A} \Delta t, \quad (9)$$

and the sensing-acting cycle repeats (see Fig. 3).

The key step in the above procedure is to compute the new velocity $\mathbf{v}^{\text{new}}_{A}$ as defined by Equations (7) and (8). This can efficiently be done using *linear programming*, as $ORCA^{\tau}_{A}$ is a *convex* region bounded by linear constraints induced by the half-planes of permitted velocities with respect to each of the other robots (see Fig. 4). The optimization function is the distance to the preferred velocity $\mathbf{v}^{\text{pref}}_{A}$. Even though this is a quadratic optimization function, it does not invalidate the linear programming characteristics, as it has only one local minimum.

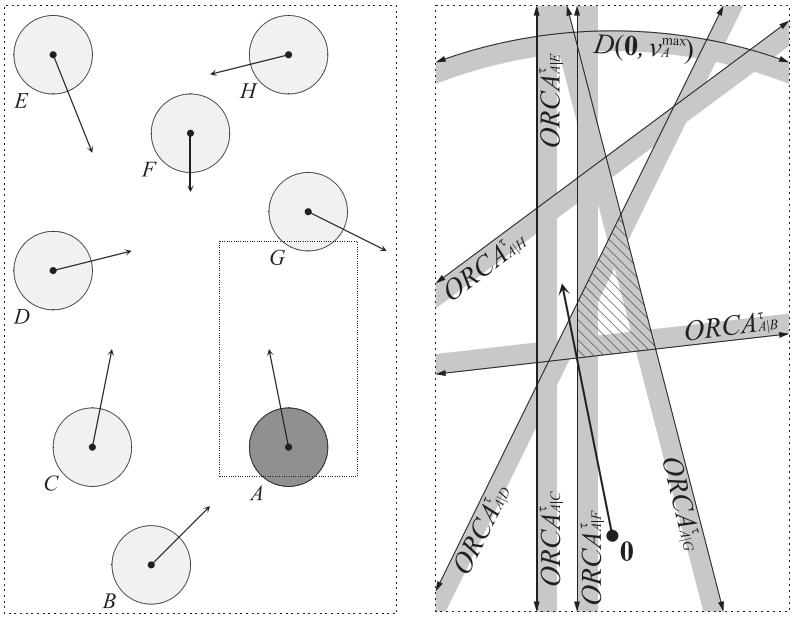
We use the efficient algorithm of [3], which adds the constraints one by one in random order while keeping track of the current optimal new velocity. The algorithm



Use ROS to simulate "sensing" other robots positions and velocities
↳ Some coordinator node will have to be responsible for collecting and redistributing this data
Design a representation of the $VO^{\tau}_{A|B}$ polygon while keeping in mind how it'll be used
↳ Check if a vector $V=[V_x, V_y]$ is within its bounds
↳ Find a point on its boundary closest to a given

See Figure 4B

Could represent $VO^{\tau}_{A|B}$ as four half planes and a circle semi-algebraic model. Easily solves Polygonal contains use case, but not boundary seeking case...



(a) (b)

Fig. 4 **(a)** A configuration with eight robots. Their current velocities are shown using arrows. **(b)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and with $\mathbf{v}_*^{\text{opt}} = \mathbf{v}_*$ for all robots (i.e. the optimization velocity equals the current velocity). The half-planes of E and C coincide. The dashed region is $ORCA_A^\tau$, and contains the velocities for A that are permitted with respect to all other robots. The arrow indicates the current velocity of A.

has an expected running time of $O(n)$, where n is the total number of constraints in the linear program (which equals the number of robots in our case). The fact that we include a circular constraint for the maximum speed does not significantly alter the algorithm, and does not affect the running time. The algorithm returns the velocity in $ORCA_A^\tau$ that is closest to $\mathbf{v}_A^{\text{pref}}$, and reports failure if the linear program is infeasible, i.e. when $ORCA_A^\tau = \emptyset$. If the optimization velocities for the robots are chosen carefully (as we will discuss in Section 5.2), $ORCA_A^\tau$ will never be empty, and hence there will always be a solution that guarantees that the robots are collision-free for at least τ time.

We can increase the efficiency of selecting velocities by not including the constraints of all other robots, but only of those that are "close" by. In fact, robots B that are farther away from robot A than $(v_A^{\max} + v_B^{\max})\tau$ will never collide with robot A within τ time, so they can safely be left out of the linear program when computing the new velocity for robot A. A minor issue is that robot A does not know the maximum speed of other robots, but this can be worked around by "guessing" the maximum speed of other robots to be equal A's own. We can efficiently find the set of close-by robots whose constraints should be included using a *kD-tree*.

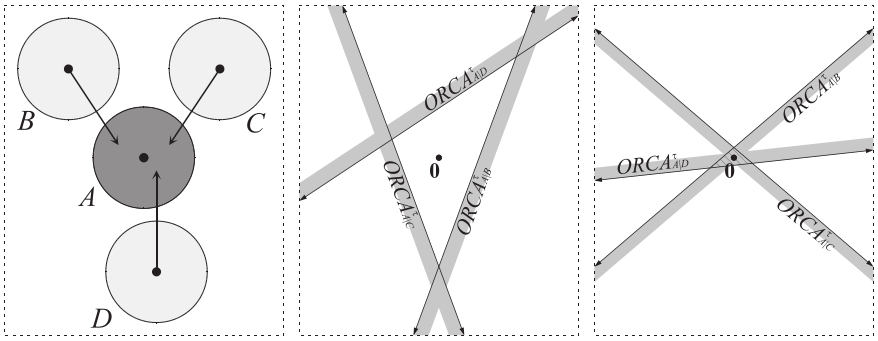


Fig. 5 **(a)** A dense configuration with three robots moving towards robot A. The current velocities of the robots are shown using arrows; robot A has zero velocity. **(b)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and $\mathbf{v}^{\text{opt}}_* = \mathbf{v}_*$ for all robots. The region $ORCA^{\tau}_A$ is empty, so avoiding collisions within τ time cannot be guaranteed. **(c)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and $\mathbf{v}^{\text{opt}}_* = \mathbf{0}$ for all robots. The dashed region is $ORCA^{\tau}_A$.

5.2 Choosing the Optimization Velocity

One issue that we have left open is how to choose $\mathbf{v}^{\text{opt}}_A$ for each robot A. In order for the robots to infer the half-planes without communication, $\mathbf{v}^{\text{opt}}_A$ must be observable to other robots. Here, we discuss some reasonable possibilities:

- $\mathbf{v}^{\text{opt}}_A = \mathbf{0}$ for all robots A. If we set the optimization velocity to zero for all robots, it is *guaranteed* that $ORCA^{\tau}_A$ is non-empty for all robots A (see Fig. 5(c)). Hence, the linear programming algorithm as described above will find a velocity for all robots that guarantees them to be collision-free for at least τ time. This can be seen as follows. For any other robot B, the point $\mathbf{0}$ always lies outside the velocity obstacle $VO^{\tau}_{A|B}$ (for finite τ). Hence the half-plane $ORCA^{\tau}_{A|B}$ always includes at least velocity $\mathbf{0}$. In fact, the line delimiting $ORCA^{\tau}_{A|B}$ is perpendicular to the line connecting the current positions of A and B.
  A drawback of setting the optimization velocity to zero is that the behavior of the robots is unconvincing, as they only take into account the current positions of the other robots, and not their current velocities. In densely packed conditions, this may also lead to a global deadlock, as the chosen velocities for the robots converge to zero when the robots are very close to one another.
- $\mathbf{v}^{\text{opt}}_A = \mathbf{v}^{\text{pref}}_A$ (i.e. the preferred velocity) for all robots A. The preferred velocity is part of the internal state of the robots, so it cannot be observed by the other robots. Let us, for the sake of the discussion, assume that it is somehow possible to infer the preferred velocity of the other robots, and that the optimization velocity is set to the preferred velocity for all robots. This would work well in low-density conditions, but, as the *magnitude* of the optimization velocity increases, it is increasingly more likely that the linear program is infeasible. As

Well that's no fun...

Somewhat defeats the purpose of decentralized...?

in most cases the preferred velocity has a constant (large) magnitude, regardless of the density conditions, this would lead to unsafe navigation in even medium density conditions.

- $\mathbf{v}_A^{\text{opt}} = \mathbf{v}_A$ (i.e. the current velocity) for all robots A. Setting the optimization to the current velocity is the ideal trade-off between the above two choices, as the current velocity automatically adapts to the situation: it is (very) indicative of the preferred velocity in low-density cases, and is close to zero in dense scenarios. Also, the current velocity can be observed by the other robots. Nevertheless, the linear program may be infeasible in high-density conditions (see Fig. 5(b)). In this case, choosing a collision-free velocity cannot be guaranteed. Instead, we select the 'safest possible' velocity for the robot using a 3-D linear program (which we discuss in Section 5.3).

Probably best to try this, then plan B would be $V_{opt}=0$

5.3 Densely Packed Conditions

Can probably ignore this scenario... Our usage won't have very robots at a time, so a densely packed condition would be unlikely.

If we choose $\mathbf{v}_A^{\text{opt}} = \mathbf{v}_A$ for all robots A, there might not be a single velocity that satisfies all the constraints of the linear program in situations where the density of the robots is very high. In other words, the set $ORCA_A^\tau$ is empty (see Fig. 5(b)), and the algorithm of Section 5.1 returns that the linear program is infeasible. In this case, choosing a collision-free velocity cannot be guaranteed. Instead, we select the 'safest possible' velocity for the robot, i.e. the velocity that minimally 'penetrates' the constraints induced by the other robots. More formally, let $d_{A|B}(\mathbf{v})$ denote the *signed* (Euclidean) distance of velocity $\mathbf{v}$ to the edge of the half-plane $ORCA_{A|B}^\tau$. If $\mathbf{v} \in ORCA_{A|B}^\tau$, then $d_{A|B}(\mathbf{v})$ is negative. We now choose the velocity that minimizes the maximum distance to any of the half-planes induced by the other robots:

$$\mathbf{v}_A^{\text{new}} = \underset{\mathbf{v} \in D(\mathbf{0}, v_A^{\max})}{\arg\min} \max_{B \neq A} d_{A|B}(\mathbf{v}). \tag{10}$$

Geometrically, this can be interpreted as moving the edges of the half-planes $ORCA_{A|B}^\tau$ perpendicularly outward with equal speed, until exactly one velocity becomes valid.

We can find this velocity using a *three-dimensional* linear program. For each other robot B, the distance function $d_{A|B}(\mathbf{v})$ is a plane in the three-dimensional $(\mathbf{v}, d)$ space. We now look for a point $(\mathbf{v}^*, d^*)$ that lies *above* all planes induced by the distance functions, and has a minimal d-value. Our new velocity $\mathbf{v}_A^{\text{new}}$ is then set to $\mathbf{v}^*$.

We can use the same randomized algorithm as above to solve this 3-D linear program. It still runs in $O(n)$ expected time, where n is the number of other robots. In fact, we can project the problem down on the $\mathbf{v}$-plane, such that all geometric operations can be performed in 2-D. The 3-D linear program is always feasible, so it always returns a solution.

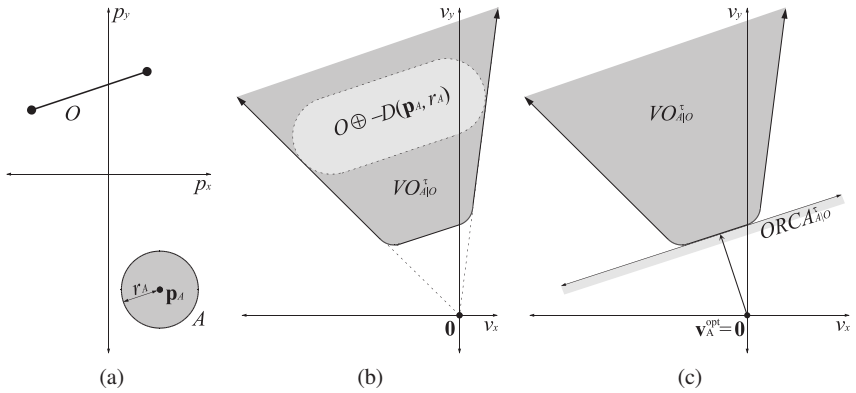


Fig. 6 **(a)** A configuration of a robot A and a line-segment obstacle O. **(b)** Geometric construction of the velocity obstacle $VO^{\tau}_{A|O}$ for $\tau = 2$. **(c)** The delimiting line of the half-plane $ORCA^{\tau}_{A|O}$ is tangent to $VO^{\tau}_{A|O}$ at the closest point on its boundary to $\mathbf{v}_A^{\text{opt}}$, which equals $\mathbf{0}$ in the case of obstacles.

Note that in these dense cases, the new velocity selected for the robot does not depend on the robot's preferred velocity. This means that the robot 'goes with the flow', and its behavior is fully determined by the robots surrounding the robot.

5.4 Static Obstacles

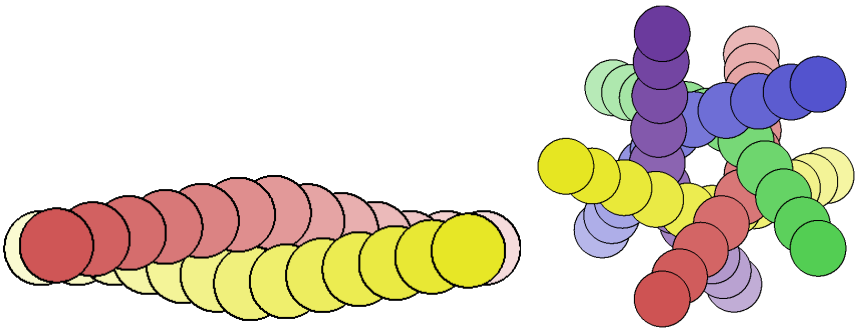
Will come back to this if we decide to include static obstacles, I know it was something we had never finalized.

So far we have only discussed how robots avoid collisions with each other, but typical multi-robot environments also contain (static) obstacles. We can easily incorporate those in the above framework. We basically follow the same approach as above, with a key difference being that obstacles do not move, so the robots should take full responsibility of avoiding collisions with them.

We can generally assume that obstacles are modeled as a collection of line segments. Let O be one of such line segments, and let A be a robot with radius r_A positioned at $\mathbf{p}_A$. Then, the velocity obstacle $VO^{\tau}_{A|O}$ induced by the obstacle O is defined as (see Fig. 6(a) and (b)):

$$VO^{\tau}_{A|O} = \{\mathbf{v} \,|\, \exists t \in [0,\tau] :: t\mathbf{v} \in O \oplus -D(\mathbf{p}_A, r_A)\}. \tag{11}$$

Agent A will collide with obstacle O within τ time if its velocity $\mathbf{v}_A$ is inside $VO^{\tau}_{A|O}$, and it will be collision-free for at least τ time if its velocity is outside the velocity obstacle. Hence, we could define the region of permitted velocities for A with respect to O as the complement of $VO^{\tau}_{A|O}$. However, this would disallow us to use the efficient linear programming algorithm of Section 5.1, as the complement of $VO^{\tau}_{A|O}$ is a *non-convex* region. Therefore, we define the set of permitted velocities for A



(a) (b)

Fig. 7 Trace of robots in two small behavioral simulations. Robots are shown as colored disks which are light at their initial positions and darken as time progresses. **(a)** Trace of two simulated robots passing each other. **(b)** Trace of five simulated robots crossing each other to antipodal points in a circle.

with respect to O as the half-plane $ORCA^{\tau}_{A|O}$ whose delimiting line is the *tangent* line to $VO^{\tau}_{A|O}$ at the closest point to $\mathbf{v}^{\text{opt}}_A$ on the boundary of $VO^{\tau}_{A|O}$ (see Fig. 6(c)).

In case of obstacles, we choose $\mathbf{v}^{\text{opt}}_A = \mathbf{0}$ for all robots A. This guarantees that there always exists a valid velocity for the robot that avoids collisions with the obstacles within τ time. We can typically use a smaller value of τ with respect to obstacles than with respect to other robots, as robots should typically not be 'shy' to move towards an obstacle if this is necessary to avoid other robots. On the other hand, the constraints on the permitted velocities for the robot with respect to obstacles should be *hard*, as collisions with obstacles should be avoided at all cost. Therefore, when the linear program of Section 5.1 is infeasible due to a high density of robots, the constraints of the obstacles are not relaxed.

We note that the half-planes of permitted velocities with respect to obstacles as defined above only make sure that the robot avoids collisions with the obstacle; they do not make the robot move around them. The direction of motion around obstacles towards the robot's goal should be reflected in the robot's *preferred velocity*, which could be obtained using (efficient) global path planning techniques.

6 Experimental Results

To test our technique we ran several simulations. We performed both small-scale simulations to test local behavior and large-scale simulations to analyze performance scaling.

Behavioral Results: We first show two scenarios which highlight how robots smoothly avoid collisions with each other on the local level. In the first, shown in Fig. 7(a), two robots exchange position. When the robots notice that a collision is imminent (i.e. it will happen within τ time), they change velocities to smoothly

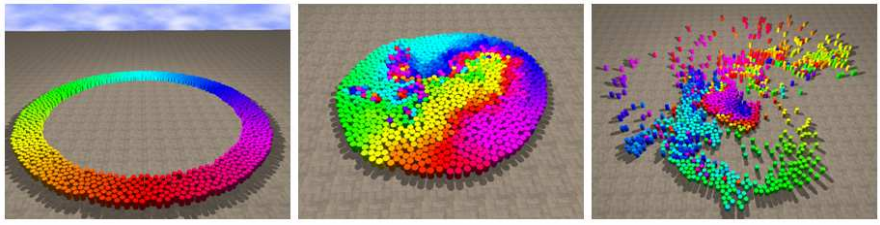


Fig. 8 Simulation of 1,000 agents trying to move through the center of a circle to antipodal positions. Robots smoothly move through the congestion that forms in the center.

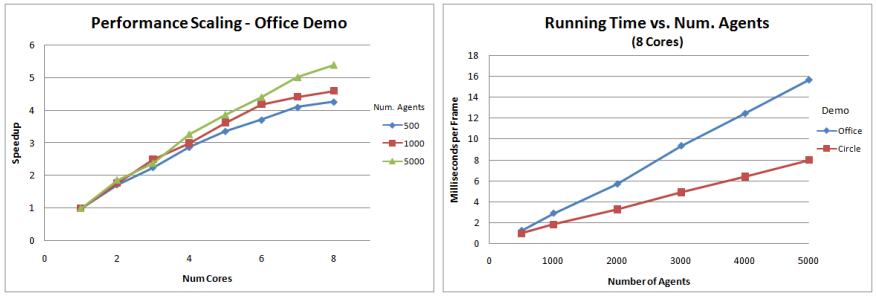
Fig. 9 Snapshots from simulation of 1,000 virtual agents evacuating an office as part of a crowd simulation.

avoid it. The second scenario shows five robots whose goal is to move across a circle to the antipodal position. As Fig. 7(b) shows, the robots smoothly spiral around each other to avoid collisions.

Performance Results: In order to test the performance of our method we ran two large-scale simulations. The first test was a simulation of 1,000 agents in a large circle moving to antipodal positions as shown in Fig. 8. For the second test, shown in Fig. 9, we incorporated our optimal reciprocal collision avoidance formulation into the existing crowd simulation framework of [10]. In this simulation, virtual agents attempt to evacuate an office environment. The preferred velocity for each agent is set to follow a globally-planned path out of the office.

Because each agent makes independent decisions, we are able to efficiently parallelize the simulation by distributing the computations for agents across multiple processors. We used OpenMP multi-threading to parallelize key computations across eight Intel Xeon 2.66GHz (Clovertown) cores. Fig. 10(a) shows how our method scales across various numbers of cores in the Office scenario. There is a fairly good scaling in all scenarios – with best observed results in nearly linear scaling for a large number of agents where the constant system overhead becomes far less significant in the overall computation time.

In terms of absolute performance, Fig. 10(b) shows the running time for various numbers of agents for both simulations. For 5,000 agents on eight cores, it takes 8 ms (125 frames per second) to solve the collision-avoidance linear program for every agent in the large circle simulation, and 15.6 ms (64 frames per second) to update every agent in the office evacuation simulation.



(a) (b)

Fig. 10 Performance Graphs: **(a)** Performance scaling on the evacuation simulation for 1 to 8 cores. **(b)** Runtime for various number of agents on 8 cores (lower is better). Both simulations scale approximately linearly with the number of agents.

7 Conclusion and Future Work

In this paper, we have presented an efficient method that provides a sufficient condition for multiple robots to select an action that avoids collisions with other robots, though each acts independently without communication with others. Our approach to reciprocal *n*-body collision avoidance exhibits fast running times and smooth, convincing behavior in our experiments.

We have used a simple robot model, in which kinematics and dynamics are ignored. An important extension for future work is to take such constraints into account. We can either do this as a post-processing step, in which the computed new velocity is 'clamped' to what the kinematic and dynamic constraints allow. This would not strictly guarantee avoiding collisions anymore, but it may work well in practice [25]. A more thorough solution would be to take these factors intrinsically into account in the derivation of the permitted velocities for the robots. [27] and [19] provide some interesting ideas in this direction.

In this paper, we have demonstrated results for only 2-D environments. However, all definitions and the algorithm can be extended to 3-D. This may be interesting for applications such as autonomous aerial vehicles, or flocking simulation of birds or fish. Another important direction for future work is to implement the presented framework on real robots and incorporate sensing uncertainty. This has been done for reciprocal velocity obstacles in [25]. We believe that we can relatively easily replace the RVO formulation by our ORCA formulation in that implementation.

References

1. Y. Abe, M. Yoshiki. Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. *IEEE RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1207-1212, 2001.
2. J. Borenstein, Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation* 7(3):278-288, 1991.

3. M. de Berg, O. Cheong, M. van Kreveld, M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.
4. B. Faverjon, P. Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. *IEEE Int. Conf. Robot. Autom.*, pp. 1152–1159, 1987.
5. P. Fiorini, Z. Shiller. Motion planning in dynamic environments using Velocity Obstacles. *Int. Journal of Robotics Research* 17(7), pp. 760-772, 1998.
6. D. Fox, W. Burgard, S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* 4:23-33, 1997.
7. T. Fraichard, H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics* 18(10):10011024, 2004.
8. C. Fulgenzi, A. Spalanzani, C. Laugier. Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. *IEEE Int. Conf. Robot. Autom.*, pp. 1610-1616, 2007.
9. J. Gil de Lamadrid. Avoidance of Obstacles With Unknown Trajectories: Locally Optimal Paths and Periodic Sensor Readings. *Int. Journal of Robotics Research* 13(6):496–507, 1994.
10. S. Guy, J. Chhugani, C. Kim, N. Satish, P. Dubey, M. Lin, D. Manocha. Highly parallel collision avoidance for multi-agent simulation. *University of North Carolina at Chapel Hill, Tech. Rep.*, 2009.
11. D. Helbing, I. Farkas, T. Vicsek. Simulating dynamical features of escape panic. *Nature* 407:487–490, 2000.
12. D. Hsu, R. Kindel, J. Latombe, S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robot. Res.* 21(3):233-255, 2002.
13. F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, J.-P. Laumond. A local collision avoidance method for non-strictly convex polyhedra. *Robotics: Science and Systems*, 2008.
14. O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. Journal of Robotics Research* 5(1):90–98, 1986.
15. B. Kluge, E. Prassler. Reflective navigation: Individual behaviors and group behaviors. *IEEE Int. Conf. Robot. Autom.*, pp. 4172-4177, 2004.
16. J. Kuchar, L. Chang. Survey of conflict detection and resolution modeling methods. *AIAA Guidance, Navigation, and Control Conf.*, 1997.
17. M. Lin. *Efficient collision detection for animation and robotics*. PhD thesis, University of California, Berkeley, 1993.
18. S. LaValle, Planning Algorithms. Cambridge University Press, 2006.
19. L. Martinez-Gomez, T. Fraichard. Collision avoidance in dynamic environments: an ICS-based solution and its comparative evaluation. *IEEE Int. Conf. on Robotics and Automation*, 2009.
20. J. McLurkin, E. Demaine. A Distributed Boundary Detection Algorithm for Multi-Robot Systems. *(under review)*, 2009.
21. J. Pettré, P. de Heras Ciechomski, J. Maïm, B. Yershin, J.-P. Laumond, D. Thalmann. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds* 17:445–455, 2006.
22. S. Petti, T. Fraichard. Safe motion planning in dynamic environments. *IEEE RSJ Int. Conf. Intell. Robot. Syst.*, pp. 2210-2215, 2005.
23. C. Reynolds. Flocks, herds and schools: A distributed behavioral model. *Int. Conf. on Computer Graphics and Interactive Techniques*, pp. 25–34, 1987.
24. R. Simmons. The curvature-velocity method for local obstacle avoidance. *IEEE Int. Conf. on Robotics and Automation*, pp. 3375–3382, 1996.
25. J. Snape, J. van den Berg, S. Guy, D. Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2009.
26. J. van den Berg, M. Lin, D. Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. *IEEE Int. Conf. on Robotics and Automation*, pp. 1928–1935, 2008.
27. D. Wilkie, J. van den Berg, D. Manocha. Generalized velocity obstacles. *IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 2009.
28. M. Zucker, J. Kuffner, M. Branicky. Multipartite RRTs for rapid replanning in dynamic environments. *IEEE Int. Conf. on Robotics and Automation*, pp. 1603-1609, 2007.