2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation

Jur van den Berg Ming Lin Dinesh Manocha

Abstract— **In this paper, we propose a new concept — the "Reciprocal Velocity Obstacle"— for real-time multi-agent navigation. We consider the case in which each agent navigates independently without explicit communication with other agents. Our formulation is an extension of the Velocity Obstacle concept [3], which was introduced for navigation among (passively) moving obstacles. Our approach takes into account the reactive behavior of the other agents by implicitly assuming that the other agents make a similar collision-avoidance reasoning. We show that this method guarantees safe and oscillation-free motions for each of the agents. We apply our concept to navigation of hundreds of agents in densely populated environments containing both static and moving obstacles, and we show that real-time and scalable performance is achieved in such challenging scenarios.**

I. Introduction

Recently, multi-agent systems have been gaining increasing attention, especially to carry out tasks that can be done more efficiently and effectively with a team of agents such as assembly, demining, search and rescue, etc. Other than control and coordination of multiple agents, one of the central problems in this area is motion planning among multiple moving agents. In this paper, we address the problem of real-time navigation for multi-agent motion planning in dynamic environments containing both static and moving obstacles. Each agent navigates *independently* without explicit communication with the other agents. Therefore, we can formulate the basic problem as navigating a single agent to its goal location without colliding with the obstacles and the other agents in the environment.

This problem is not only of interest to robotics but also has been widely studied for crowd simulation in computer graphics, virtual environments, video gaming, traffic engineering and architecture design, where each agent can be considered as a virtual human, a moving car, or an individual pedestrian. A common approach to this problem is continuous navigation. It involves a continuous cycle of sensing and acting, and during each cycle, each agent 'makes a move' based on its observation of its surroundings. Global path planning and local collision avoidance are often decoupled in this scheme. Typically, a global path to the goal location indicates the global direction of motion, while collisions with other agents and obstacles are avoided by locally navigating around them.

The local collision avoidance technique is an important module for these planners, and many approaches have been proposed. However, often these approaches deal with obstacles that are assumed to move passively through the environment without perception of their surroundings. In a multi-agent setting, this assumption does not hold as the agents do perceive each other, and actively adapt their motions accordingly. When each of the agents does not take into account that the other agents also have the decision-making ability to avoid collisions, the resulting motion is prone to contain undesirable and unrealistic *oscillations*. Although this problem has been identified by several prior works (see, e.g., [2], [1], [9]), no good solution is known for safe and oscillation-free navigation among multiple agents in large, cluttered environments.

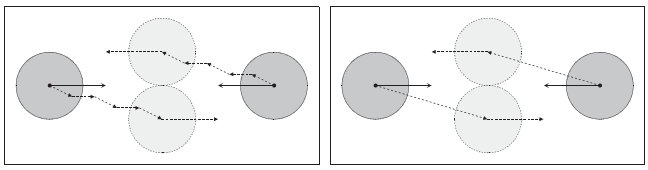

Fig. 1. The paths followed by two agents that have opposite preferred velocities and are on a head-on collision course, using the original Velocity Obstacle concept (left) and the Reciprocal Velocity Obstacle concept (right).

Main Results: In this paper, we introduce a new concept for local reactive collision avoidance called the *Reciprocal Velocity Obstacle*, which implicitly assumes that the other agents make a similar collision-avoidance reasoning. Under this assumption, our framework is guaranteed to generate safe and oscillation-free motions.

Our method is an extension of the Velocity Obstacle concept, introduced by Fiorini and Shiller in [3], which is a generally applicable, well-defined, and simple technique that has been widely used for safe navigation among moving obstacles (see, e.g., [16], [9], [5]). Our approach inherits all of its appealing properties, but we introduce an important new capability to resolve the common oscillation problem in multi-agent navigation.

The only information each agent is required to have about the other agents is their current position and velocity, and their exact shape (which can be acquired by sensors). We assume that the agents and the obstacles are translating objects in the 2-D plane (e.g. discs or polygons). This assumption is applicable to most applications with mobile agents, whose orientation can be inferred from the heading of each agent's motion.

We show the potential of the Reciprocal Velocity Obstacle approach by applying it to scenarios in which hundreds of similar agents navigate independently in a complex environment. Our experiments show that smooth and realistic motions are generated even when the agents form very dense,

This research is supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583, and 0636208, DARPA/RDECOM Contract N61339-04-C-0043, and Intel.

The authors are with the Department of Computer Science, University of North Carolina at Chapel Hill, USA. E-mail: {berg, lin, dm}@cs.unc.edu. Project website (including videos): http://gamma.cs.unc.edu/RVO.

packed groups. Moreover, real-time performance can be achieved in such challenging scenarios, and the approach is especially well suitable for parallelization, as an independent computation is performed for each agent.

Organization: The rest of this paper is organized as follows. We give a brief overview of prior work in Section II. In Section III, we review the concept of Velocity Obstacles and show that it generates oscillations when it is used for multi-agent navigation. In Section IV, we present our new concept, the *Reciprocal Velocity Obstacle* and show that it generates safe and oscillation-free motions. In Section V, we describe how we use this method for navigating many agents in several challenging environments containing both static and moving obstacles. We demonstrate the real-time performance of our approach on several benchmarks in Section VI and conclude in Section VII.

II. Prior Work

In this section, we give a brief overview of prior work on multi-agent navigation and planning. Besides the Velocity Obstacle approach [3], [16], many other methods have been proposed for collision-avoidance, navigation, and planning among moving obstacles [4], [7], [8], [14], [23], [21], [20], [24]. However, most of the existing work do not take into account that the obstacles' motion may be affected by the presence of the agent. Some approaches consider the moving obstacles to be static and replan when it appears that they have moved. Such approaches are generally not able to plan safe paths among obstacles moving at high speeds.

There is also an extensive amount of literature on multi-agent navigation, in which each agent navigates individually among the other agents, which are considered as obstacles, e.g. [13], [19], [17], [6], [12]. Most of these techniques have focused on crowd simulation. Also in these cases, the other agents are assumed to be either passively moving obstacles or static obstacles. A number of approaches (roughly) follow the Velocity Obstacle concept to avoid other agents [2], [10].

We distinguish decoupled multi-agent *navigation* from centralized multi-agent *planning* here. In multi-agent planning, the composite configuration space of the agents is considered, and a path is centrally planned in this space (see, e.g., [11], [15], [18]). These works focus on different aspects of the problem (e.g., finding optimal coordinations) and are mostly not suited for on-line real-time application.

Only few attempts have been made to incorporate the reactive behavior of other entities into the navigation. Kluge and Prassler [9] proposed Recursive Velocity Obstacles. The idea is that the first agent chooses a velocity based on the expected behavior of the second agent, which in turn is acquired based on the expected behavior of the first agent, and so on, up to some level of recursion. However, this approach may not be able to address the oscillation problem well. In fact, the velocities chosen oscillate between odd and even levels of recursion and may not converge. Abe and Matsuo [1] proposed the Common Velocity Obstacle, which is defined in the 4-dimensional space of all combinations of velocities of two agents. It addresses the oscillation issue, but it is unclear how this notion is extended for use with multiple agents or how well it scales to more complex environments.

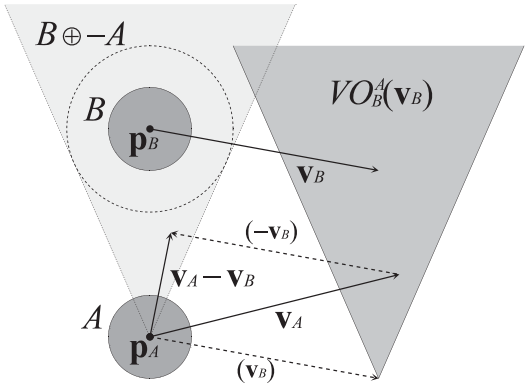

Fig. 2. The Velocity Obstacle $VO_B^A(\mathbf{v}_B)$ of a disc-shaped obstacle B to a disc-shaped agent A.

III. Velocity Obstacles

In this section, we briefly review the original concept of Velocity Obstacles (as introduced in [3]), derive some of its elementary properties, and show that it generates oscillatory motions when used in navigation among autonomous entities with a symmetric collision-avoidance strategy.

A. Velocity Obstacles: Definition

Let A be an agent translating in the plane with its reference point positioned at $\mathbf{p}_A$, and let B be a planar (moving) obstacle with its reference point positioned at $\mathbf{p}_B$. The velocity obstacle $VO_B^A(\mathbf{v}_B)$ of obstacle B to agent A is then the set consisting of all those velocities $\mathbf{v}_A$ for A that will result in a collision at some moment in time with obstacle B moving at velocity $\mathbf{v}_B$.

The Velocity Obstacle can geometrically be defined as follows (see Fig. 2). Let $A \oplus B$ denote the Minkowski sum of two objects A and B, and let $-A$ denote the object A reflected in its reference point:

$$A \oplus B = \{\mathbf{a} + \mathbf{b} \,|\, \mathbf{a} \in A, \mathbf{b} \in B\},\ -A = \{-\mathbf{a} \,|\, \mathbf{a} \in A\}.$$

Let $\lambda(\mathbf{p}, \mathbf{v})$ denote the a ray starting at $\mathbf{p}$ and heading in the direction of $\mathbf{v}$:

$$\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + t\mathbf{v} \,|\, t \geq 0\}. \tag{1}$$

If the ray starting at $\mathbf{p}_A$ and heading in the direction of the relative velocity of A and B (which is $\mathbf{v}_A - \mathbf{v}_B$) intersects the Minkowski sum of B and $-A$ centered at $\mathbf{p}_B$, velocity $\mathbf{v}_A$ is in the velocity obstacle of B. Hence, the velocity obstacle of B to A is defined as follows:

Definition 1 (Velocity Obstacle)**.**
$VO_B^A(\mathbf{v}_B) = \{\mathbf{v}_A \,|\, \lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B) \cap B \oplus -A \neq \emptyset\}.$

This means that if $\mathbf{v}_A \in VO_B^A(\mathbf{v}_B)$, A and B will collide at some point in time. If $\mathbf{v}_A$ is outside the velocity obstacle of B, both objects will never collide. If $\mathbf{v}_A$ is on the boundary of the velocity obstacle, it will touch B at some moment in

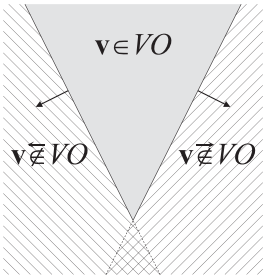

Fig. 3. The velocity obstacle (grey), and left and right half-planes (striped) outside the velocity obstacle. The symbols that apply to the velocities in each of the regions with respect to the velocity obstacle are shown.

time. The velocity obstacle is a cone with its apex at $\mathbf{v}_B$, as can be seen in Fig. 2.

The concept of Velocity Obstacles can be used for navigation among moving obstacles as follows. In each planning cycle, the agent chooses a velocity that lies outside any of the velocity obstacles induced by the moving obstacles. If among the free velocities, the velocity chosen is most directed towards the agent's goal position, the agent will safely navigate towards its goal (see, e.g. [3], [5]).

B. Velocity Obstacles: Properties

Here, we deduce some elementary properties and notations of velocity obstacles that we will use in this paper:

Lemma 2 (Symmetry)**.**
$\mathbf{v}_A \in VO^A_B(\mathbf{v}_B) \Leftrightarrow \mathbf{v}_B \in VO^B_A(\mathbf{v}_A)$.

Lemma 3 (Translation Invariance)**.**
$\mathbf{v}_A \in VO^A_B(\mathbf{v}_B) \Leftrightarrow \mathbf{v}_A + \mathbf{u} \in VO^A_B(\mathbf{v}_B + \mathbf{u})$.

These properties follow immediately from Definition 1.

Let us also consider the region outside the velocity obstacle. We distinguish the region to the *left* and the region to the *right* of the velocity obstacle, defined by the half-planes delimited by the two boundaries of the velocity obstacle (see Fig. 3). We introduce two new notations here, '$\overleftarrow{\notin}$' and '$\overrightarrow{\notin}$', and denote

$$\mathbf{v}_A \overleftarrow{\notin} VO^A_B(\mathbf{v}_B)$$

if $\mathbf{v}_A$ is in the half-plane to the *left* of $VO^A_B(\mathbf{v}_B)$. Such velocities let A pass B on the left side. Similarly, we denote

$$\mathbf{v}_A \overrightarrow{\notin} VO^A_B(\mathbf{v}_B),$$

if $\mathbf{v}_A$ is in the half-plane to the *right* of $VO^A_B(\mathbf{v}_B)$. These velocities let A pass B on the right side. Note that the left and the right half-planes overlap. Velocities in this region let A and B diverge.

Lemmas 2 and 3 also hold for the regions outside the velocity obstacle. That is, the '$\in$' in Lemmas 2 and 3 can freely be replaced by '$\notin$', '$\overleftarrow{\notin}$', or '$\overrightarrow{\notin}$'.

We prove the following property for the half planes (the '$\overrightarrow{\notin}$' can freely be replaced by '$\overleftarrow{\notin}$'):

Lemma 4 (Convexity)**.**
$\mathbf{v}_A \overrightarrow{\notin} VO^A_B(\mathbf{v}_B) \wedge \mathbf{v}'_A \overrightarrow{\notin} VO^A_B(\mathbf{v}_B) \Rightarrow$
$(1-\alpha)\mathbf{v}_A + \alpha\mathbf{v}'_A \overrightarrow{\notin} VO^A_B(\mathbf{v}_B)$, for $0 \leq \alpha \leq 1$.

This lemma follows from the fact that a half-plane is convex.

C. Oscillation

The Velocity Obstacle concept can be used for multi-agent navigation when each agent regards the other agents as moving obstacles and chooses a velocity for itself that lies outside any of the velocity obstacles induced by the other agents (see, e.g., [2], [10]). However, this approach results in undesirable oscillatory motions, as we show here.

Imagine the following situation. Two agents A and B are moving with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$, respectively, such that $\mathbf{v}_A \in VO^A_B(\mathbf{v}_B)$ and $\mathbf{v}_B \in VO^B_A(\mathbf{v}_A)$. Hence, continuing along the current velocities will result in a collision. As a result, agent A decides to alter its velocity to $\mathbf{v}'_A$, such that it is outside the velocity obstacle of B (i.e., $\mathbf{v}'_A \notin VO^A_B(\mathbf{v}_B)$). At the same time, B alters its velocity to $\mathbf{v}'_B$ to be outside the velocity obstacle of A (i.e., $\mathbf{v}'_B \notin VO^B_A(\mathbf{v}_A)$).

However, in the new situation, the old velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ are outside the velocity obstacles of B and A, respectively (i.e., $\mathbf{v}_A \notin VO^A_B(\mathbf{v}'_B)$ and $\mathbf{v}_B \notin VO^B_A(\mathbf{v}'_B)$). This follows directly from Lemma 2. If both agents prefer the old velocities, for instance because it leads them directly to their goals, they will choose these again. In the next cycle, it appears that these velocities will result in a collision, and they will probably choose $\mathbf{v}'_A$ and $\mathbf{v}'_B$ again, and so on. Thus, the agents oscillate between these two velocities when the Velocity Obstacle approach is used to avoid each other (see Fig. 1), even if the agents initially choose the same side to pass each other.[1]

IV. Reciprocal Velocity Obstacles

In this section, we present a new concept called the *Reciprocal Velocity Obstacle* to overcome the oscillation problem mentioned above. It provides a simple approach to safely and smoothly navigate multiple agents amongst each other without explicit communication between them.

A. Reciprocal Velocity Obstacles: Definition

The basic idea is simple: instead of choosing a new velocity for each agent that is outside the other agent's velocity obstacle, we choose a new velocity that is the *average* of its current velocity and a velocity that lies outside the other agent's velocity obstacle.

We formalize this principle and propose the concept of the *Reciprocal Velocity Obstacle*, which is defined as follows:

Definition 5 (Reciprocal Velocity Obstacle)**.**
$RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v}'_A \,|\, 2\mathbf{v}'_A - \mathbf{v}_A \in VO^A_B(\mathbf{v}_B)\}$.

The reciprocal velocity obstacle $RVO^A_B(\mathbf{v}_B, \mathbf{v}_A)$ of agent B to agent A contains all velocities for agent A that are the average of the current velocity $\mathbf{v}_A$ and a velocity inside the velocity obstacle $VO^A_B(\mathbf{v}_B)$ of agent B. It can geometrically be interpreted as the velocity obstacle $VO^A_B(\mathbf{v}_B)$ that is translated such that its apex lies at $\frac{\mathbf{v}_A+\mathbf{v}_B}{2}$ (see Fig. 4).

[1]Note that these oscillations are fundamentally different from "reciprocal dances", which occur when there is no agreement among the agents about which side to pass each other on.

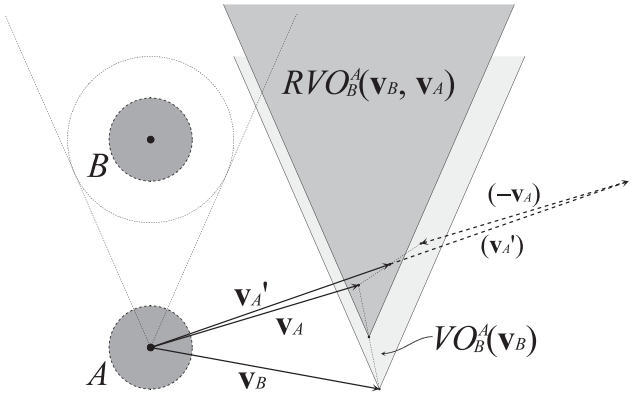

Fig. 4. The Reciprocal Velocity Obstacle $RVO^A_B(\mathbf{v}_B, \mathbf{v}_A)$ of agent B to agent A.

B. Guarantees

We now prove that the Reciprocal Velocity Obstacles can be used to generate collision-free and oscillation-free motions for each agent.

1) Collision-Free Navigation: Let $\mathbf{v}_A$ be the current velocity of agent A, and let $\mathbf{v}_B$ be the current velocity of agent B, and let both A and B choose new velocities ($\mathbf{v}'_A$ and $\mathbf{v}'_B$) outside each other's reciprocal velocity obstacle. The following theorem proves that this is safe, provided that both agents choose the same side to pass each other (the '$\overrightarrow{\notin}$' can freely be replaced by '$\overleftarrow{\notin}$'):

Theorem 6 (Collision-Free)**.**

$$\mathbf{v}'_A \overrightarrow{\notin} RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) \wedge \mathbf{v}'_B \overrightarrow{\notin} RVO^B_A(\mathbf{v}_A, \mathbf{v}_B) \Rightarrow \mathbf{v}'_A \overrightarrow{\notin} VO^A_B(\mathbf{v}'_B) \wedge \mathbf{v}'_B \overrightarrow{\notin} VO^B_A(\mathbf{v}'_A)$$

Proof: $\mathbf{v}'_A \overrightarrow{\notin} RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) \wedge \mathbf{v}'_B \overrightarrow{\notin} RVO^B_A(\mathbf{v}_A, \mathbf{v}_B)$
$\Leftrightarrow$ {Definition 5 and Lemma 2}
$2\mathbf{v}'_A - \mathbf{v}_A \overrightarrow{\notin} VO^A_B(\mathbf{v}_B) \wedge \mathbf{v}_A \overrightarrow{\notin} VO^A_B(2\mathbf{v}'_B - \mathbf{v}_B)$
$\Leftrightarrow$ {Lemma 3}
$2\mathbf{v}'_A - \mathbf{v}_A - \mathbf{v}_B \overrightarrow{\notin} VO^A_B(\mathbf{0}) \wedge \mathbf{v}_A + \mathbf{v}_B - 2\mathbf{v}'_B \overrightarrow{\notin} VO^A_B(\mathbf{0})$
$\Rightarrow$ {Lemma 4, with $\alpha = \frac{1}{2}$}
$\mathbf{v}'_A - \mathbf{v}'_B \overrightarrow{\notin} VO^A_B(\mathbf{0})$
$\Leftrightarrow$ {Lemma 3 and Lemma 2}
$\mathbf{v}'_A \overrightarrow{\notin} VO^A_B(\mathbf{v}'_B) \wedge \mathbf{v}'_B \overrightarrow{\notin} VO^B_A(\mathbf{v}'_A)$ □

2) Same Side: We can guarantee that both agents automatically choose the same side to pass each other if each of them chooses the velocity outside the other agent's reciprocal velocity obstacle that is *closest* to its current velocity.

This is proven by the following facts: (1) If for agent A, $\mathbf{v}_A + \mathbf{u}$ is the velocity closest to $\mathbf{v}_A$ outside B's reciprocal velocity obstacle, then for agent B, $\mathbf{v}_B - \mathbf{u}$ is the velocity closest to $\mathbf{v}_B$ outside A's reciprocal velocity obstacle. (2) If for agent A, this closest velocity appears to be on the right (or left) side of B's reciprocal velocity obstacle, then the closest velocity for agent B is on the right (or left) side of A's reciprocal velocity obstacle as well. These facts both follow from the following lemma (the '$\notin$' can freely be replaced by '$\overrightarrow{\notin}$', '$\overleftarrow{\notin}$', or '$\in$'):

Lemma 7 (Same Side)**.**

$$\mathbf{v}_A + \mathbf{u} \notin RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) \Leftrightarrow \mathbf{v}_B - \mathbf{u} \notin RVO^B_A(\mathbf{v}_A, \mathbf{v}_B)$$

Proof: $\mathbf{v}_A + \mathbf{u} \notin RVO^A_B(\mathbf{v}_B, \mathbf{v}_A)$
$\Leftrightarrow$ {Definition 5}
$2(\mathbf{v}_A + \mathbf{u}) - \mathbf{v}_A \notin VO^A_B(\mathbf{v}_B)$
$\Leftrightarrow$ {Lemma 3 and Lemma 2}
$2(\mathbf{v}_B - \mathbf{u}) - \mathbf{v}_B \notin VO^B_A(\mathbf{v}_A)$
$\Leftrightarrow$ {Definition 5}
$\mathbf{v}_B - \mathbf{u} \notin RVO^B_A(\mathbf{v}_A, \mathbf{v}_B)$ □

3) Oscillation-Free Navigation: Choosing the closest velocity outside the other agent's reciprocal velocity obstacle also guarantees oscillation-free navigation. This is proven by the following theorem:

Theorem 8 (Oscillation-Free)**.**

$$\mathbf{v}_A \in RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) \Leftrightarrow \mathbf{v}_A \in RVO^A_B(\mathbf{v}_B - \mathbf{u}, \mathbf{v}_A + \mathbf{u})$$

Proof: $\mathbf{v}_A \in RVO^A_B(\mathbf{v}_B, \mathbf{v}_A)$
$\Leftrightarrow$ {Definition 5 and Lemma 3}
$2\mathbf{v}_A - \mathbf{v}_A - \mathbf{v}_B \in VO^A_B(\mathbf{0})$
$\Leftrightarrow$
$2\mathbf{v}_A - \mathbf{v}_A - \mathbf{v}_B - \mathbf{u} + \mathbf{u} \in VO^A_B(\mathbf{0})$
$\Leftrightarrow$ {Lemma 3 and Definition 5}
$\mathbf{v}_A \in RVO^A_B(\mathbf{v}_B - \mathbf{u}, \mathbf{v}_A + \mathbf{u})$ □

Hence, the old velocity $\mathbf{v}_A$ of A is inside the new reciprocal velocity obstacle of B, given the new velocities $\mathbf{v}_A + \mathbf{u}$ and $\mathbf{v}_B - \mathbf{u}$ for agent A and B, respectively.[2] The same holds for agent B. Therefore, after choosing the new velocity, the old (preferred) velocity is invalid and will not be chosen (in contrast to when the orginal Velocity Obstacles are used – see Section III-C). In fact, by choosing the closest velocity outside the reciprocal velocity obstacle for both A and B, the reciprocal velocity obstacles stay exactly in the same position. Hence, the velocities $\mathbf{v}_A + \mathbf{u}$ and $\mathbf{v}_B - \mathbf{u}$ are still the closest to the preferred velocities among all valid velocities. As a result, no oscillations will occur (see Fig. 1).

C. Generalized Reciprocal Velocity Obstacles

In the above we have implicitly assumed that each agent takes an equal share in the effort to avoid mutual collisions. However, there may be natural priorities among agents that motivate a different balance, and we can generalize the Reciprocal Velocity Obstacle concept accordingly. Let us denote the share of the effort agent A takes to avoid agent B by α^A_B. By definition, $\alpha^B_A = 1 - \alpha^A_B$ (thus far, we implicitly assumed $\alpha^A_B = \alpha^B_A = \frac{1}{2}$).

The idea is that agent A chooses a new velocity that is the *weighted* average of $1-\alpha^A_B$ of its current velocity $\mathbf{v}_A$ and α^A_B of a velocity outside the velocity obstacle $VO^A_B(\mathbf{v}_B)$ of agent B, and that agent B does exactly the opposite, i.e., choosing a new velocity that is the weighted average of $1 - \alpha^B_A = \alpha^A_B$ of its current velocity $\mathbf{v}_B$ and $\alpha^B_A = 1 - \alpha^A_B$ of a velocity outside the velocity obstacle $VO^B_A(\mathbf{v}_A)$ of agent A.

[2]There is a more generous set of new velocities that will not result in oscillations. These are all pairs of velocities $\mathbf{v}'_A$ and $\mathbf{v}'_B$, such that $(\mathbf{v}'_A + \mathbf{v}'_B)/2$ is both inside $VO^A_B(\mathbf{v}_B)$ and inside $VO^B_A(\mathbf{v}_A)$.

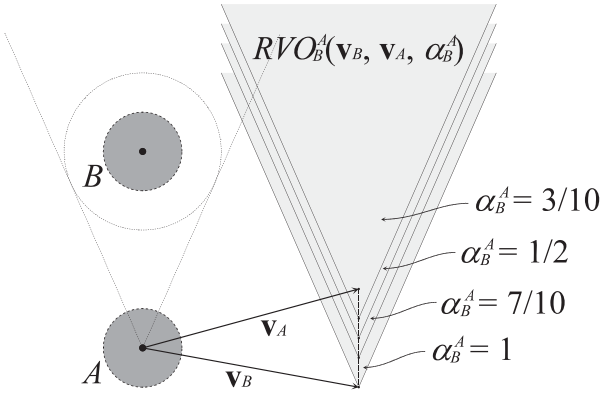

Fig. 5. The Generalized Reciprocal Velocity Obstacle $RVO^A_B(\mathbf{v}_B, \mathbf{v}_A, \alpha^A_B)$ of agent B to agent A for various values of α^A_B.

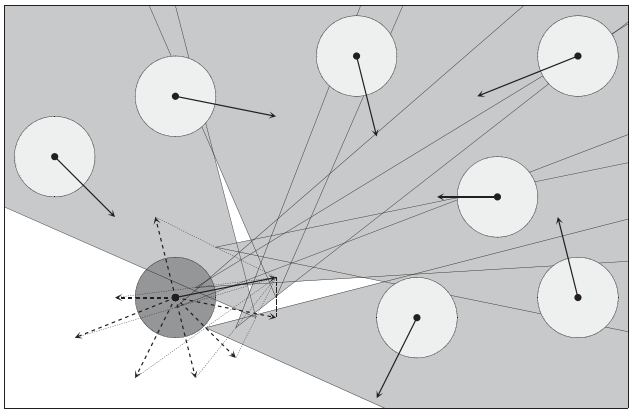

Fig. 6. The combined reciprocal velocity obstacle for the agent (dark) is the union of the individual reciprocal velocity obstacles of the other agents.

The Generalized Reciprocal Velocity Obstacle of agent B to agent A is defined as follows:

Definition 9 (Generalized Reciprocal Velocity Obstacle)**.**

$$RVO^A_B(\mathbf{v}_B, \mathbf{v}_A, \alpha^A_B) = \{\mathbf{v}'_A \mid \tfrac{1}{\alpha^A_B}\mathbf{v}'_A + (1 - \tfrac{1}{\alpha^A_B})\mathbf{v}_A \in VO^A_B(\mathbf{v}_B)\}.$$

It can geometrically be interpreted as the velocity obstacle $VO^A_B(\mathbf{v}_B)$, whose apex is translated to $(1-\alpha^A_B)\mathbf{v}_A + \alpha^A_B\mathbf{v}_B$ (see Fig. 4).

All the theorems we have proved above can easily be extended for the Generalized Reciprocal Velocity Obstacles. The main idea is that Lemma 4 not only holds for $\alpha = \frac{1}{2}$ but for any α between 0 and 1. So, also in the generalized case, the generated motions are safe and oscillation-free.

V. Multi-Agent Navigation

In this section, we show how the Reciprocal Velocity Obstacle concept can be used to simultaneously navigate a large number of agents to their goals in a common environment containing both static and moving obstacles. Given n (planar translating) agents $A_1, \ldots, A_n$, each agent A_i has a current position $\mathbf{p}_i$ (defined by its reference point), a current velocity $\mathbf{v}_i$, a goal location $\mathbf{g}_i$, and a preferred speed v^{pref}_i. Furthermore, let there be a set of (planar translating) obstacles $\mathcal{O}$, where each obstacle $O \in \mathcal{O}$ has current position $\mathbf{p}_O$ (defined by its reference point) and velocity $\mathbf{v}_O$. Static obstacles have zero velocity.

The overall approach is as follows. We choose a small amount of time Δt, which is the time step of the simulation. In each cycle of the simulation, we select for each agent independently a new velocity and update its position accordingly. This process continues until all of the agents have reached their goal positions.

We show in this section how to select a velocity for all agents, such that they safely navigate towards their goals.

A. Combined Reciprocal Velocity Obstacles

In the forgoing, we have seen how the Reciprocal Velocity Obstacle is defined for a pair of agents. If the concept is applied for an agent moving among many other agents and passively moving or static obstacles, the *combined* reciprocal velocity obstacle RVO^i for agent A_i becomes the *union* of all reciprocal velocity obstacles generated by the other agents individually, possibly with varying mutual priorities, *and* the velocity obstacles generated by the (passively) moving obstacles (See Fig. 6).

Definition 10 (Combined Reciprocal Velocity Obstacle)**.**

$$RVO^i = \bigcup_{j \neq i} RVO^i_j(\mathbf{v}_j, \mathbf{v}_i, \alpha^i_j) \cup \bigcup_{O \in \mathcal{O}} VO^i_O(\mathbf{v}_O).$$

Each agent can safely navigate by choosing a velocity outside its combined reciprocal velocity obstacle.

B. Kinematic and Dynamic Constraints

Each agent A_i may be subject to kinematic and dynamic constraints that restrict the set of admissible new velocities, given the current velocity $\mathbf{v}_i$. We denote this set $AV^i(\mathbf{v}_i)$. It may have any shape depending on the nature of the agent. For example, if the agent is subject to a maximum speed $v^{\max}_i$ and a maximum acceleration $a^{\max}_i$, the set of admissible velocities is:

$$AV^i(\mathbf{v}_i) = \{\mathbf{v}'_i \mid \|\mathbf{v}'_i\| < v^{\max}_i \wedge \|\mathbf{v}'_i - \mathbf{v}_i\| < a^{\max}_i \Delta t\}.$$

C. Selecting Velocities

In each cycle of the simulation, we start with computing for each agent A_i its preferred velocity $\mathbf{v}^{\text{pref}}_i$. This is the vector with a magnitude equal to the preferred speed in the direction of the target location. If the agent is close to its goal, we set the preferred velocity to the null vector.

Subsequently, we select for each agent A_i a new velocity $\mathbf{v}'_i$. Ideally, this is the velocity closest to $\mathbf{v}^{\text{pref}}_i$ that is outside the combined reciprocal velocity obstacle RVO^i and inside the set AV^i of admissible velocities. However, the environment may become so crowded that the combined reciprocal velocity obstacle fills up the entire set of admissible velocities. To address this issue, the algorithm is allowed to select a velocity inside RVO^i, but is penalized by this choice. The penalty of a candidate velocity $\mathbf{v}'_i$ depends on its distance to the preferred velocity and on the expected *time to collision* $tc_i(\mathbf{v}'_i)$ this velocity will give:

$$penalty_i(\mathbf{v}'_i) = w_i \frac{1}{tc_i(\mathbf{v}'_i)} + \|\mathbf{v}^{\text{pref}}_i - \mathbf{v}'_i\|,$$

for some factor w_i, where w_i can vary among the agents to reflect differences in aggressiveness and sluggishness.

The expected time to collision $tc_i(\mathbf{v}'_i)$ can easily be calculated. For the original velocity obstacle $VO^A_B(\mathbf{v}_B)$, the time to collision when A chooses velocity $\mathbf{v}_A$ inside $VO^A_B(\mathbf{v}_B)$ is computed by solving the equation $\lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B) = B \oplus -A$ for t (see Eq. (1)). This follows from Definition 1. For the reciprocal velocity obstacle $RVO^A_B(\mathbf{v}_B, \mathbf{v}_A)$, the expected time to collision when A chooses velocity $\mathbf{v}'_A$ is calculated similarly by solving $\lambda(\mathbf{p}_A, 2\mathbf{v}'_A - \mathbf{v}_A - \mathbf{v}_B) = B \oplus -A$ for t. This follows from Definition 5. For the combined reciprocal velocity obstacle, the expected time to collision is the minimum of all expected times to collision with respect to the individual other agents and obstacles, and infinity when there is no collision.

We select the velocity with minimal penalty among the velocities in AV^i as the new velocity $\mathbf{v}'_i$ for agent A_i:

$$\mathbf{v}'_i = \arg \min_{\mathbf{v}''_i \in AV^i} penalty_i(\mathbf{v}''_i).$$

We approximate this minimum by sampling a number N of velocities evenly distributed over AV^i.

D. Neighbor Region

We do not need to take all other agents into account when selecting a new velocity, as the penalty of the velocities will not depend much, if at all, on agents that are far away. Therefore, we define a neighbor region NR^i around the current position of agent A_i and only take into account the agents and obstacles inside this neighbor region in the combined reciprocal velocity obstacle. The optimal size of the neighbor region depends on the average speed of the agents and the obstacles, the size of the time step of the simulation, etc.

A neighbor region not only provides a speed-up for the computation; it may also be used to model natural (human) behavior. For example, the neighbor region may be restricted to the region that the agent can actually see, given the direction of motion of the agent, its view angle, and the position of the static obstacles (and perhaps the other agents).

VI. Experimental Results

We have implemented and tested our multi-agent navigation approach in three challenging scenarios:

- **Circle:** (See Figs. 7 and 8) A variable number of agents are distributed evenly on a circle, and their goal is to navigate to the antipodal position on the circle. In doing so, the agents will form a dense crowd in the middle.
- **Narrow passage:** (See Fig. 9) Four groups of 25 agents in each corner of the environment move to the opposite corner of the environment. In the middle, there are four square-shaped static obstacles that form narrow passages. The groups with opposite goal directions meet in the narrow passage.
- **Moving obstacle:** (See Fig. 10) A number of agents cross a street on which a car is driving. The car is considered to be a passively moving obstacle.

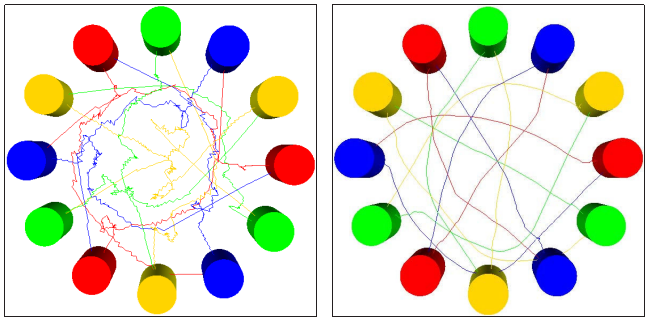

Fig. 7. The resulting paths in the *Circle* scenario using the original Velocity Obstacle approach (left) and the Reciprocal Velocity Approach (right).

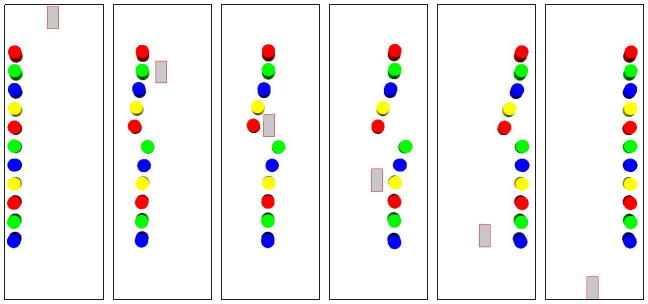

Fig. 10. Eleven agents cross a street on which a car is driving in the *Moving Obstacle* scenario.

We performed our experiments on a Intel Core2 Duo 1.66 GHz with 1GByte of memory. For the first benchmark, *Circle*, we started with an experiment containing 12 agents to show the difference between the original Velocity Obstacle approach and our Reciprocal Velocity Obstacle approach. All of the agents are discs with equal radii, have the same preferred and maximum speeds, and do not have constraints on the acceleration. In both experiments, all parameters are equal. The traces of the agents are shown in Fig. 7 for both methods. As can be seen clearly from the figures, the original Velocity Obstacle approach generates chaotic and oscillatory motions. In contrast, the motions generated by our method are smooth and straight-forward. Also, there were no collisions among agents.

Next, we varied the number of agents in the *Circle* benchmark to see how our approach scales when the number of agents grows (see Fig. 8 and the accompanying video for the experiment with 250 agents). In this case, we used a circular neighbor region around each agent. We chose the radius of this region such that the navigation of the agents is still safe (which was 8 times the agent radius). The results are given in Fig. 11.

Clearly, the amount of time needed to generate one frame (i.e., process one cycle in the simulation) scales linearly with the number of agents. Only the neighbor selecting routine has a quadratic nature, but this step is negligible in the total running time. The graph also shows that even for 1000 agents, we are able to generate more than 10 frames per second. We chose the time step Δt to be 0.25 seconds in this experiment, so these results are obtained in real-time

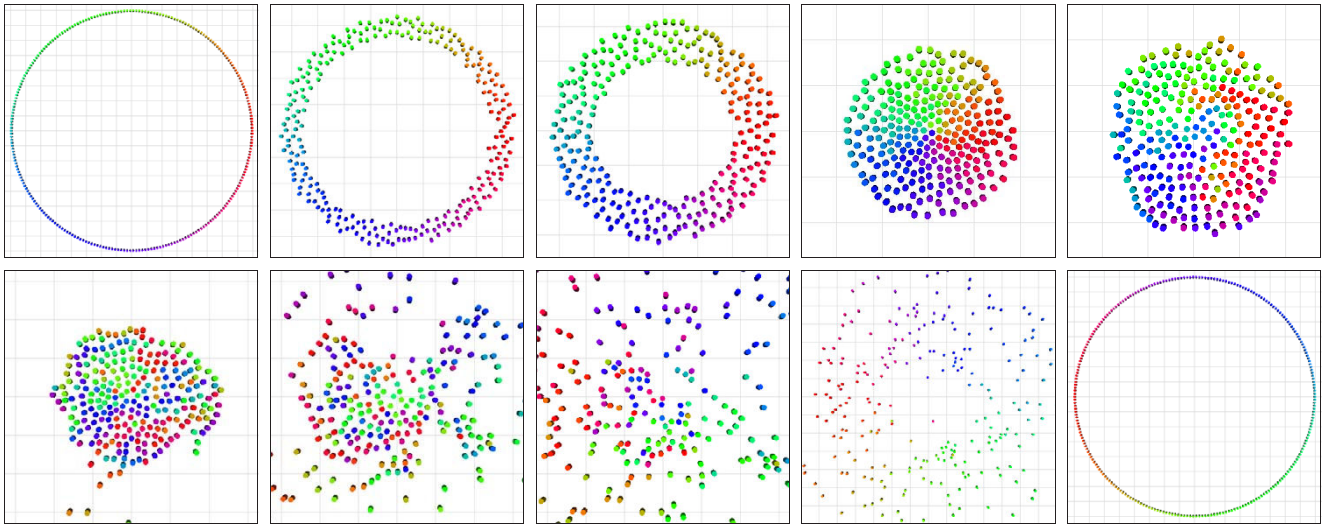

Fig. 8. The *Circle* scenario for 250 agents. Each agent moves to the diametrically opposite position on the circle (the zoom level varies between stills).

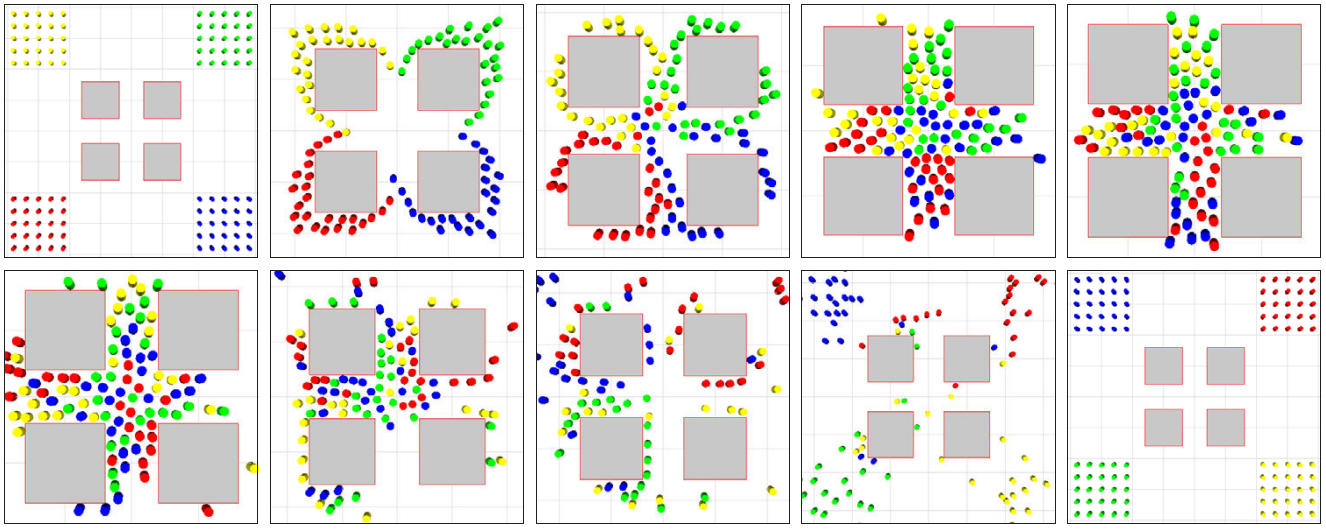

Fig. 9. Four groups in opposite corners of the environment exchange positions in the *Narrow Passage* scenario (the zoom level varies between stills).

frame rates. We note that the running time of our method scales linearly with the value of parameter N, the number of velocities sampled for each agent in each frame. We fixed this value at 250 in our experiments. We believe that smarter sampling can further improve the performance. Furthermore, since we perform an independent computation for each agent, the approach is fully parallelizable. We took advantage of this feature and used two processors for performing the experiments.

To see how our approach performs in presence of narrow passages generated by static obstacles, we performed the experiment in the *Narrow Passage* benchmark. The agents from the different groups will meet inside the narrow passage with opposite goal directions. Even though the passage becomes very crowded (see Fig. 9), eventually all agents reach their goals safely. We note that if the static obstacles would form a U-shaped obstacle, the agents may get stuck in there when using our method. To overcome this, the method can be extended with a roadmap that governs the preferred velocities of the agent (instead of the velocity directed towards the goal) [22].

Finally, we show in the *Moving Obstacle* benchmark that our approach easily deals with high-speed moving obstacles (which do not react on their surroundings). Eleven agents cross a street on which a car is driving. The ones that are not able to cross in front of the car wait until it has passed. The others move quickly before they are run over. See Fig. 10 for a series of screenshots from this simulation. Although this result could have been achieved using the original Velocity Obstacle approach, many contemporary works in multi-robot navigation cannot handle moving obstacles, especially when they move at high speeds. This is because only their position is taken into account and not their velocity.

Videos of these and other scenarios can be found at `http://gamma.cs.unc.edu/RVO`.

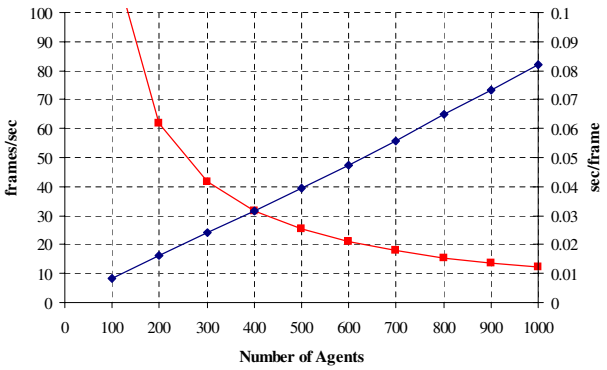

Fig. 11. Results of our experiments in the *Circle* scenario for a varying number of agents. The diamond-marked graph gives the run time per frame (right axis) and the square-marked graph gives the frame rate (left axis).

VII. Conclusion

In this paper, we have introduced the concept of Reciprocal Velocity Obstacles for safe and oscillation-free navigation among autonomous decision-making entities, as well as static and moving obstacles. It has been applied to multi-agent navigation and shown to compute smooth, natural paths at interactive rates for more than 1000 agents moving simultaneously in the same environment.

Although our current implementation is mainly for agents translating in the two-dimensional plane, the orientation of the agents can be can be inferred from the heading direction of the agent's motion. In addition, Reciprocal Velocity Obstacles can easily be extended for agents moving in the three-dimensional space.

This method offers several advantages over the existing techniques. First, our approach takes into account that other moving entities react to the agent; thereby it prevents oscillatory motions resulting from the assumption that the other entities are passively moving obstacles. Also, our approach automatically deals with high-speed moving obstacles. Furthermore, the Reciprocal Velocity Obstacle is a simple and natural formulation that is generally applicable and easy to implement. We plan to further investigate the application of Reciprocal Velocity Obstacles for multi-robot navigation and crowd simulation in complex, densely packed environments (see [22]).

References

[1] Y. Abe and Y. Matsuo, "Collision avoidance method for multiple autonomous mobile agents by implicit cooperation," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 1207–1212.

[2] F. Feurtey, "Simulating the collision avoidance behavior of pedestrians," Master's thesis, University of Tokyo, 2000.

[3] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[5] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1610–1616.

[6] R. Gayle, A. Sud, M. Lin, and D. Manocha, "Reactive deforming roadmaps: Motion planning of multiple robots in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.

[7] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[8] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2004, pp. 1606–1611.

[9] B. Kluge and E. Prassler, "Reflective navigation: Individual behaviors and group behaviors," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 4172–4177.

[10] F. Lamarche and S. Donikian, "Crowd of virtual humans: a new approach for real time navigation in complex and structured environments," *Computer Graphics Forum*, vol. 23, no. 3, pp. 509–518, 2004.

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

[12] Y. Li and K. Gupta, "Motion planning of multiple agents in virtual environments on parallel architectures," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1009–1014.

[13] J. Pettré, P. de Heras Ciechomski, J. Maïm, B. Yersin, J.-P. Laumond, and D. Thalmann, "Real-time navigating crowds: Scalable simulation and rendering," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 445–455, 2006.

[14] S. Petty and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005, pp. 3726–3731.

[15] G. Sánchez and J. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 2112–2119.

[16] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: obstacles moving along arbitrary trajectories," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 3716–3721.

[17] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha, "Real-time path planning for virtual agents in dynamic environments," in *Proc. IEEE Virtual Reality*, 2007.

[18] P. Švestka and M. Overmars, "Coordinated path planning for multiple robots," *Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 125–152, 1998.

[19] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," in *Proc. ACM SIGGRAPH*, 2006.

[20] J. van den Berg, "Path planning in dynamic environments," Ph.D. dissertation, Universiteit Utrecht, 2007.

[21] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 2366–2371.

[22] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of individual agents in crowded environments," in *Symposium on Interactive 3D Graphics and Games*, 2008.

[23] J. Vannoy and J. Xiao, "Real-time planning of mobile manipulation in dynamic environments of unknown changes," 2006.

[24] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1603–1609.