Goal Velocity Obstacles for Spatial Navigation of Multiple Autonomous Robots or Virtual Agents

Jamie Snape and Dinesh Manocha

University of North Carolina at Chapel Hill, Chapel Hill, NC 27599
{snape,dm}@cs.unc.edu

Abstract. We present the *goal velocity obstacle* for the spatial navigation of multiple autonomous robots or virtual agents, such as are found in mobile robotics, video games, and simulated environments, to planar goal regions in the two-dimensional workspace. Our approach uses the notion of velocity obstacles not only to compute collision-avoiding velocities with respect to other agents, but also to specify velocities that will direct an agent toward its spatial goal region. The goal velocity obstacle provides a unified formulation that allows for goals specified as points, line segments, and bounded, planar regions in two dimensions that may be static or moving. An agent may have multiple goal regions without requiring an explicit goal allocation algorithm that would choose a particular goal region to navigate toward in advance. We have applied our approach to experiments with hundreds of agents, demonstrating shorter path lengths and fewer collisions with only microseconds of additional computation per agent per time step than when using velocity-based methods that optimize on a single, preferred velocity toward the goal of each agent.

Keywords: multiagent planning, motion planning, mobile robot coordination, collective behaviors, implicit cooperation

1 Introduction

The spatial navigation of groups of multiple autonomous robots or virtual agents to specified goal locations is an important problem in the fields of mobile robotics, video games, and simulated environments. Teams of autonomous robots may cooperate to perform some task, such as surveillance or industrial warehousing, while large numbers of virtual agents may be incorporated into game levels and simulated environments. Often, autonomous robots must interact with nearby humans, and virtual agents in a game must interact with agents controlled by a player. Efficient collision avoidance algorithms that can react to a dynamic environment are particularly important in these circumstances since the autonomous robots or virtual agents must adapt their motion to the unpredictable actions of the human. A human or human-controlled agent may also represent the goal location of a group of pursuing autonomous robots or virtual agents. In this case, the group of autonomous robots or virtual agents must be able to converge

on a possibly moving, planar goal region that is the footprint of the human or human-controlled agent.

In previous work [1,2], each agent, such as an autonomous robot or a virtual agent, chooses an avoiding new velocity based on some optimization to make progress toward its goal. Commonly, this optimization is on a *preferred velocity* that is directed to a roadmap node [3] or a fixed point in the center of a navigation mesh edge or face [4]. However, often these points approximate planar goal regions, and this contraction of the goal region to a point can cause artifacts, such as collisions when several agents converge on a single point. Behavior would be improved if the agent could navigate to any point in a goal region. However, with limited exceptions [5], most velocity-based methods are simply coupled with a cluster of point goals and a goal allocation algorithm that chooses the point goal of an agent based on some heuristic. When goal regions are moving, optimizing on a preferred velocity ignores that the position of the goal region may have changed significantly by the time the agent has computed a new velocity. Hence, the trajectory of the agent will not necessarily be directed toward its goal region, and the lengths of paths to the goal region will be increased. If the velocity of the goal region were considered during the optimization of the velocity, then the motion of the agent toward its goal region would again be improved

In this paper, we introduce the *goal velocity obstacle* for navigating multiple agents to planar, spatial goal regions that counters the above-described disadvantages of formulations that optimize on preferred velocity. The basic idea is that instead of only using the notion of velocity obstacles [1] to compute collision-avoiding velocities, we also use them to define the goal regions of the agent within velocity space. We call the velocity obstacle of an agent induced by its goal region a *goal velocity obstacle*, and if the agent chooses a velocity that is inside the goal velocity obstacle at each time step, then it will eventually reach its goal region.

The goal velocity obstacle provides a unified formulation that allows for goals specified as points, line segments, and bounded, planar regions in two dimensions that may be static or moving. An agent navigating using goal velocity obstacles may have multiple goal regions without requiring an explicit goal allocation algorithm that would choose a particular goal region to navigate toward in advance. Goal regions may also have a time dependency, such that the goal region is only available to an agent during a specific window of time, without requiring an explicit scheduling algorithm.

In experiments with hundreds of agents, those navigating using goal velocity obstacles toward static, moving, and multiple goal regions have shorter path lengths from their starting positions to their goal regions and fewer collisions with other agents, than when using velocity-based methods that optimize on a single preferred velocity toward the goal of each agent. The additional computational overhead is just a few microseconds, per agent, per time step, compared to previous velocity-based methods.

The rest of this paper is organized as follows. In Sect. 2, we survey related work on the navigation of multiple agents in video games, mobile robotics, and

simulated environments. We define our problem and give a brief overview of the concepts of velocity obstacles and preferred velocity for navigating multiple agents in Sect. 3. In Sect. 4, we introduce our formulation of goal velocity obstacles, where we use velocity obstacles as spatial goals for agents in velocity space. We describe our implementation and report experimental results in Sect. 5.

2 Related Work

The prevalent approach to navigation in mobile robotics, video games, and simulated environments has been roadmaps [3]. In methods based on roadmaps, agents are constrained to the immediate vicinity of the edges of a graph between intermediate goal nodes placed in the environment. Increasingly, navigation meshes [4, 6, 7] and similar methods [8, 9] are superseding that approach. Randomized methods [10, 11] may be used for roadmap generation, and the Hertel-Mehlhorn algorithm [12] and space-filling volumes [13] allow for automatic navigation mesh generation.

In static environments, derivatives of the A* search algorithm [14] are used on the roadmap or navigation mesh to plan a path to the goal. In dynamic environments, the D* algorithm [15] may be used to repair a previously planned path instead of re-planning from scratch. Planners based on roadmaps have also been adapted to accommodate dynamic environments by reusing previously computed information [16–19] or integrating obstacle movement directly into the planner [20]. Alternatives to changing a pre-computed roadmap or navigation mesh include potential field planners [21], inevitable collision states [22], and adaptive roadmaps [23].

Video games and simulated environments have, historically, used force-based methods, such as flocking [24], in combination with roadmap and navigation mesh approaches to provide local collision avoidance for groups of agents moving through the environment. Velocity-based methods, such as the velocity obstacle [1], and its derivatives [2, 25, 26], popular in mobile robotics, have exhibited improvements in terms of computational performance and local collision avoidance.

Many other methods from mobile robotics, such as the dynamic window approach [27], and rule-based or social-force models from crowd simulation [28–32] are equally suited to navigation of agents in video games and simulated environments. Generally, current collision avoidance approaches are limited to using some form of point goal [33] or line segment goal [5] in connection with the global planner.

3 Background

In this section, we define of the problem of navigating multiple agents in the two-dimensional plane with respect to the two-dimensional velocity space. In addition, we briefly review the concepts of velocity obstacles and preferred velocity to provide context for our formulation of goal velocity obstacles.

3.1 Problem Definition

Let there be a set of $n > 1$ disc-bounded holonomic agents $\mathcal{A}$ sharing an environment in the two-dimensional workspace $\mathbb{R}^2$ and the two-dimensional velocity space, which we denote $\mathbb{V}^2$. The environment may also contain static and moving obstacles. We assume that static and moving obstacles can be identified by each agent as not actively adapting their velocity to avoid any of the agents.

Each agent $A \in \mathcal{A}$ has a constant radius $r_A > 0$, a current position $\boldsymbol{p}_A \in \mathbb{R}^2$, and a current velocity $\boldsymbol{v}_A \in \mathbb{V}^2$ for each time step, all of which are known to every agent. Furthermore, let each agent A also have one or more bounded *goal regions* $G \subset \mathbb{R}^2$ that are not necessarily known to the other agents. Each goal region may be of any shape, need not simply be a point, and may have a nonzero linear velocity $\boldsymbol{v}_G \in \mathbb{V}^2$. For simplicity, we assume that goal regions do not rotate, i.e., their angular velocity is zero

The objective of each agent A is to choose, independently and simultaneously, a new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ at each time step to compute a trajectory toward any point in a goal region G without collisions with other agents, static obstacles, or moving obstacles. The agents should not perform any sort of central coordination, but we may assume that all agents are using the same strategy to choose new velocities. An agent A has reached a goal region G if $A \cap G \neq \emptyset$.

3.2 Velocity Obstacles

The *velocity obstacle* [1] of an agent A induced by an agent or moving obstacle B is the set of all velocities of A that may cause a collision with agent or moving obstacle B within the short window of time $[0, \tau]$, assuming that each agent or moving obstacle continues on the same trajectory (see Fig. 1). More formally, let

$$A \oplus B = \{\boldsymbol{a} + \boldsymbol{b} \mid \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$

be the Minkowski sum of agent A and agent or moving obstacle B, and let

$$-A = \{-\boldsymbol{a} \mid \boldsymbol{a} \in A\}$$

denote the agent A reflected in its reference point. Then the velocity obstacle for agent A induced by agent or moving obstacle B, denoted $VO^{\tau}_{A|B} \subset \mathbb{V}^2$, is defined by

$$VO^{\tau}_{A|B} = \{\boldsymbol{v} \mid \exists s \in [0, \tau] :: s\,(\boldsymbol{v} - \boldsymbol{v}_B) \in B \oplus -A\} \quad .$$

There are many variations of velocity obstacles, however, for simplicity, we will use this definition, unless noted otherwise.

When there are more than two agents in the environment, we define the *combined velocity obstacle* of an agent A, denoted $VO^{\tau}_{A} \subset \mathbb{V}^2$, as the union of all velocity obstacles of the agent induced by other agents $A_1, \ldots, A_{n-1}$ in the environment, i.e.,

$$VO^{\tau}_{A} = VO^{\tau}_{A|A_1} \cup \cdots \cup VO^{\tau}_{A|A_{n-1}} \quad .$$

If an agent chooses a velocity outside its combined velocity obstacle at each time step, then it is guaranteed not to collide with any of the other agents in the environment during the window of time $[0, \tau]$.

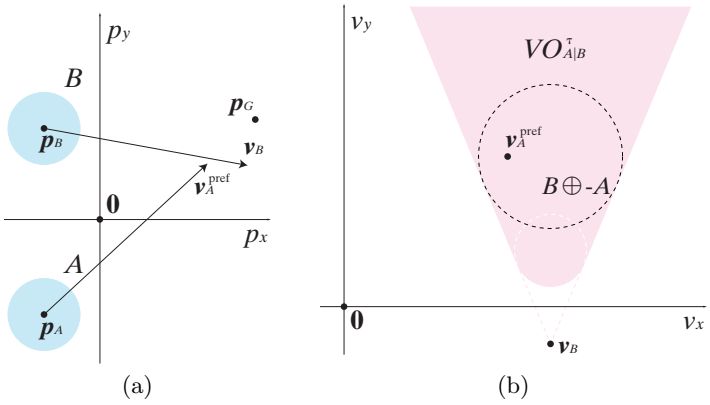

Fig. 1. (a) Agent A navigating toward a static point goal $\boldsymbol{p}_G$ with preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ while avoiding agent B. (b) The velocity obstacle $VO^{\tau}_{A|B}$ for agent A induced by agent B.

3.3 Preferred Velocities

The *preferred velocity* of an agent is, informally, the velocity that the agent would take were there no other agents, static obstacles, or moving obstacles for it to avoid (see Fig. 1). Typically, the preferred velocity $\boldsymbol{v}_A^{\text{pref}} \in \mathbb{V}^2$ of an agent A would be directed toward a fixed point goal $\boldsymbol{p}_G \in \mathbb{R}^2$ and have a magnitude $v_A^{\text{pref}} > 0$, known as the preferred speed, i.e.,

$$\boldsymbol{v}_A^{\text{pref}} = v_A^{\text{pref}} \frac{\boldsymbol{p}_A - \boldsymbol{p}_G}{\|\boldsymbol{p}_A - \boldsymbol{p}_G\|_2}$$

It follows that in order to progress toward its point goal without collisions, an agent A will choose, at each time step, the new velocity $\boldsymbol{v}_A^{\text{new}}$ closest to its preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ that lies outside the combined velocity obstacle, i.e.,

$$\boldsymbol{v}_A^{\text{new}} = \underset{\boldsymbol{v} \notin VO^{\tau}_A}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}_A^{\text{pref}} \right\|_2 \quad .$$

4 Goal Velocity Obstacles

In this section, we introduce the new concept of using velocity obstacles to specify the goal regions of agents in the two-dimensional velocity space from which to choose collision-free velocities toward their goal regions in the two-dimensional workspace.

4.1 Overall Approach

Instead of using velocity obstacles purely for excluding velocities that may cause collisions with other agents or moving obstacles, then optimizing with respect to a preferred velocity for navigation to a goal in the workspace $\mathbb{R}^2$, we propose the additional use of velocity obstacles to define the goal regions of an agent within the velocity space $\mathbb{V}^2$. More precisely, we define the *goal velocity obstacle* of agent A toward the goal region G, denoted $GVO_{A|G} \subset \mathbb{V}^2$, as

$$GVO_{A|G} = VO^{\tau}_{A|G} = \{\boldsymbol{v} \mid \exists s \in [0, \tau] :: s\,(\boldsymbol{v} - \boldsymbol{v}_B) \in G \oplus -A\} \quad .$$

We then choose a new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ of agent A such that $\boldsymbol{v}_A^{\text{new}}$ lies not only *outside* the velocity obstacles induced by other agents, but also *inside* the goal velocity obstacle toward the goal region G, i.e., $\boldsymbol{v}_A^{\text{new}} \in GVO_{A|G} \setminus VO^{\tau}_A$. This is illustrated in Fig. 2.

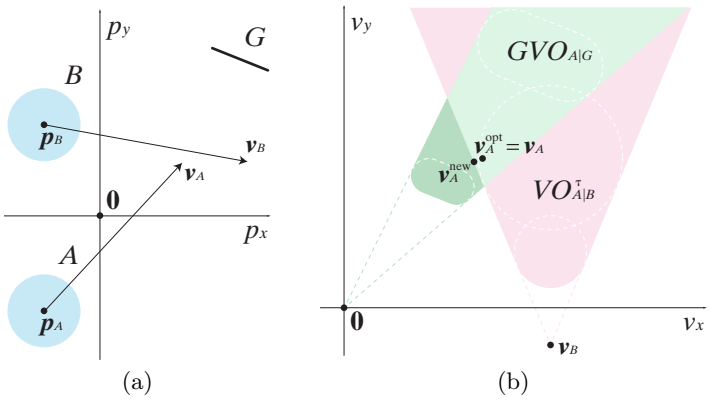

Fig. 2. (a) Agent A navigating toward a static line segment goal region G while avoiding agent B. (b) The goal velocity obstacle $GVO_{A|G}$ for agent A toward goal region G and the velocity obstacle $VO^{\tau}_{A|B}$ for agent A induced by agent B.

4.2 Choice of Velocities

In general, there will be a choice of collision-free velocities $\boldsymbol{v}_A^{\text{new}}$ that will navigate the agent A to some point in its goal region. Assuming that there is no preference as to which point in the goal region an agent A ultimately reaches, we choose a velocity $\boldsymbol{v}_A^{\text{opt}} \in \mathbb{V}^2$, which we call the *optimization velocity*, with respect to which we must optimize from those velocities that are collision-free and inside the goal velocity obstacle, i.e.,

$$\boldsymbol{v}_A^{\text{new}} = \underset{\boldsymbol{v} \in GVO_{A|G} \setminus VO^{\tau}_A}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}_A^{\text{opt}} \right\|_2 \quad .$$

Motivated by a desire for agents to make as minimal change in velocity as possible at each time step [32], we choose the optimization velocity $\boldsymbol{v}_A^{\text{opt}}$ of an agent A as follows.

1. If the current velocity $\boldsymbol{v}_A$ is *inside* the goal velocity obstacle $GVO_{A|G}$, we choose the current velocity as the optimization velocity, whether or not that velocity is collision-free, i.e.,

$$\boldsymbol{v}_A^{\text{opt}} = \boldsymbol{v}_A \ .$$

2. If the current velocity is *outside* the goal velocity obstacle, so the agent is moving away from its goal region, we choose the closest velocity to the current velocity $\boldsymbol{v}_A$, with respect to Euclidean distance in the velocity space $\mathbb{V}^2$, that lies inside the goal velocity obstacle, i.e.,

$$\boldsymbol{v}_A^{\text{opt}} = \underset{\boldsymbol{v} \in GVO_{A|G}}{\arg\min} \|\boldsymbol{v} - \boldsymbol{v}_A\|_2 \ .$$

The optimization velocity $\boldsymbol{v}_A^{\text{opt}}$ is distinct from the notion of preferred velocity $\boldsymbol{v}_A^{\text{pref}}$, and, in general, much less influences the path taken by the agent A.

4.3 High Densities of Agents

By definition, if an agent chooses a velocity inside the goal velocity obstacle at every time step, it will reach its goal region at some future moment in time, assuming that such a velocity exists. If, however, due to a high density of agents, there is no such velocity, i.e.,

$$GVO_{A|G} \subset VO_A^\tau = \bigcup_{\substack{B \in \mathcal{A} \\ A \neq B}} VO_{A|B}^\tau \ ,$$

this means that either the goal region G is moving away from the agent A at a faster speed than the agent can attain, or else the path to the goal region is blocked by other agents or moving obstacles. In the first case, it is not possible for the agent to reach its goal region. However, in the second case, we choose a new velocity by relaxing some of the constraints in velocity space. In relaxing constraints, we must balance the possibly conflicting objectives of avoiding collisions and reaching the goal region.

1. If the priority is to avoid collisions, above all else, we can replace the goal velocity obstacle $GVO_{A|G}$ with the whole of the velocity space $\mathbb{V}^2$ at each time step until a velocity within the goal velocity obstacle becomes available, i.e.,

$$\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2 \setminus VO_A^\tau \neq \emptyset \ .$$

2. If an increased possibility of collision is allowable, we remove, in turn, the velocity obstacle induced by the most distant agent, with respect to Euclidean

distance in the workspace $\mathbb{R}^2$, until a velocity within the goal velocity obstacle becomes free, i.e.,

$$\boldsymbol{v}_A^{\text{new}} \in GVO_{A|G} \setminus \left(VO^{\tau}_{A_1} \cup \cdots \cup VO^{\tau}_{A_m} \right) \neq \emptyset \ ,$$

where

$$\|\boldsymbol{p}_A - \boldsymbol{p}_{A_i}\|_2 \leq \left\|\boldsymbol{p}_A - \boldsymbol{p}_{A_{i+1}}\right\|_2$$

and

$$GVO_{A|G} \subset VO^{\tau}_{A_1} \cup \cdots \cup VO^{\tau}_{A_{m+1}} \ ,$$

for $m < n$.

In our experiments, we favored the second of these options.

4.4 Moving Goal Regions

Consider now a goal region moving in a near-perpendicular direction to an agent (see Fig. 3). When navigating using preferred velocities, the future trajectory $\boldsymbol{p}_G + s\boldsymbol{v}_G$ of the goal region G is not taken into account, only the instantaneous position of its center $\boldsymbol{p}_G$. Therefore, the agent is always changing its preferred velocity, and, hence, current velocity, at each time step, navigating toward the position of the goal region at the previous time step. This will occur even though it is apparent that the goal region will cross the path of the agent if the agent continues on its original trajectory. Contrast this with the goal velocity obstacle, which, by definition, considers the future trajectory of the goal region, coupled with an optimization velocity that favors minimal changes in current velocity.

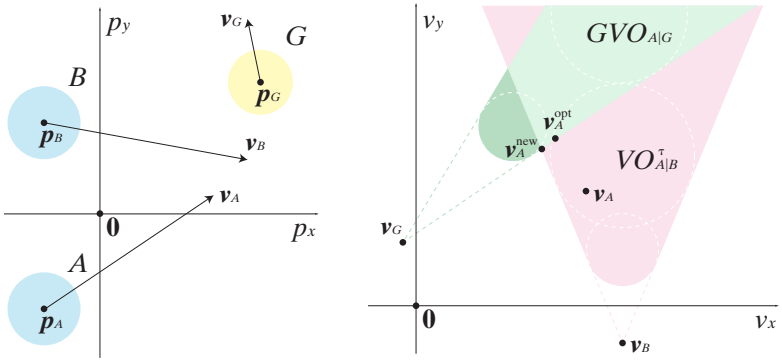

Fig. 3. (a) Agent A navigating toward a moving disc-shaped goal region G with velocity $\boldsymbol{v}_G$ while avoiding agent B. (b) The goal velocity obstacle $GVO_{A|G}$ for agent A toward goal region G and the velocity obstacle $VO^{\tau}_{A|B}$ for agent A induced by agent B.

4.5 Multiple Goal Regions

When navigating using preferred velocities, if a goal region consists of the union of all elements of a set $\mathcal{G}$ of multiple goal sub-regions (see Fig. 4), agent A would be required to choose in advance one goal sub-region $G \in \mathcal{G}$ toward which to navigate explicitly and set $\boldsymbol{v}_A^{\text{pref}}$ in the direction of $\boldsymbol{p}_G$. However, with the formulation of goal velocity obstacles, we can construct goal velocity obstacles of each individual goal sub-region and then define the goal velocity obstacle of the entire goal region as the union of these individual goal velocity obstacles, i.e.,

$$GVO_{A|\mathcal{G}} = \bigcup_{G \in \mathcal{G}} GVO_{A|G} \ .$$

This has the advantage that if the path to one of the goal sub-regions G is blocked by other agents, then the navigating agent will automatically divert to another goal sub-region since the goal velocity obstacle $GVO_{A|G}$ corresponding to the blocked goal sub-region will be completely covered with velocity obstacles, i.e., $GVO_{A|G} \subset VO_A^\tau$.

We choose the optimization velocity to be such that it lies inside the goal velocity obstacle of the goal sub-region toward which the agent moved at the previous time step, which reduces the possibility that the velocity of the agent will oscillate between goal velocity obstacles of different goal sub-regions. While the optimization velocity is chosen relative to a particular goal velocity obstacle, this does not preclude a velocity being chosen if that goal sub-region is later found to be blocked, and no explicit changes to the formulation are required to accommodate this situation.

4.6 Goal Regions with Time Windows

Suppose now that a goal region is only available to an agent during some time window $[\tau_1, \tau_2]$ relative to the current time. We can express this time dependency by truncating the goal velocity obstacle both at the apex and toward the base (see Fig. 5). For this doubly truncated goal velocity obstacle, the values of τ_1 and τ_2 decrease at each successive time step, and the definition of the goal velocity obstacle $GVO_{A|G}^{\tau_1,\tau_2}$ of agent A toward the goal region G with time window $[\tau_1, \tau_2]$ becomes

$$GVO_{A|G}^{\tau_1,\tau_2} = \{\boldsymbol{v} \mid \exists s \in [\tau_1, \tau_2] :: s\,(\boldsymbol{v} - \boldsymbol{v}_B) \in G \oplus -A\} \ .$$

5 Experimentation

In this section, we describe the implementation of our approach and discuss the results of our experiments involving multiple agents.

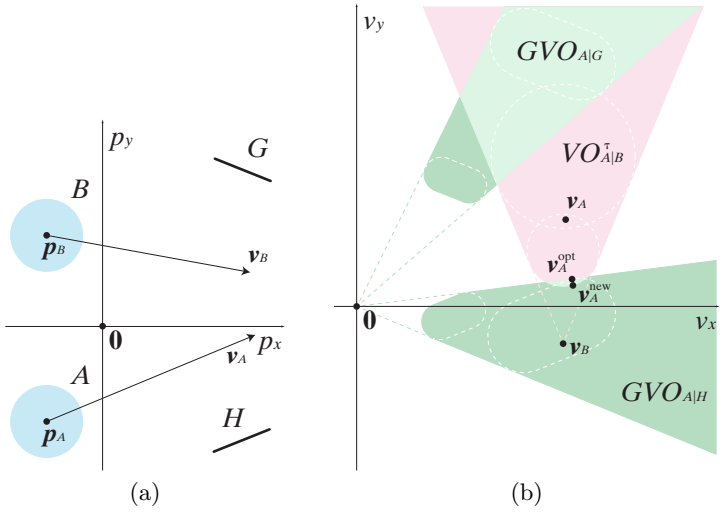

Fig. 4. (a) Agent A navigating toward a choice of two static line segment goal sub-regions G and H while avoiding agent B. (b) The goal velocity obstacles $GVO_{A|G}$ for agent A toward goal sub-region G and $GVO_{A|H}$ for agent A toward goal sub-region H, and the velocity obstacle $VO^{\tau}_{A|B}$ for agent A induced by agent B.

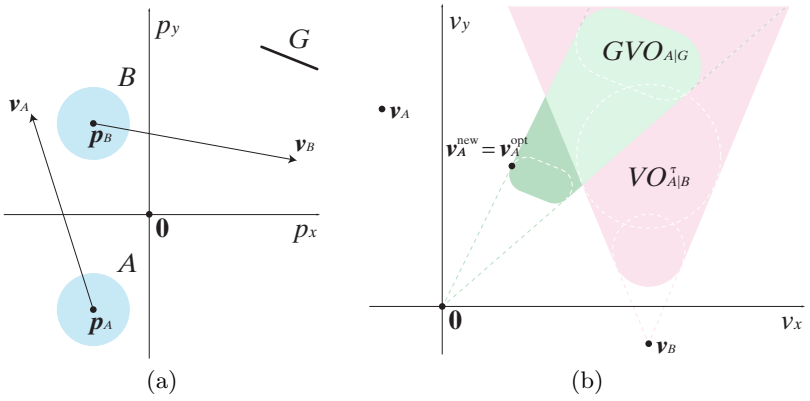

Fig. 5. (a) Agent A navigating toward a static line segment goal region G with limited time window $[\tau_1, \tau_2]$ while avoiding agent B. (b) The goal velocity obstacle $GVO_{A|G}$ for agent A toward goal region G with limited time window $[\tau_1, \tau_2]$ and the velocity obstacle $VO^{\tau}_{A|B}$ for agent A induced by agent B.

5.1 Implementation

We implemented our approach in C++ using hybrid reciprocal velocity obstacles [26] for collision avoidance between pairs of agents. Our algorithm to choose the new velocity of each agent at every time step was based on the ClearPath efficient geometric algorithm [34]. Calculations for each agent were carried out in separate and independent threads, and in parallel, where possible, using Intel Threading Building Blocks 4.1. The code was compiled using the Intel C++ Compiler XE 13.

For efficiency reasons, only a subset of all other agents within a fixed radius of each agent, with respect to Euclidean distance in the two-dimensional plane, were considered for collision avoidance, and these agents were selected at the beginning of every time step using an algorithm based on k-D trees [35].

5.2 Experiments

We applied our approach to multiple challenging experiments containing 25–200 agents as follows.

1. The agents are positioned evenly along one side of a rectangular environment in two dimensions. The agents must navigate to one static line segment goal region located midway along the opposite side of the environment to the starting positions of the agents, see Fig. 6.
2. The agents must navigate across the environment toward one moving line segment goal region. The goal region moves at a constant velocity along the opposite side of the environment to the starting positions of the agents, perpendicular to the direct paths of the agents to the goal region, see Fig. 7.
3. The agents must navigate across the environment toward two static line segment goal regions located at each end of the opposite side of the environment to the starting positions of the agents, see Fig. 8.

Each experiment was performed twice, first using goal velocity obstacles and hybrid reciprocal velocity obstacles, and then using preferred velocities and hybrid reciprocal velocity obstacles. Each agent had a radius of 1 m, an initial or preferred speed of 1.4 m/s, and a maximum speed of 2.5 m/s.

Tables 1–3 list the total number of collisions between agents for the entirety of each experiment, the average path length from start position to goal region for each agent in each experiment, and the average computation time for each time step. Timings are for one core of a quad-core 2.8 GHz Intel Core i5 processor within a standard desktop system containing 8 GB of memory and running OS X 10.8 Mountain Lion.

5.3 Discussion

Figures 6–8 show that, for 25 agents, almost the entire goal region is utilized by the agents in each experiment using goal velocity obstacles. In Fig. 7, the agents take into account the motion of the goal region, and, in particular, agents

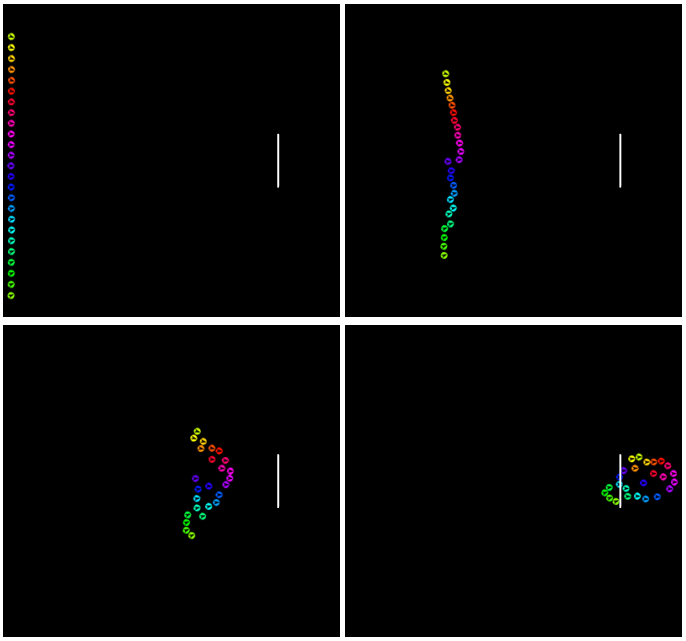

Fig. 6. 25 agents navigating toward one static line segment goal region using goal velocity obstacles.

farthest from the starting position of the goal region maintain a direct path to intercept the goal region as it passes close by, later in the experiment. Figure 8 demonstrates the agents splitting into two groups to move toward the closest goal region to their starting position.

It is clear that in all experiments, for 25–200 agents, the number of collisions between agents is significantly less when using goal velocity obstacles rather than preferred velocities, see Table 1. Specifically, there are at least 55% fewer collisions for the experiments with one static goal region, at least 97% fewer collisions for the experiments with one moving goal region, and at least 90% fewer collisions for the experiments with two static goal regions.

The length of the path that each agent takes to a goal region is also less when using goal velocity obstacles, see Table 2. For the experiments with one and two static goal regions, the paths are at least 5% shorter, and for the experiment with one moving goal region the paths are always at least 10% shorter and more than 25% shorter in most cases.

Computationally, it takes 2–21% longer, each time step, to compute new velocities using goal velocity obstacles rather than preferred velocities, see Table 3. Mostly, the difference is less than 10%, however, and, overall, using goal velocity obstacles adds only a few microseconds of computation per agent at each time step.

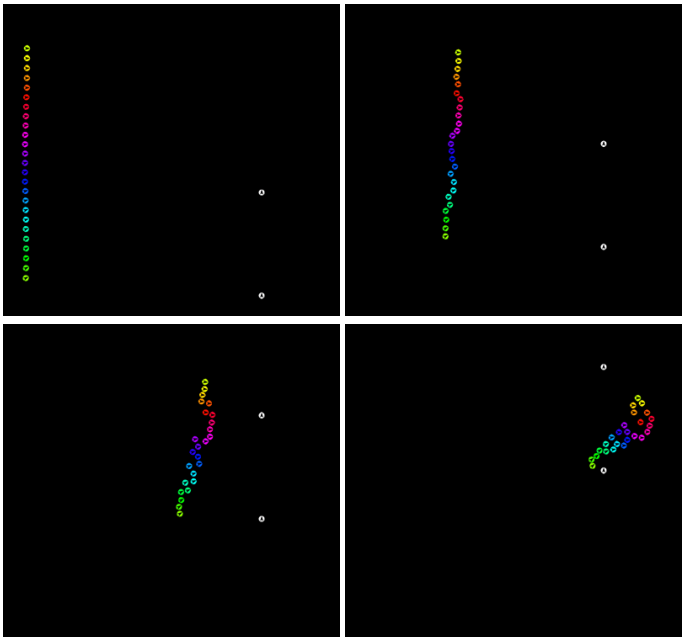

Fig. 7. 25 agents navigating toward one moving line segment goal region using goal velocity obstacles.

Table 1. Total number of collisions between agents for the entirety of each experiment for experiments using goal velocity obstacles (GVO) and preferred velocities (PV).

	Number of agents	Total number of collisions	
		GVO	PV
1. One static goal region	25	0	56
	50	10	62
	100	24	80
	200	46	106
2. One moving goal region	25	4	144
	50	2	132
	100	2	166
	200	0	218
3. Two static goal regions	25	4	348
	50	10	498
	100	28	778
	200	94	996

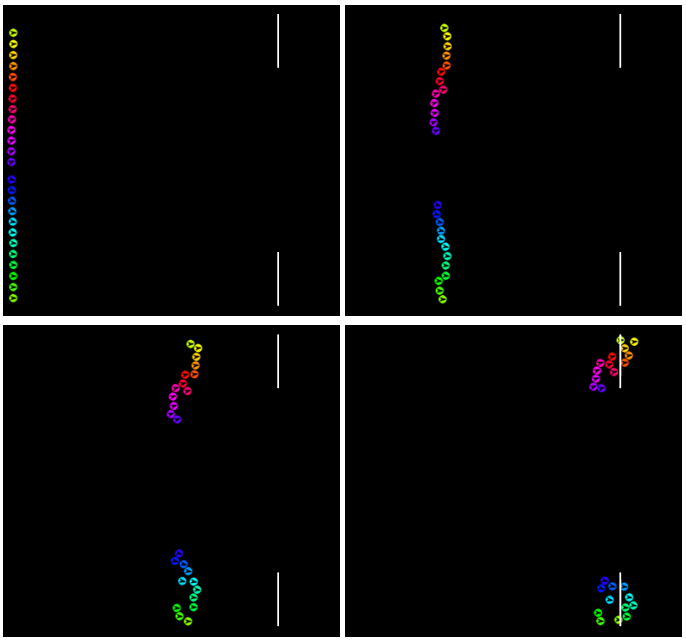

Fig. 8. 25 agents navigating toward two static line segment goal regions using goal velocity obstacles.

Table 2. Average path length from start position to goal region for each agent for experiments using goal velocity obstacles (GVO) and preferred velocities (PV).

	Number of agents	Average path length to goal region (m)	
		GVO	PV
1. One static goal region	25	229.0	243.3
	50	245.5	266.2
	100	307.6	339.8
	200	404.2	441.2
2. One moving goal region	25	239.4	323.3
	50	240.5	340.9
	100	301.3	406.9
	200	405.1	454.6
3. Two static goal regions	25	205.7	217.4
	50	227.0	241.3
	100	295.8	314.8
	200	406.9	427.8

Table 3. Average computation time for each time step for experiments using goal velocity obstacles (GVO) and preferred velocities (PV).

	Number of agents	Average computation time (ms)	
		GVO	PV
1. One static goal region	25	0.58	0.51
	50	1.53	1.46
	100	3.26	3.08
	200	6.59	6.44
2. One moving goal region	25	2.90	2.41
	50	4.87	4.78
	100	7.33	6.86
	200	9.68	9.05
3. Two static goal regions	25	0.36	0.32
	50	1.12	1.04
	100	2.26	1.97
	200	4.12	4.01

6 Conclusion

In this paper, we have presented the *goal velocity obstacle* for the spatial navigation of multiple agents to arbitrary-shaped, planar goal regions in the two-dimensional plane. Our approach uses velocity obstacles not only to compute velocities that may cause collisions with other agents, but also to define the goal velocity obstacle, which specifies velocities in the two-dimensional velocity space that will direct an agent toward its goal region on the plane.

Our goal velocity obstacle formulation is general, allowing for planar goal regions of any shape without the need to approximate the goal region as a point or line segment, as is required by most previous collision avoidance methods. Goal regions may be static or they may be moving with a nonzero linear velocity. We may specify multiple goal regions for each agent without requiring an explicit goal allocation algorithm to choose a particular goal region for each agent in advance of each time step. Our approach also allows for goal regions that are available for a limited time window.

We have applied our approach to multiple challenging experiments by integrating with the hybrid reciprocal velocity obstacle formulation for collision avoidance. On average, the agents traverse shorter path lengths and have fewer collisions than when simply using preferred velocities directed to a single point in their goal region instead of goal velocity obstacles.

Acknowledgments This work was supported in part by Army Research Office Contract W911NF-04-1-0088, by National Science Foundation Awards 1000579 and 1117127, and by Intel Corporation.

References

1. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal on Robotics Research **17** (1998) 760–772
2. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2008) 1928–1935
3. Canny, J.F.: The Complexity of Robot Motion Planning. The MIT Press, Cambridge, Mass. (1988)
4. Snook, G.: Simplified 3D movement and pathfinding using navigation meshes. In DeLoura, M., ed.: Game Programming Gems. Charles River, Hingham, Mass. (2000) 288–304
5. Curtis, S., Snape, J., Manocha, D.: Way portals: efficient multi-agent navigation with line-segment goals. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. (2012) 15–22
6. Kallmann, M.: Shortest paths with arbitrary clearance from navigation meshes. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2010) 159–168
7. Van Toll, W., Cook, IV, A., Geraerts, R.: Navigation meshes for realistic multi-layered environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2011) 3526–3532
8. Pettré, J., Laumond, J.P., Thalmann, D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. In: Proceedings of the First International Workshop on Crowd Simulation. (2005) 81–89
9. Geraerts, R., Kamphuis, A., Karamouzas, I., Overmars, M.: Using the corridor map method for path planning for a large number of characters. In: Proceedings of the First International Workshop on Motion in Games. (2008) 11–22
10. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12** (1996) 566–580
11. LaValle, S.M., Kuffner, Jr., J.J.: Randomized kinodynamic planning. The International Journal on Robotics Research **20** (2001) 378–400
12. Hertel, S., Mehlhorn, K.: Fast triangulation of the plane with respect to simple polygons. Information and Control **64** (1985) 52–76
13. Tozour, P.: Search space representations. In Rabin, S., ed.: AI Game Programming Wisdom 2. Charles River, Hingham, Mass. (2003) 85–102
14. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4** (1968) 100–107
15. Stentz, A.: The focussed D* algorithm for real-time replanning. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. (1995) 1652–1659
16. Jaillet, L., Simeon, T.: A PRM-based motion planner for dynamically changing environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2004) 1606–1611
17. Kallmann, M., Mataric, M.: Motion planning using dynamic roadmaps. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2004) 4399–4404
18. Ferguson, D., Kalra, N., Stentz, A.: Replanning with RRTs. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2006) 1243–1248

19. Zucker, M., Kuffner, J., Branicky, M.: Multipartite RRTs for rapid replanning in dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2007) 1603–1609
20. Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. The International Journal on Robotics Research **21** (2002) 233–255
21. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal on Robotics Research **5** (1986) 90–98
22. Petti, S., Fraichard, T.: Safe motion planning in dynamic environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2005) 2210–2215
23. Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., Manocha, D.: Real-time navigation of independent agents using adaptive roadmaps. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. (2007) 99–106
24. Reynolds, C.: Flocks, herds and schools: a distributed behavioral model. ACM SIGGRAPH Computer Graphics **21** (1987) 25–34
25. Van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Proceedings of the Fourteenth International Symposium on Robotics Research. (2011) 3–19
26. Snape, J., Van den Berg, J., Guy, S.J., Manocha, D.: The hybrid reciprocal velocity obstacle. IEEE Transactions on Robotics **27** (2011) 696–706
27. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine **4** (1997) 23–33
28. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E **51** (1995) 4282–4286
29. Kluge, B., Prassler, E.: Reflective navigation: individual behaviors and group behaviors. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2004) 4172–4177
30. Karamouzas, I., Overmars, M.: Simulating the local behaviour of small pedestrian groups. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. (2010) 183–190
31. Van Welbergen, H., Van Basten, B., Egges, A., Ruttkay, Z., Overmars, M.: Real time animation of virtual humans: a trade-off between naturalness and control. Computer Graphics Forum **29** (2010) 2530–2554
32. Guy, S.J., Curtis, S., Lin, M.C., Manocha, D.: Least-effort trajectories lead to emergent crowd behaviors. Physical Review E **85, 016110** (2012)
33. Van den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. (2008) 139–147
34. Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P.: ClearPath: highly parallel collision avoidance for multi-agent simulation. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2009) 177–187
35. De Berg, M., Cheong, O., Van Kreveld, M., Overmars, M.: Computational Geometry. third edn. Springer, Heidelberg, Germany (2008)