Auton Robot (2012) 32:227–242
DOI 10.1007/s10514-011-9266-8

Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles

Albert Wu · Jonathan P. How

Received: 31 January 2011 / Accepted: 28 November 2011 / Published online: 8 February 2012

Abstract This paper presents a new approach to guaranteeing collision avoidance with respect to moving obstacles that have constrained dynamics but move unpredictably. Velocity Obstacles have been used previously to plan trajectories that avoid collisions with obstacles under the assumption that the trajectories of the objects are either known or can be accurately predicted ahead of time. However, for real systems this predicted trajectory will typically only be accurate over short time-horizons. To achieve safety over longer time periods, this paper instead considers the set of all reachable points by an obstacle assuming that the dynamics fit the unicycle model, which has known constant forward speed and a maximum turn rate (sometimes called the Dubins car model). This paper extends the Velocity Obstacle formulation by using reachability sets in place of a single "known" trajectory to find matching constraints in velocity space, called *Velocity Obstacle Sets*. The Velocity Obstacle Set for each obstacle is equivalent to the union of all velocity obstacles corresponding to any dynamically feasible future trajectory, given the obstacle's current state. This region remains bounded as the time horizon is increased to infinity, and by choosing control inputs that lie outside of these Velocity Obstacle Sets, it is guaranteed that the host agent can always actively avoid collisions with the obstacles, even without knowing their exact future trajectories. Furthermore it is proven that, subject to certain initial conditions, an iterative planner under these constraints guarantees safety for all time. Such an iterative planner is implemented and demonstrated in simulation.

Keywords Mobile robot · Guaranteed safety · Mobile obstacle avoidance · Unpredictable obstacles · Velocity-obstacle · Iterative motion planning · Moving obstacles · Unicycle dynamics · Dubins cars · Partial motion planning

A. Wu (✉) · J.P. How
Department of Aeronautics and Astronautics, MIT, Cambridge, MA, USA
e-mail: albertwu87@gmail.com

J.P. How
e-mail: jhow@mit.edu

1 Introduction

Dynamic obstacle avoidance is an important, ubiquitous, and often challenging problem for autonomous mobile robots. The characteristics of specific scenarios call for different sets of assumptions and different collision avoidance algorithms. In cases with long time-scales and significant uncertainty about the future, accurately generating an entire plan at once may not be feasible. Instead, partial motion planners (Petti and Fraichard 2005; Frazzoli et al. 2002; Van Den Berg et al. 2006) can be used to create or modify the trajectory as new information becomes available. Often, it is impossible to guarantee being able to find a collision-free trajectory to the goal. Instead, such planners iteratively extend the trajectory along the most promising segments, while ideally maintaining safety with respect to the obstacles in the environment.

For planners that use a look-ahead time horizon of some finite duration τ (Frazzoli et al. 2002), collision avoidance is ensured for one τ-interval at a time. However, τ-safety is difficult to propagate forward between planning iterations, so sustained, long-term safety cannot be guaranteed. Instead, safety guarantees for such planners are often achieved by defining safe states and restricting the planner to only construct trajectories composed of such states (Schouwenaars et al. 2004; Kuwata et al. 2008); or equivalently, defining inevitable collision states (Petti and Fraichard 2005; Fraichard and Asama 2004; Bekris 2010) and avoiding them. Safe states are those that do not violate the imposed collision constraints and can transition into other safe states, while inevitable collision states are those from which there are no safe transitions. For example, in structured environments in which the other vehicles practice reasonable collision avoidance, coming to a complete stop and staying stationary can be considered reaching an invariant safe state (which can be propagated indefinitely), so any state that can make this transition would be deemed safe (Kuwata et al. 2008).

However, if the dynamic obstacles exhibit *unpredictable behavior*, as is often the case in many real-world environments, it becomes much more difficult to define sufficient conditions that allow safe states to be propagated forward in time. The host vehicle may be struck by other vehicles if it stays at rest, or it may become surrounded by multiple other vehicles such that collision becomes inevitable. Likewise, this kind of potential interaction between multiple unpredictable obstacles makes it difficult to exhaustively solve for and concisely represent all of the inevitable collision states.

Guaranteed infinite horizon collision avoidance with respect to unpredictable dynamic obstacles is therefore a challenging problem. The nature of guaranteed safety necessarily deals with worst-case conditions, and thus some restrictive conditions and assumptions must first be met for the problem to be at all feasible. Firstly, Friachard and Bouraine (2011) shows that the host robot must have an unlimited sensing horizon. Furthermore, L'Esperance et al. (2011) shows that with enough static obstacles in the environment, it may often be impossible to avoid collisions; so in this work the environment is assumed to be unbounded, i.e., the host robot is outdoors. In addition, as shown in Sect. 4.3, the host robot must be able to travel faster than the dynamic obstacles. This paper shows that, under these assumptions, it is possible to derive necessary conditions that can be used in a partial motion planner to guarantee infinite horizon collision avoidance. Many real-world scenarios may not directly fit into this framework, but even so, the method presented here represents a crucial extension of the process for guaranteeing safety using partial motion planning.

1.1 Contributions

This paper presents a method for finding velocity-space constraints for a motion planner that guarantees infinite horizon safety of a host disc robot in an unbounded environment with multiple disc obstacles that have unicycle dynamics, but can move unpredictably. The obstacles' current poses are observed, but no explicit predictions about their future trajectories are made. The safety guarantee is achieved by combining the obstacles' reachable sets (as a functions of time) subject to their dynamics (Cockayne and Hall 1975) with the velocity obstacle concept (Fiorini and Shiller 1998). Most significantly, by using reachable sets instead of predicted trajectories, this work extends the existing velocity obstacle literature to be applicable in an infinite horizon setting. As discussed in Sect. 5, while the algorithm presented here finds trajectories that are provably safe, it is not *complete*.

Fraichard (2007) state that a motion planner should (i) account for the host robot's dynamics, (ii) account for the obstacles' future behavior, and (iii) consider an appropriate time horizon. The algorithm in this paper deals directly with conditions (ii) and (iii): (ii) is addressed by solving for the obstacles' reachable sets given their current poses and the assumed dynamical model, and (iii) is addressed by considering the infinite horizon such that rigorous guarantees can be made.

Meanwhile, condition (i) can be dealt with *independently* of the results of the algorithm presented here; with the sole assumption that it is dynamically feasible for the host robot to continue in the same direction without any acceleration. Depending on the specific application, various existing methods can be used to generate potential trajectories that are consistent with the host robot's full dynamics model, and thus this step is not explored in this work. Sections 4.4, 4.5, and 5 discuss in detail how the results of this work are to be applied to check the infinite horizon safety of an arbitrary proposed trajectory. It is shown that these results could be directly used to generate single-velocity infinite trajectories that satisfy the necessary conditions for guaranteed safety, and this method is demonstrated in simulation in Sect. 6. This approach could be generalized for future partial motion planners by using other methods to generate short-term trajectories for as long as the obstacles are predictable and then apply the derived conditions to ensure long-term safety.

The obstacles in this work are assumed to be discs that obey unicycle (or Dubins car) dynamics; *i.e.,* they move with some fixed forward speed v and can turn at any turn-rate less than some maximum value ω. Guaranteed safety is derived as a function of just these two parameters and the initial locations and orientations of the obstacles. While this model alone does not generalize to all dynamical obstacle behaviors that could be of interest, it is commonly used in UAV and robot literature (see for example Vendittelli et al. 1999; Chitsaz and LaValle 2007) and serves as a reasonable initial base case. The model can be immediately used to describe obstacles that are limited only by a maximum speed by assigning them unlimited turn rate (Fig. 8) as well as static disc obstacles by assigning them 0 speed. The *velocity obstacle sets* (Sect. 4) for arbitrarily shaped static obstacles are also trivial to generate from geometry. For dynamic obstacles described by other dynamics (where at least the velocity is upper-bounded), it may possible to re-trace the steps in the ensuing sections to derive algorithms that are similar to the one presented here.

1.2 Literature review

Conditions for infinite horizon safety in the presence of multiple unpredictable obstacles have not been directly addressed in the existing literature. Using the original formulation of the velocity obstacle (Fiorini and Shiller 1998), one can find single velocity trajectories that are guaranteed collision-free, given the exact trajectory of the obstacles for some time-scale (see Sect. 4). This time-scale could be infinite, but that would unrealistically require that all obstacle trajectories be known perfectly for all future time.

Instead of assuming a known infinite trajectory, in Large et al. (2005), the motion of the obstacles are predicted for a finite duration with an associated uncertainty. Velocity obstacles are computed after growing the obstacles by the uncertainty, and then used to plan safe finite segments. Since the time-scale is finite, there is no guarantee that a trajectory can be continued safely when a re-plan is necessary. Further, as stated in Large et al. (2005), growing obstacles by the uncertainty often results in all reachable velocities creating potential collisions, and safety is not ensured.

Gal et al. (2009) appeal to a safe state argument. They use the dynamics of the host robot and of each obstacle to calculate an optimal time horizon, such that *inevitable collision states* are avoided by heeding the bounds of the corresponding finite-time velocity obstacle. Firstly, this method still requires the trajectories of the dynamic obstacles be known (at least up to some finite horizon), and this information may not always be available. Further, it is argued that "any velocity that does not penetrate this [finite, optimal time horizon] velocity obstacle should allow sufficient time under the given control authority to avoid collision." But this does not account for the interaction of the constraints imposed by multiple obstacles. The extremal trajectories that are computed to avoid collisions with an obstacle may interfere with neighboring obstacles, so a velocity that respects the optimal horizon velocity obstacle of one object may in fact be forced to collide with a different obstacle.

In the field of differential games, this type of safety problem is well explored as a "pursuit-evasion" game (Rzymowski 1986; Borowko et al. 1988; Chodun 1989; Sinitsyn 1993; Pashkov and Sinitsyn 1995) in which control inputs for the pursuers and the evader that respectively minimize or maximize the time until capture are found and executed. These formulations typically do not allow collision avoidance to be treated as a constraint to be observed while the "objective function" of the planner is still goal-seeking. One approach that does consider collision avoidance is Tomlin et al. (1998), which finds a set of vehicle states that form the *least restrictive control scheme* for any desired safety margin. Tomlin et al. (1998) shows that outside of this set, any control input can be safe as long as specific controls that maintain the margin are executed on the boundary of the set. However, in general, solving the resulting Hamilton-Jacobi equations explicitly to find this set and the associated controls can be very difficult, especially when there are multiple pursuers to consider. The algorithm in this paper simplifies this process (for the assumed conditions) by considering the reachability of each pursuer separately, producing a set of decoupled constraints that are easily combined. Similar to the "least restrictive control scheme," the result is a set of constraints that keeps the vehicle safe while some other objective can be pursued.

Other work includes Qu et al. (2004), in which safe, analytic trajectories are found given robot and obstacle dynamic constraints, but only on a finite horizon with a given endpoints and without conditions for propagating safety. In Vatcha and Xiao (2009), an explicit (finite) candidate trajectory is taken as an input, and raw sensor data is checked against a maximum velocity parameter to see if the trajectory is safe. This approach assumes a coarser level of available information, and is unable to directly generate safe, infinite trajectories. In Van den Berg et al. (2008), guaranteed safety is achieved for multi-agent systems in which all the agents use the same collision avoidance policy, but this cannot handle unpredictable agents.

1.3 Outline

The formulation presented here guarantees, subject to the aforementioned assumptions, infinite horizon safety that is robust to unpredictable obstacle behavior by using the reachability sets of the dynamic obstacles to define invariant safe states in velocity space. This paper first reviews unicycle model reachability sets and velocity obstacles in Sects. 2 and 3, respectively. The two are then combined, and sufficient conditions for guaranteed safety in the form of Velocity Obstacle Sets are presented in Sect. 4. The implementation of these constraints in iterative planner is discussed in Sect. 5, and simulation results are presented in Sect. 6.

2 Unicycle model reachability sets

This section will find the boundaries of regions in physical space that may result in collisions, as a function of time. These results will be converted into velocity space constraints in Sect. 4.

Cockayne and Hall (1975) find "the set of all possible positions" for a particle that "moves in the plane with constant speed and subject to an upper bound on the curvature of its path." These dynamics match the standard unicycle model used here, and the results can be directly applied to find the reachability set of a given obstacle as a function of time. Without loss of generality, consider an obstacle P that starts at the origin with heading (clockwise angle measured from the y axis) $\theta = 0$ at time $t = 0$. P moves with constant speed v, and it has a maximal turn rate (radians per second) of ω. Equivalently, P has a minimal turning radius $\rho = \frac{v}{\omega}$.

It is shown in Cockayne and Hall (1975) that the reachable region at time t is bounded by 4 parametric curves (Fig. 1). $A_{1,2}$ are given by

$$A_1(\theta, t) = \begin{bmatrix} -\rho(1-\cos\theta) + (vt+\rho\theta)\sin\theta \\ -\rho\sin\theta + (vt+\rho\theta)\cos\theta \end{bmatrix}, \tag{1}$$

$$A_2(\theta, t) = \begin{bmatrix} \rho(1-\cos\theta) + (vt-\rho\theta)\sin\theta \\ \rho\sin\theta + (vt-\rho\theta)\cos\theta \end{bmatrix}, \tag{2}$$

with $-wt \le \theta \le 0$ for A_1 and $0 \le \theta \le wt$ for A_2. The points on these boundaries correspond to the minimal-time Dubins paths (Dubins 1957) of maximal-rate turn followed by driving straight (black and blue dashed lines in Fig. 1). $B_{1,2}$ are given by

$$B_1(\psi, t) = \begin{bmatrix} -\rho(2\cos\psi - 1 - \cos(2\psi - wt)) \\ \rho(-2\sin\psi - \sin(-2\psi - wt)) \end{bmatrix}, \tag{3}$$

$$B_2(\psi, t) = \begin{bmatrix} \rho(2\cos\psi - 1 - \cos(2\psi - wt)) \\ \rho(2\sin\psi - \sin(2\psi - wt)) \end{bmatrix}, \tag{4}$$

with $-\psi^\star \le \psi \le 0$ for B_1 and $0 \le \psi \le \psi^\star$ for B_2. $\psi^\star$ is found by solving

$$2\cos\psi^\star - 1 - \cos(2\psi^\star - wt) = 0. \tag{5}$$

The points on these boundaries correspond to trajectories of maximal-rate turn in one direction followed by maximal-rate turn in the other direction (red and magenta dashed lines in Fig. 1).

2.1 Simplified collision region

Equations (1)–(4) describe the possible locations of the obstacle for a given time t. This region is grown by the collision radius r to find the set of locations at which the robot could be in contact with the obstacle at time t. For simplicity, it is assumed that the host robot is circular with radius r_1 and the obstacles are circles of radius r_2, such that the collision radius r is $r = r_1 + r_2$. As long as the robot is outside of this grown collision region CR(t) (light blue in Fig. 2) at time t, there will not be a collision with the obstacle at the specified instant in time.

To simplify the representation of the various segments bounding the collision region, segment S_5 (solid black line in Fig. 2) is introduced to replace the grown boundaries associated with $B_{1,2}(\psi, t)$ (solid magenta line). This expands the original collision region, safely producing a larger region to avoid, referred to as the *simplified collision region* (SCR). (See Appendix A.1 for discussion of why this is a safe simplification.) This region is defined for each obstacle at a specific time, written SCR$_i(t)$, where i indexes the obstacles. Note however, that as $t \to \infty$, SCR and CR cover the same space (see Figs. 3 and 4). This simplification is made because, when combining all the candidate boundary points of the VOS together into a single curve, it is convenient to use the fact that the value of the parameter θ uniquely identifies each point on the boundary of a given SCR(t); SCR is convex but CR is not. To work with the exact CR, additional rules would need to be built into the function that combines candidate points and segments together into a continuous

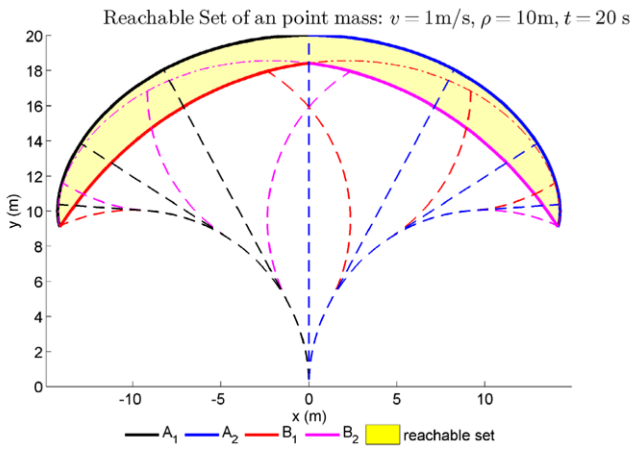

Fig. 1 Example reachability set for $v = 1$ m/s, $\rho = 10$ m, $t = 20$ s. The reachable region is bound by segments of the 4 curves A_1, A_2, B_1, B_2. Corresponding trajectories to points on these curves are shown in *dashed lines*

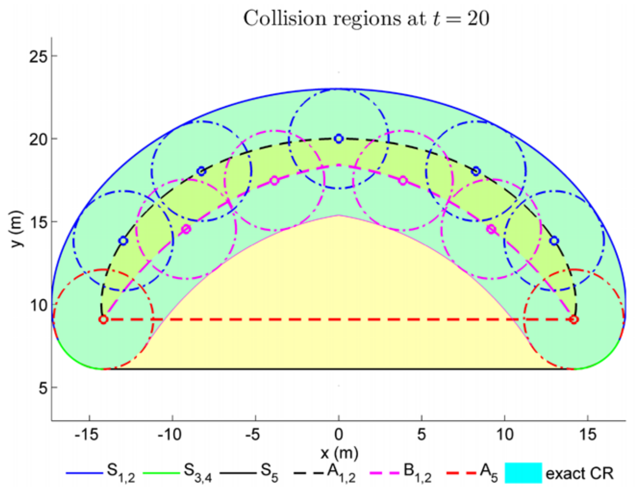

Fig. 2 (Color online) Example simplified collision region at $t = 20$ s. The region from Fig. 1 is grown by the collision radius to produce CR(t), in *light blue*. CR(t) is then expanded using S_5 into SCR(t), the entire enclosed area, for simplicity

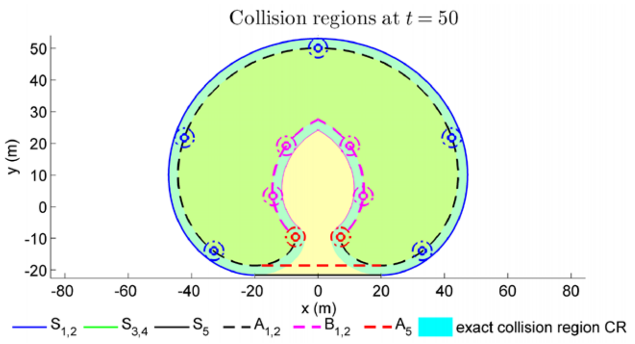

Fig. 3 Simplified collision region at $t = 50$ s

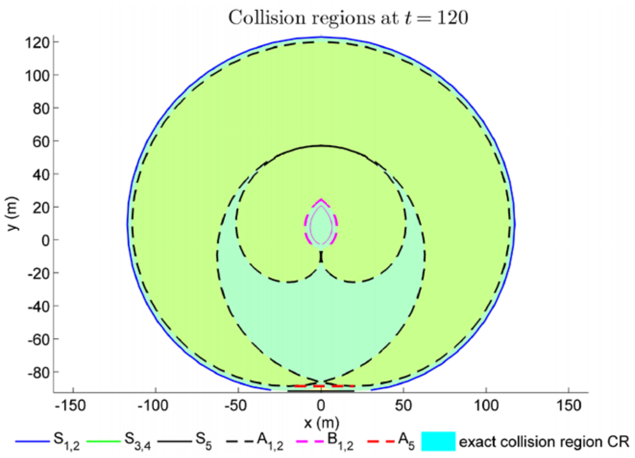

Fig. 4 Simplified collision region at $t = 120$ s. The shading of the collision regions is not applied where it overlaps with itself. Here, both CR and SCR essentially look like disks, and the level of approximation is negligible

loop. Note, that for the final boundary of the VOS, this simplification introduces very little conservatism: in fewer than 1% of cases are the resulting velocity space constraints any different (Wu 2011); that is, candidate boundary points on Q_5 are seldom found in the final boundary of the VOS.

Figure 2 shows that the boundary of the simplified collision region for a given instant in time is a combination of up to 5 possible segments, $S_{1,2,3,4}(\theta, t)$ and $S_5(t)$. Four of these segments are parameterized in θ, which is the direction of the normal vector as measured from the y axis, ranging from $-\pi$ to π. That is, at any point on $S_{1,2,3,4}(\theta, t)$, the normal unit vector $\hat{n}$ is given by

$$\hat{n}(\theta) = \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}. \tag{6}$$

This equation is easily confirmed by checking that the normal vector is orthogonal to the tangent:

$$\hat{n}(\theta) \cdot \frac{\partial S_j}{\partial \theta} = 0 \quad j = 1, \dots, 4. \tag{7}$$

$S_{1,2}(\theta, t)$ come from growing out $A_{1,2}(\theta, t)$ by adding r to (1) and (2) along the outward (with respect to the interior of the reachable set) normal. They are parametrically represented as

$$S_1(\theta, t) = \begin{bmatrix} -\rho(1-\cos\theta) + (vt + \rho\theta + r)\sin\theta \\ -\rho\sin\theta + (vt + \rho\theta + r)\cos\theta \end{bmatrix}, \tag{8}$$

$$S_2(\theta, t) = \begin{bmatrix} \rho(1-\cos\theta) + (vt - \rho\theta + r)\sin\theta \\ \rho\sin\theta + (vt - \rho\theta + r)\cos\theta \end{bmatrix}, \tag{9}$$

with respective domains

$$S_1\colon \quad \max(-\omega t, -\pi) \le \theta \le 0, \tag{10}$$

$$S_2\colon \quad 0 \le \theta \le \min(\omega t, \pi). \tag{11}$$

$S_{3,4}(\theta, t)$ are circular arcs of radius r that wrap around the left and right bottom corners of the reachability set. They are parametrically represented as

$$S_3(\theta, t) = \begin{bmatrix} -\rho(1-\cos(\omega t)) + r\sin\theta \\ \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{12}$$

$$S_4(\theta, t) = \begin{bmatrix} \rho(1-\cos(\omega t)) + r\sin\theta \\ \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{13}$$

with respective domains

$$S_3\colon \quad -\pi \le \theta \le -\omega t, \tag{14}$$

$$S_4\colon \quad \omega t \le \theta \le \pi. \tag{15}$$

Note that for $t > \frac{\pi}{\omega}$, $S_{3,4}$ are no longer part of the boundary of $\mathrm{SCR}_i(t)$.

Finally, $S_5(t)$ is the horizontal line segment joining the lowest points in $S_{1,2,3,4}(\theta, t)$. These are either $S_3(-\pi, t)$ and $S_4(\pi, t)$, or $S_1(-\pi, t)$ and $S_2(\pi, t)$, depending on which pair of curves is defined for $\theta = \pm\pi$ at time t. For this segment, the outward normal unit vector is $[0\ -1]^T$.

$S_{1,2,3,4}(\theta, t)$ can be combined into a piecewise curve $R(\theta, t)$ for which $\theta \in [-\pi, \pi)$. The region below R and above S_5 defines the simplified collision region $\mathrm{SCR}_i(t)$ for obstacle i at any time t. For clarity, specific segments $S_i(\theta, t)$ will be referred to instead of $R(\theta, t)$ as a whole.

3 Velocity obstacles

The objective of this section is to review the conversion of constraints in physical space to constraints in *velocity space*. Fiorini and Shiller (1998) introduce the concept of *velocity obstacles*. Given a set of coordinates $X(t = \tau)$ that must not be entered at time τ (collision region and time of a physical obstacle), there is a corresponding set of velocities $V(t = \tau)$ found by simply dividing X by τ. If the vehicle were to start at the origin at $t = 0$ and take a linear, single-velocity trajectory using any of the velocities in the set $V(\tau)$, it would end up at a coordinate inside set X at time τ. Any other single-velocity trajectory would successfully stay clear of this constraint. This set $V(t)$ of velocities to avoid, associated with collisions occurring at a specific future time, is an instantaneous velocity obstacle.

For moving and stationary obstacles that exist for longer than a single time instant, the host vehicle must jointly avoid all the collision regions at their matching time instants, as given by the trajectory of the obstacles. The union of these resulting constraints form a 2D shape in velocity space. This defines the velocity obstacle

$$\mathrm{VO}(t_0, t_f) = \bigcup V(t) \quad \forall t \in [t_0, t_f]. \tag{16}$$

Single-velocity trajectories using velocities from inside this velocity obstacle would intersect the constrained regions at some time within the specified window. If the obstacle is moving with fixed velocity over the duration of interest, this results in the linear velocity obstacle. Fiorini and Shiller (1998) shows that the linear velocity obstacle is a truncated segment of the profile of a cone that expands from the origin towards the moving object. Where the truncation occurs is determined by the time window; as the lower time bound approaches 0, the open end of the cone tends towards infinity, and as the upper time bound approaches infinity, the sharp end of the cone converges to the obstacle's terminal velocity.

Nonlinear velocity obstacles can be computed for objects that move in arbitrary ways over the time window. Layering the velocity constraints generated at each time instant produces a warped cone, and the calculation and representation of this shape is discussed in Shiller et al. (2005). Velocities outside of this region generate safe, single-velocity trajectories; the safety of nonlinear or speed-varying *robot* trajectories are explored in Wilkie et al. (2009). The standard approach in the existing literature requires a mapping that gives the location of the object as a function of time over the specified window. This typically limits the horizon to accurately predictable time-scales and requires any prediction error to be handled by growing the collision size.

4 Velocity obstacle sets

The objective of this work is to find conditions that guarantee collision avoidance, even without exact knowledge of the obstacles' future locations as functions of time. This is achieved by mapping the reachability sets of the obstacles into regions in velocity space and using the velocity obstacle concept. The conditions are calculated as a function of the obstacles' speeds, turn-rates, and current locations and headings.

Given the speed and turn-rate constraint of any unicycle model obstacle, one can compute an over-bound $\mathrm{SCR}_i(t)$ of its dynamically feasible collision set as a function of time

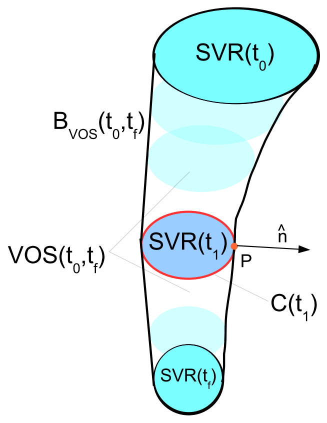

Fig. 5 Example velocity obstacle set (region in velocity space). $\mathrm{VOS}(t_0, t_f)$ is the union of $\mathrm{SVR}(t)$ for $t \in [t_0, t_f]$. The boundary of each $\mathrm{SVR}(t)$ is $C(t)$. $C(t)$ is composed of segments $Q_{1,2,3,4}(\theta, t)$ and $Q_5(t)$. The boundary of VOS is B_{VOS}. A point P on the boundary has an associated normal vector $\hat{n}$

(Sect. 2.1). At every instant in time, one can divide $\mathrm{SCR}_i(t)$ by the time to extract a corresponding region in velocity space, the *simplified velocity region* (SVR), expressed as

$$\mathrm{SVR}_i(t) = \frac{\mathrm{SCR}_i(t)}{t} \tag{17}$$

for any $t > 0$ (see Sect. 4.3 for limits on domain of t). The boundary $C_i(t)$ of $\mathrm{SVR}_i(t)$ consists of the segments

$$Q_{1,2,3,4}(\theta, t) = \frac{1}{t} S_{1,2,3,4}(\theta, t), \tag{18}$$

$$Q_5(t) = \frac{1}{t} S_5(t), \tag{19}$$

using (8)–(15). Any input outside of this region in velocity space will return a linear, single-speed trajectory that will be safe at the given instant. For safety over a window of time, one needs to layer all the instantaneous constraints together. The union of these constrained regions in velocity space is defined as the *velocity obstacle set* (VOS) (Fig. 5), which represents all of the single-velocity trajectories that could possibly generate a collision within the time window. For the ith obstacle,

$$\mathrm{VOS}_i(t_0, t_f) = \bigcup \mathrm{SVR}_i(t) = \bigcup \frac{\mathrm{SCR}_i(t)}{t}, \quad \forall t \in [t_0, t_f]. \tag{20}$$

The VOS is thus a nonlinear velocity obstacle computed using the entire reachability set of a dynamic object, instead of just a single estimated trajectory as in prior work (Shiller et al. 2005). Velocities outside of the VOS give a linear trajectory that is guaranteed safe with respect to all dynamically feasible trajectories of the obstacle for the duration of the time window.

As demonstrated below, using the reachability sets as a function of time is a tractable way to compute the VOS. On the other hand, it would not be computationally feasible to arrive at the same theoretical result by simulating all possible physical trajectories over an infinite time horizon, computing the associated velocity obstacles, and then taking the union of the results. Ultimately, the desired result is a representation of the boundary of the VOS.

4.1 Conditions for boundary points of the VOS

This subsection derives Algorithm 1 for finding the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$ of the region $\mathrm{VOS}_i(t_0, t_f)$, where $t_0 > 0$ and $t_f \leq \infty$ (see Sect. 4.3 for a detailed discussion of these limits). All candidate points from the curves $C_i(t)$ whose time derivatives are perpendicular to the normal vector are found. These points define the points of tangency between $\mathrm{SVR}_i(t)$ and $\mathrm{SVR}_i(t+\delta t)$ (see Fig. 5). The set of all points satisfying this condition contains the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$, as summarized in Remark 1. A detailed explanation follows.

Every point P on $B_{\mathrm{VOS}_i}(t_0, t_f)$ must be a boundary point on at least one of the underlying simplified velocity regions $\mathrm{SVR}_i(t)$ for some $t \in [t_0, t_f]$ (Fig. 5). This follows trivially from the fact that P must be found in some $\mathrm{SVR}_i(t)$ by definition of VOS_i in (20), and that if P were on the interior instead of the boundary $C_i(t)$ of $\mathrm{SVR}_i(t)$, some neighboring point of P would lie outside of $B_{\mathrm{VOS}_i}|_P$.

With the exception of corners, every point P on the curve $B_{\mathrm{VOS}_i}(t_0, t_f)$ has a local outward normal vector $\hat{n}|_P$. See Fig. 7 for an example of a corner. Corners in the boundary would have two different normals approaching from either side, and the normal at that point is undefined. These intersections between local boundary segments are dealt with at the last step in this approach, but the following conditions applied locally to each segment and its normal are still valid as necessary conditions of boundary points.

By definition of a boundary, there must not be any neighboring points inside $\mathrm{VOS}_i(t_0, t_f)$ that reach farther out in the direction of $\hat{n}|_P$. This condition implies the standard equation for the normal vector

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} B_{\mathrm{VOS}_i}|_P(s) = 0, \tag{21}$$

where s is any variable parameterizing the curve B_{VOS_i}. $B_{\mathrm{VOS}_i}(s \pm \delta s)$ gives relevant candidate neighboring points of P within the set VOS_i that must not be farther out along $\hat{n}|_P$. Similarly, since P is found in some $\mathrm{SVR}_i(t)$ which is in VOS_i, one can also write the necessary condition

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} C_i|_P(s) = 0 \tag{22}$$

for any variable s for which $C_i(s)$—the boundary of SVR_i—is continuously defined such that $C_i(s \pm \delta s) \in \mathrm{VOS}_i(t_0, t_f)$.

C_i is composed of the segments $Q_{1,2,3,4}(\theta, t)$ and $Q_5(t)$ from (18)–(19), and the parametric expressions for these segments can each take the place of C_i in (22), yielding

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} Q_{i,j}|_P(s) = 0, \tag{23}$$

where i indexes the dynamic obstacle, and $j \in \{1, \ldots, 5\}$ as piece-wise segments of C_i.

By choosing $s = \theta$ where θ is the parameter for the segments $Q_{1,2,3,4}(\theta, t)$, (7) is recovered (scaled by a factor of $\frac{1}{t}$), the solution of which is (6). This means that, if P is found on some segment $Q_{1,2,3,4}(\theta, t)$, then the normal vector $\hat{n}$ associated with B_{VOS_i} is the same normal vector

$$\hat{n}|_P(\theta) = \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \tag{24}$$

associated with $\mathrm{SCR}_i(t)$ and $\mathrm{SVR}_i(t)$. Similarly, $Q_5(t)$ could be generically parameterized by arc-length s, recovering the normal vector

$$\hat{n}|_{P,Q_5(t)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{25}$$

Intuitively, these results indicate that the normal vectors associated with the overall boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$ at any point P must match with the normal vectors found locally on the segments $Q_j(t)$ at P (see Fig. 5). Otherwise, a point on $Q_j(t)$ would lie outside the boundary.

The expression for $\hat{n}|_P$ can then be substituted into the necessary condition ((23) with the time parameter t substituted for s) to obtain

$$\begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \cdot \frac{\partial}{\partial t} Q_{i,j}|_P(\theta, t) = 0; \quad j \in \{1, \ldots, 4\} \tag{26}$$

and

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \frac{\partial}{\partial t} Q_{i,5}|_P(t) = 0 \tag{27}$$

as necessary conditions for a point P to be on the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$, as long as $Q_{i,j}(\theta, t \pm \delta t) \in \mathrm{VOS}_i(t_0, t_f)$. This latter condition is true for all $t \notin \{t_0, t_f\}$.

Remark 1 Any point P in $B_{\mathrm{VOS}_i}(t_0, t_f)$ must either satisfy (26) or (27), or lie on the curves $C_i(t_0)$ or $C_i(t_f)$. Graphically, (26) and (27) describe points that are local tangent points between $\mathrm{SVR}_i(t)$ and $\mathrm{SVR}_i(t+\delta t)$ (Fig. 5). Finding all such points results in a larger set that contains all possible points in $B_{\mathrm{VOS}_i}(t_0, t_f)$. This set also includes local extrema that are not necessarily global boundary points; see Fig. 6. Algorithm 1 is the result of these conditions.

4.2 Solving the necessary conditions

This section show in detail how (26) and (27) are solved. Without loss of generality, assume the host robot is located at the origin, and the obstacle is located at (x_0, y_0), pointing along the positive y axis (so that the heading is zero). A different initial condition can be first rotated to match this heading, and the corresponding (x_0, y_0) is easily computed (lines 1 and 20 in Algorithm 1).

Algorithm 1: Solving for the boundary of VOS

```
   Input: obstacle$_i$ : $\{x_0, y_0, \theta_0, v, \rho\}, t_0, t_f$
   Output: $B_{\mathrm{VOS}_i}$
1  rotate $(x_0, y_0)$ by $-\theta_0$ (vehicle heading);
2  candidate segments $\leftarrow \emptyset$;
   // add $C(t_0)$ and $C(t_f)$ to consideration
3  for $t = \{t_0, t_f\}$ do
4      find $C(t)$ using (18),(19);
5      add $C(t)$ to candidate segments;
   // add candidate points on $Q_{1,2}$
6  critical angles = solve conditions in (33),(32);
7  for $\theta^\star \in$ critical angles do
8      $t_1 = \max(\frac{\theta^\star}{w}, t_0)$;
       // points make linear segment
          between $Q_{1,2}(\theta^*, t_1)$ and $Q_{1,2}(\theta^*, t_f)$
9      find segments using ((28));
10     add segment to candidate segments;
   // add candidate points on $Q_{3,4}$
11 local segments $\leftarrow \emptyset$;
12 for $t \in [t_0, \min(\frac{\pi}{\omega}, t_f)]$ do      // discretize t
13     points = solve conditions in (41), (39);
14     append points onto local segments;
15 add local segments to candidate segments;
   // add candidate points on $Q_5$
16 $t_h$ = solve condition in (43);
   // endpoints of $Q_5(t_h)$ =
      $Q_{\{2,1\}\mathrm{or}\{4,3\}}(\pm\pi, t_h)$
17 find $Q_5(t_h)$ using (28), (37);
18 add $Q_5(t_h)$ to candidate segments;
   // candidate segments now includes
      all possible boundary points
19 boundary = combine candidate segments;
20 rotate boundary by $\theta_0$;
21 return boundary;
```

(i) *Points on* $Q_{1,2}(\theta, t)$: This corresponds to lines 6–9 in Algorithm 1. First consider points on the segments $Q_{1,2}(\theta, t)$ that may satisfy (26). From (18) and (9),

$$Q_2(\theta, t) = \frac{1}{t}\begin{bmatrix} x_0 + \rho(1-\cos\theta) + (vt - \rho\theta + r)\sin\theta \\ y_0 + \rho\sin\theta + (vt - \rho\theta + r)\cos\theta \end{bmatrix}. \tag{28}$$

The domain for $Q_2(\theta, t)$ is

$$Q_2(\theta, t): \quad t \in [t_0, t_f], \ \theta \in [0, \min(\omega t, \pi)], \tag{29}$$

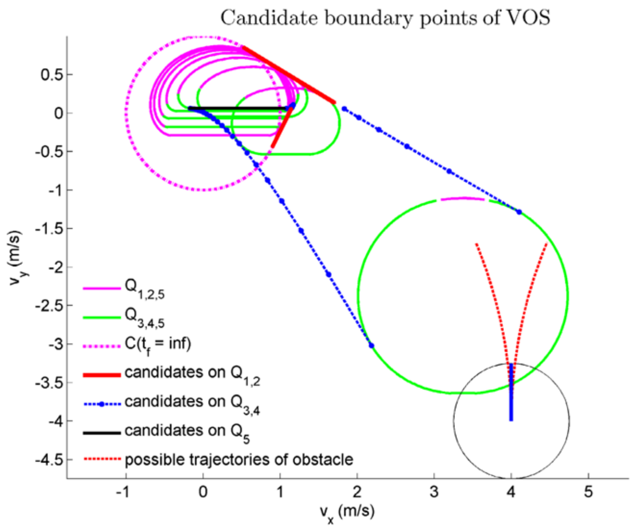

Fig. 6 (Color online) Example candidate points in $B_{\mathrm{VOS}}(t_0, t_f)$, using parameters $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$. Various instances of $C(t)$ are outlined in *green* and *magenta*. Candidate boundary points drawn in *red*, *blue*, and *black*. The physical obstacle is sketched for reference, but note that the rest of the plot is in velocity space, and the physical obstacle cannot actually be expressed in these coordinates

which comes directly from (11). Taking the derivative with respect to time and substituting into (26) yields

$$\frac{\partial Q_2(\theta, t)}{\partial t} = -\frac{1}{t^2}\begin{bmatrix} (x_0 + \rho) + \rho\cos\theta + (r - \rho\theta)\sin\theta \\ y_0 - \rho\sin\theta + (r - \rho\theta)\cos\theta \end{bmatrix}, \tag{30}$$

$$0 = \frac{\partial}{\partial t} Q_2(\theta, t) \cdot \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}, \tag{31}$$

$$0 = -\frac{1}{t^2}((x_0 + \rho)\sin\theta + y_0\cos\theta + r - \rho\theta). \tag{32}$$

For any t and θ in the specified domain that satisfies (32), $Q_2(\theta, t)$ is a potential boundary point in $B_{\mathrm{VOS}}(t_0, t_f)$.

Identical steps lead to the necessary condition

$$-\frac{1}{t^2}((x_0 - \rho)\sin\theta + y_0\cos\theta + r + \rho\theta) = 0 \tag{33}$$

for points of $Q_1(\theta, t)$, defined in the domain

$$Q_1(\theta, t): \quad t \in [t_0, t_f], \quad \theta \in [\max(-\omega t, -\pi), 0]. \tag{34}$$

For any combination of t and θ within the domain of Q_1 that satisfies (33), $Q_1(\theta, t)$ is a potential boundary point in $B_{\mathrm{VOS}}(t_0, t_f)$.

Note that, excluding $t = \infty$ and assuming $t \neq 0$, (32) and (33) can be reduced to functions of θ alone. Each has at most two values of $\theta^\star$ that satisfy the equation within its domain, and these values are easily found numerically. Candidate boundary points are given by $Q_j(\theta^\star, t)$ for $t \in [t_1, t_f]$, where $t_1 = \max(\frac{\theta^\star}{w}, t_0)$. For a solved value of $\theta^\star$, the locus of all candidate boundary points is a linear segment between $Q_j(\theta^\star, t_1)$ and $Q_j(\theta^\star, t_f)$ for $j = \{1, 2\}$ (red lines in Fig. 6). This is because $Q_j(\theta^\star, t)$ (28) is a linear function of $\frac{1}{t}$, which is well defined for $t \neq 0$.

(ii) *Points on $Q_{3,4}(\theta, t)$:* This corresponds to lines 11–15 in Algorithm 1. Next, all possible boundary points on $Q_{3,4}(\theta, t)$ are found. Like segments $S_{3,4}(\theta, t)$ (12)–(13), the domains of segments $Q_{3,4}(\theta, t)$ are

$$Q_3(\theta, t): \quad t \in [t_0, t_f],\ \theta \in [-\pi, -\omega t], \tag{35}$$

$$Q_4(\theta, t): \quad t \in [t_0, t_f],\ \theta \in [\pi, \omega t]. \tag{36}$$

Due to the limits on the domain of θ, when solving for $B_{\mathrm{VOS}}(t_0, t_f)$, one only needs to consider points in $Q_{3,4}$ from times in the range $t \in [t_0, \min(\frac{\pi}{w}, t_f)]$. As seen in Fig. 6, as t increases, $Q_{3,4}$ eventually disappear.

From (18) and (13),

$$Q_4(\theta, t) = \frac{1}{t}\begin{bmatrix} x_0 + \rho(1 - \cos(\omega t)) + r\sin\theta \\ y_0 + \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{37}$$

$$\frac{\partial Q_4}{\partial t} = \frac{-1}{t^2} \times \begin{bmatrix} (x_0 + \rho) - \rho\cos(\omega t) - \rho\omega t\sin(\omega t) + r\sin\theta \\ y_0 + \rho\sin(\omega t) - \rho\omega t\cos(\omega t) + r\cos\theta \end{bmatrix}. \tag{38}$$

Given any value of t from within the domain, the necessary condition from (38) requires

$$\frac{\partial}{\partial t} Q_4(\theta, t) \cdot \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} = 0, \tag{39}$$

which can be solved numerically for θ. For a given t, (39) has between zero and two valid solutions for $\theta \in [0, 2\pi)$. Any of these solutions that fall within the domain $\theta \in [\omega t, \pi]$ is considered a valid candidate boundary point in $B_{\mathrm{VOS}}(t_0, t_f)$.

Similarly, for $Q_3(\theta, t)$, using (12)

$$\frac{\partial Q_3}{\partial t} = -\frac{1}{t^2}\begin{bmatrix} (x_0 + \rho) + \rho\cos(\omega t) + \rho\omega t\sin(\omega t) + r\sin\theta \\ y_0 + \rho\sin(\omega t) - \rho\omega t\cos(\omega t) + r\cos\theta \end{bmatrix}. \tag{40}$$

Given a value of t, the necessary condition

$$\frac{\partial}{\partial t} Q_3(\theta, t) \cdot \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} = 0 \tag{41}$$

can be solved numerically for θ, and solutions that are in the domain $\theta \in [-\pi, \omega t]$ are considered valid candidate boundary points.

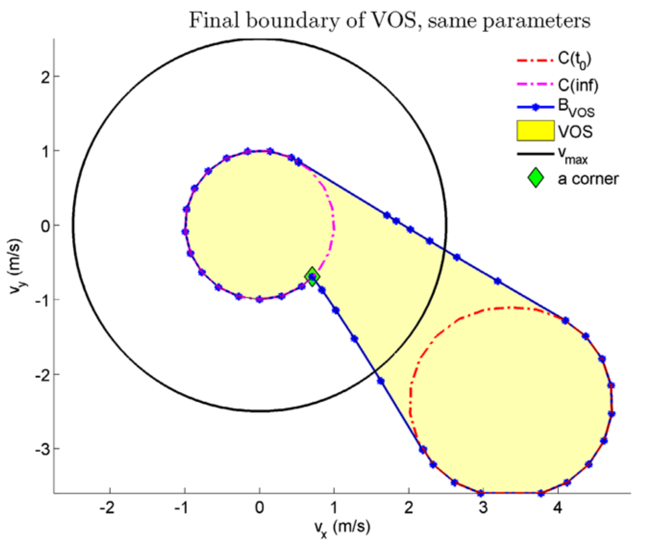

Fig. 7 (Color online) Final boundary of VOS from example in Fig. 6. The same parameters of $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$ are used. $C(t_0 = \epsilon)$ and $C(t_f \to \infty)$ are outlined in *red* and *magenta*. B_{VOS} drawn in *blue*. The maximum host vehicle velocity drawn in *black*; $C(t_0 = \epsilon)$ lies outside of this circle

While the sets of all possible boundary points from segments Q_1 and Q_2 define linear segments given by just their endpoints, the set of candidate points (the dotted blue segments in Fig. 7) found in Q_3 and Q_4 in general cannot be represented analytically in a more compact form. Instead, one can discretize the finite interval $[t_0, \min(\frac{\pi}{w}, t_f)]$ at some resolution, and at every discrete value of t, points satisfying (39) or (41) are found. Solutions points from sequential instances in time are matched together to form candidate boundary segments. The resolution of these segments depend on how finely the time interval is discretized. Note that the duration of this interval is upper-bounded by the time it takes the obstacle to turn around, capping the computational cost of this process.

(iii) *Points on $Q_5(t)$:* This corresponds to lines 16–18 in Algorithm 1. Finally, points on $Q_5(t)$ for any $t \in [t_0, t_f]$ that may be part of $B_{\mathrm{VOS}}(t_0, t_f)$ need to be found. Again, the definitive necessary condition is

$$\hat{n} \cdot \frac{\partial Q_5}{\partial t} = 0, \tag{42}$$

where $\hat{n} = [0\ -1]^T$ (from (25)) and $Q_5(t)$ is the horizontal line segment joining $Q_1(-\pi, t)$ and $Q_2(\pi, t)$, or joining $Q_3(-\pi, t)$ and $Q_4(\pi, t)$, depending on which set of domains of θ contain $\pm\pi$ at the given t. Since $Q_5(t)$ is a straight, horizontal segment, this condition holds true when the two endpoints satisfy their respective conditions of being normal to $\hat{n}|_P(\theta) = [\sin\theta\ \cos\theta]^T = [0\ -1]^T$.

When $\omega t \geq \pi$, $Q_5(t)$ joins $Q_{2,1}(\pm\pi, t)$, and (42) is satisfied if $\pm\pi$ are solutions to (32) and (33). When this is the case, it is easy to show that all the points in $Q_5(t)$ for $t \geq \frac{\pi}{\omega}$ line up with the linear segment between $Q_{2,1}(\theta^\star, t_1)$ and $Q_{2,1}(\theta^\star, t_f)$ described earlier (where $\theta^\star = \pm\pi$). These points do not need to be accounted for a second time.

When $\omega t \leq \pi$, $Q_5(t)$ joins $Q_{4,3}(\pm\pi, t)$, and (42) yields

$$y_0 + \rho \sin(\omega t_h) - \rho\omega t_h \cos(\omega t_h) - r = 0. \tag{43}$$

This can be numerically solved for t_h. Using $Q_3(-\pi, t_h)$ and $Q_4(\pi, t_h)$, $Q_5(t_h)$ (the black line in Fig. 6) is then explicitly found and added to the set of candidate boundary points.

(iv) *Combining the Results:* As described in Remark 1, considered together with the curves $C_i(t_0)$ and $C_i(t_f)$, the solutions to (32), (33), (39), (41) and (43) in their specified domains exhaustively describe all points that could lie on the boundary $B_{\text{VOS}_i}(t_0, t_f)$. Intersections between separately defined candidate segments form corners in the final boundary and are easily found, and a combined perimeter can be traced out (line 19 in Algorithm 1; see Figs. 6 and 7).

For each obstacle, (32), (33), and (43) each need to be solved only once, as a function of just one variable. Equations (39) and (41) need to be repeatedly solved as functions of θ at discretized instances of t, but the range of t is bounded as $t \in [t_0, \min(\frac{\pi}{w}, t_f)]$. $C_i(t_0)$ and $C_i(t_f)$ are directly found using (28) and (37) (plus similar equations for $Q_{1,3}$). All of these computations are very light and can easily be done on-line. Calculating and representing the conditions for each obstacle that guarantee collision avoidance is thus a straightforward process that can be implemented in real-time at each planning iteration. How a planner then uses and checks these constraints is a separate problem.

4.3 Discussion about $C(t_0)$ and $C(t_f \to \infty)$

Just as in the original velocity obstacle formulation (Fiorini and Shiller 1998), as $t_0 \to 0^+$, the velocity obstacle set extends out to infinity in the direction of the current location of the physical obstacle. This comes from increasingly higher speeds required to collide with the obstacle immediately. The velocity space constraint SVR$(t = 0)$ is not defined, since contact between the vehicles at this time depends only on the initial conditions, not any input velocity. Instead, a practical approach is to set t_0 at some $\epsilon > 0$ such that SVR$(t = \epsilon)$ lies beyond input velocities on interest. One method is to set the entire region SVR(ϵ) beyond some maximum speed $v_{\max}$ that the host vehicle can achieve (Fig. 7). To achieve this, a simple, more conservative requirement stemming from an obstacle without turn-rate constraints can be written as

$$\frac{1}{\epsilon}\left\| \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + (r + v_{obs}\epsilon)\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \right\| \geq v_{\max} \quad \forall \phi \in [0, 2\pi]. \tag{44}$$

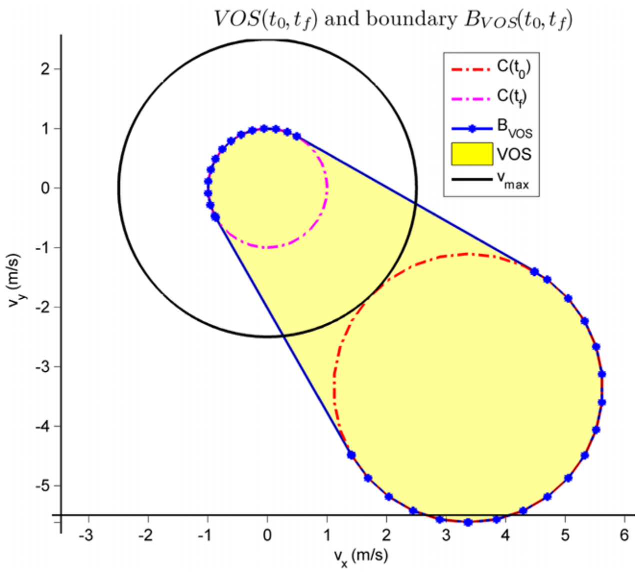

Fig. 8 Final boundary of VOS for obstacle with an infinite turn rate. The same parameters of $x_0 = 4$, $y_0 = -4$, $r = 1.5$, $v = 1$ from Fig. 7 are used, but here $\rho = 0$. This is the VOS for an obstacle with an unlimited turn rate and a maximum speed of v. A more direct solution for this region can also be easily derived from geometric principles

This yields the criteria for selecting $t_0 = \epsilon$

$$\epsilon \leq \frac{\sqrt{x_0^2 + y_0^2} - r}{v_{\max} + v_{obs}}. \tag{45}$$

For an obstacle moving along a known, linear trajectory, the velocity obstacle cone converges to an apex at $\vec{v}_{obs}$ as $t \to \infty$. Here, SVR$(t \to \infty)$ is to a circular disk of radius v centered on the origin. This can be easily derived by taking the limit of (28). Intuitively, if the host vehicle were to drive along any straight trajectory in any direction with a constant speed less than v, the obstacle could eventually catch up and collide. That is, for infinite horizon safety, the host robot must be able to move faster than the unpredictable obstacles.

4.4 Infinite horizon safety guarantees

As explained in the beginning of Sect. 4, any velocity outside of VOS$_i(t_0, t_f)$ for a given obstacle i can be used to generate a single-velocity trajectory that is guaranteed to avoid any dynamically feasible trajectory of that obstacle, for the duration of the time window. For multiple obstacles, the constraints are simply combined; a velocity that is outside of the VOSs of all the obstacles is guaranteed safe for the duration. The only remaining problem is that, if t_f is set to some finite value τ, there is no guarantee that this safety can be propagated. The host vehicle will avoid collisions up to time τ, but nothing can be said about the continued existence of safe trajectories afterwards.

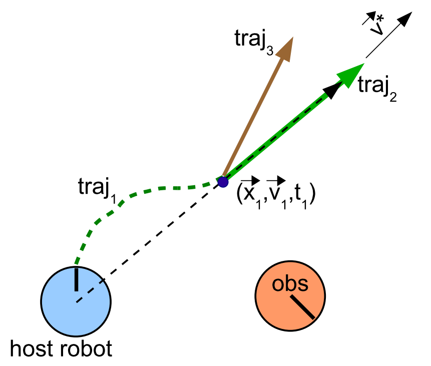

Fig. 9 Different safe trajectories in physical space, velocity obstacles not shown. Let $\vec{v}^\star$ be a velocity outside of the VOS calculated at $t = 0$. The *dotted arrow* is the single-velocity escape trajectory associated with $\vec{v}^\star$, which is guaranteed safe on an infinite time scale. If traj$_1$ terminates at $(\vec{x}_1, t_1)$ with velocity $\vec{v}^\star$, it can be safely continued at least along traj$_2$. By calculating the shifted velocity obstacle at $(\vec{x}_1, t_1)$, other single velocity trajectories (like traj$_3$) that also guarantee infinite horizon safety may be found to continue any partial trajectory ending at $(\vec{x}_1, t_1)$

However, in this formulation, it is perfectly tractable (Sect. 4.3) and reasonable (because there is no prediction of obstacle behavior involved) to set t_f at ∞, and this allows the use of the concept of *invariant safe states*. These are host vehicle states that meet all imposed conditions, and can always be propagated back to itself or some other invariant safe state. For example, in structured environments where other vehicles practice reasonable collision avoidance, coming to a complete stop and staying stationary can be considered such a state (Kuwata et al. 2008). In the presence of our dynamically constrained obstacles, any control input $\vec{v}^\star$ outside of $\text{VOS}_i(t_0,\infty)$ for all obstacles i can be propagated indefinitely, so driving along this trajectory defined by $\vec{v}^\star$ (the dashed arrow in Fig. 9) can be considered an invariant safe state. Intuitively, these are escape trajectories with speed greater than v that eventually leave all the obstacles behind the host vehicle.

In addition to the invariant safe states themselves, any vehicle state that can safely make a transition into an invariant safe state is also safe. This means that an arbitrary finite trajectory from $t = 0$ to $t = t_1$ (traj$_1$ in Fig. 9) terminating with the conditions

$$\vec{x}(t_1) = \vec{x}_0 + \vec{v}^\star t_1 \tag{46}$$

and

$$\vec{v}(t_1) = \vec{v}^\star \tag{47}$$

at some value of $t_1 \ge t_0$, without collisions for $t \in [0, t_1]$, is guaranteed to have access to a trajectory that is safe on the interval $t \in [t_1,\infty)$. If t_0 in the VOS computation is set to essentially 0 (using ϵ as described above), any single-velocity trajectory generated by immediately following a valid $\vec{v}^\star$ trivially fits this description, but more complicated solutions can be found as well (traj$_1$ in Fig. 9). Depending on specific scenario and host robot dynamics, various means can be used to generate the finite trajectories from $t = 0$ to $t = t_1$ and check for collisions.

Note that this infinite horizon safety requires the space to be non-enclosed. It is assumed that the VOSs for any static obstacles in the environment have also been calculated and considered when searching for a valid $\vec{v}^\star$. It is easy to show that the associated VOS for a static object would simply be an infinite cone in the direction of the object that engulfs the profile of the object. Therefore, if the host robot was inside an enclosed space (*i.e.,* indoors), the VOSs of the surrounding walls would cover the entire space of velocities; there would be no guaranteed safe single-velocity trajectories, since any trajectory would eventually hit a wall.

That is not to say that no safe solution exists within an enclosed space. In large enough rooms with a small number of slow obstacles, there very well may be reactive policies that the host robot can adopt to stay safe. Such a solution would depend heavily upon the assumed dynamics and the geometry of the initial conditions, and may perhaps be found using a pursuit-evasion approach (Rzymowski 1986; Borowko et al. 1988; Chodun 1989; Sinitsyn 1993; Pashkov and Sinitsyn 1995; Mitchell et al. 2005), but not by the algorithm presented here. A finite-horizon VOS formulation can be applied to achieve reasonable obstacle avoidance, but such an approach loses all infinite-horizon safety guarantees: there may be no safe trajectories in the future, even if previous planning iterations had returned safe solutions (Wu 2011).

4.5 Time-shifted VOSs

Similarly, but more generally, for any finite trajectory that terminates at time t_1 at an arbitrary position $\vec{x}_1$ with velocity $\vec{v}_1$, the terminal location $\vec{x}_1$ is safe if it is not in the reachability set $\text{SCR}_i(t_1)$ of any of the obstacles. If this condition is satisfied, then there might be a safe, infinite horizon trajectory on the interval $t \in [t_1,\infty)$ if $\vec{v}_1$ lies outside of every *time shifted VOS with the host robot moved to* $\vec{x}_1$. That is, assuming the robot gets to $\vec{x}_1$ at t_1, one must determine if continuing indefinitely at $\vec{v}_1$ is guaranteed safe.

This time-shift significantly extends the practical applicability of the VOS concept; multi-velocity trajectories can now be considered such that the infinite-horizon constraints imposed by the time-shifted VOS do not adversely restrict short-term maneuvers whose safety can be checked more directly. Such an implementation would directly alleviate the conservative behavior that inevitably results from using entire reachable sets to avoid the obstacles; other means may be employed to ensure safety on the immediate horizon up to time t_1.

To perform this shift, the obstacle is translated by $-\vec{x}_1$ such that the coordinates are defined relative to the new host vehicle location. In addition, any terms t are replaced by $t+t_1$, with the exception of the factor of $\frac{1}{t}$ in front, which remains the same. This manipulation of the time terms accounts for the obstacles having already moved for duration of t_1, before the host robot even arrives at $\vec{x}_1$ (or, after the translation is applied, "arrives at the origin" instead of $\vec{x}_1$).

For example, the expression for $Q_2(\theta, t)$ in (28) converts to

$$Q_2(\theta,\tau,\vec{x}_1,t_1)=\frac{1}{\tau}\begin{bmatrix} x_0-x_1+\rho(1-\cos\theta) \\ \quad +(v(\tau+t_1)-\rho\theta+r)\sin\theta \\ y_0-y_1+\rho\sin\theta \\ \quad +(v(\tau+t_1)-\rho\theta+r)\cos\theta \end{bmatrix} \tag{48}$$

with domain

$$Q_2(\theta,\tau,\vec{x}_1,t_1):\quad \tau\in[t_0,t_f],\ \theta\in[0,\min(\omega(\tau+t_1),\pi)]. \tag{49}$$

Here, $\tau = t - t_1$, such that $\tau = 0$ starts when the robot reaches $\vec{x}_1$. Using the steps given in Sect. 4.2, the necessary condition for boundary points on $Q_2(\theta,\tau,\vec{x}_1,t_1)$ (32) becomes

$$0=-\frac{1}{\tau^2}((x_0-x_1+\rho)\sin\theta+(y_0-y_1)\cos\theta + vt_1+r-\rho\theta. \tag{50}$$

Similar modifications can be performed for the segments $Q_{1,3,4,5}(\theta,\tau,\vec{x}_1,t_1)$.

If the velocity obstacle sets are calculated using these modifications, they represent all the potentially dangerous single-velocity trajectories that start from $\vec{x}_1$ at t_1, given the current locations of all the obstacles. Hence, any finite trajectory terminating at an arbitrary $\vec{x}_1$ at t_1 can be safely extended if the terminal velocity $\vec{v}_1$ of this trajectory lies outside of the shifted VOS of each obstacle. Note that the conditions described earlier in (46) and (47) are a specific case of this condition; $\vec{v}^\star$ is guaranteed to be one, among possibly many others, velocity that is outside of the VOSs calculated at $(\vec{x}_1, t_1)$ In Fig. 9, $\vec{v}^\star$ gives the safe trajectory traj$_2$, and other safe trajectories like traj$_3$ may be found using the shifted VOSs.

The shifted VOS can be used in on-line planning to account for delays; the planner can assign a duration of time for computation, use that as t_1, and set $\vec{x}_1 = \vec{v}t_1$ as the expected propagation, where $\vec{v}$ is the current velocity. Assuming the current velocity is maintained during planning, this formulation properly handles on-board computation time and does not compromise guaranteed collision avoidance.

5 Iterative updates

A trajectory that satisfies the velocity obstacle set constraint calculated with $t_f = \infty$ for all the obstacles can be continued indefinitely without collision. However, if the goal location does not lie along the current trajectory, or if the goal location is changed, the robot would never reach the goal, and the trajectory needs to be updated.

At any point in time, the planner can use the current state information to recompute the VOS of each obstacle. The safety guarantee is that, if the current trajectory had satisfied the infinite horizon VOS constraints at some previous time, the current velocity will still lie outside of all the updated VOSs. In addition, if there are other velocities that satisfy this condition, there may be other *feasible* trajectories that are guaranteed safe as well. These may be single-velocity trajectories using a valid $\vec{v}^*$, or more complex finite trajectories whose terminal conditions satisfy either (46) and (47) or the more general requirements given by the time-shifted VOSs (Sect. 4.5). Here, feasibility refers to scenario-dependent constraints on the host robot's dynamics; the robot may not be able to instantly change to a new velocity that is too different from the current one. If a significant amount of time or displacement is required to make the transition, the time-shifted VOS can be used.

Typically, after the VOSs are updated, more than one safe trajectory will be available. Any trajectory pointed in the direction of the goal will eventually reach it. If all velocities in this direction are unsafe, a safe velocity can be chosen subject to various heuristics. One could choose a trajectory that gets the closest to the desired way-point; this results in picking a velocity whose direction is most closely aligned with the goal. Other selection metrics could include measures of control effort.

At any later time, it is valid to recompute the VOSs and attempt to improve the trajectory. Re-planning can occur on regular intervals, or it can be triggered by how well the current plan approaches the goal. A better trajectory may not exist, and there is no guarantee that the goal can ever be reached, but at least the current plan can be executed safely and extended indefinitely. Therefore, after solving for the velocity obstacle sets given the initial state, if there exists at least one feasible velocity $\vec{v}^\star$ outside of all the VOSs, an iterative planner is guaranteed to never encounter collisions. Other receding horizon methods have not been able to provide this guarantee for unpredictable obstacles. Note that the safety guarantee does not depend on the updates, which only help the robot seek the way-point.

This approach is not *complete*; velocity obstacle sets only return single-velocity escape trajectories. There may exist a curved trajectory that extracts the host robot from a "surrounded" situation or brings it to the goal, but this method will not find it. However, this method does guarantee that it will never put the robot in such a situation, if that is not the initial condition.

6 Simulation results

This section presents simulation results using the iterative planning algorithm in Sect. 5. The host robot tracks randomly generated way-points in the presence of four obstacles moving with constant speed and limited turn-rate. The planner has access to these two parameters of each obstacle, as well as the obstacles' initial locations and headings. Whenever the planner performs an update, it is given the new locations and headings of each obstacle. The obstacles' turn-rates are chosen randomly. As derived and proven in Sect. 4, using these parameters, the planner guarantees infinite horizon safety by computing the velocity obstacle sets. This simulation runs indefinitely without the robot colliding with any obstacles, and select screen-shots are shown below to illustrate the safety behavior resulting from Algorithm 1.

The host robot and the obstacles are all disks of radius 0.5 m. The obstacles move with speed 1 m/s and turn at a maximum rate of $\pi/5$ rad/s, and the host robot has a maximum speed of 2.5 m/s and has an unbounded turn rate. It is necessary for the environment to be unbounded and for the host robot to have a faster speed, while the turn-rate of the host robot does not affect guaranteed safety. A bounded turn-rate limits the host robot's ability to effectively navigate way-points, especially without an appropriately designed short-term planner (Wu 2011).

To represent all feasible obstacle behavior, obstacle turn rates are uniformly sampled from their full range every one to two seconds. If new turn-rates were sampled too often, the trajectories would become noisy—but mostly straight—lines, which would not be a challenging collision-avoidance scenario. To keep them clustered in a local area and to sufficiently test limiting cases, obstacles are made to turn at maximum rate whenever outside a bounding box. Collisions between obstacles do not affect their trajectories so as not to violate the assumed dynamics.

The host robot is sequentially fed randomly generated way-points that are biased to be close to the obstacles in order to create interesting scenarios. Every second, or whenever a way-point is reached, the velocity obstacle sets are updated according to Algorithm 1. The robot then selects a safe velocity whose direction is the most closely aligned with the relative direction of the next way-point, as discussed in Sect. 5. If there are multiple safe velocities pointed in the same desired direction, the velocity with speed closest to the preferred value of 1.5 m/s is chosen. This tie-breaker can be set to anything within the robot's range.

Velocities are selected by first testing key speeds along preferred direction. If no direct solution within the safe region is found, points defining the boundaries of each VOS are then considered. These points are checked with respect to each other VOS. When the best velocity is found on a boundary, it is then slightly perturbed such that it is on the exterior of the velocity obstacle, so that the robot will not graze the object.

On a desktop PC (quad-core Intel i7 at 3.20 GHz with 12 Gb RAM), each VOS typically takes less than 0.05 seconds to compute, and selecting a valid velocity with up to twelve velocity obstacle sets takes less than 0.01 seconds. All computations are run in Matlab.

A few snapshots from the simulation are shown in Fig. 10. At $t = 5.5$ seconds, a velocity pointed directly to the goal is guaranteed safe, and this new velocity is selected. At $t = 6.5$, that direction is continued at a slightly slower, preferred speed that had become safe. This change comes from the two left-most obstacles having turned away from the robot's intended trajectory.

Similarly at $t = 11.3$ seconds, a velocity is found that squeezes between two velocity obstacles sets. 2 seconds later, the updated VOSs show that this direct trajectory is still safe, and the actual trajectory of the left-most obstacle had allowed wider margins in one direction while the upper obstacle continues to threaten collision if the host robot were to slightly alter its course.

In the few seconds before $t = 22.7$, the planner had chosen a trajectory that would bring the robot too far the left of the goal. At $t = 22.7$, the updated VOSs show that there is a safe trajectory nearly lined up with the goal that jointly avoids both lower obstacles. In the one re-plan between $t = 22.7$ and $t = 24.7$ the robot was able to make one more minor adjustment in its heading, lining up directly with the goal.

At $t = 26.1$ seconds, the new way-point is blocked off and robot must drive around the obstacles. At the next update, it finds a clear trajectory towards the goal. Note that just reaching the goal is not considered sufficiently safe; the robot must maintain an open path forward from there. This allows the simulation to run indefinitely without the robot ever colliding.

7 Conclusion

Velocity Obstacles have been widely applied in predictive environments to generate single-velocity trajectories that avoid collisions with moving obstacles. These formulations require knowledge of the future locations of the obstacles, and thus struggle with long-term safety guarantees. Here, a new method that focuses on the dynamical constraints of the moving obstacles is presented. Modeling the obstacles as single speed, maximum turn-rate vehicles allows the set of all their feasible trajectories on an infinite horizon to be succinctly captured in *velocity obstacle sets* as a function

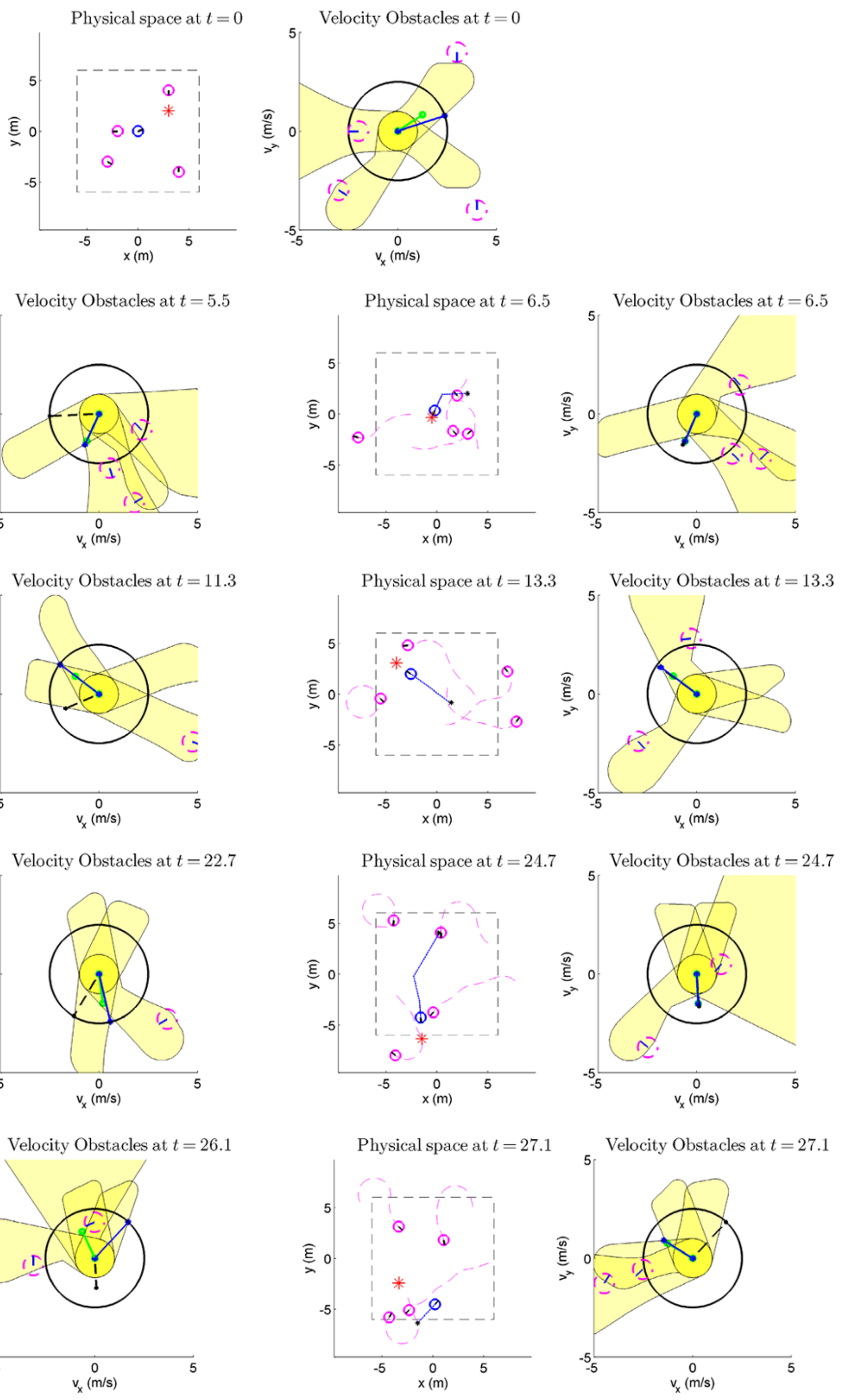

Fig. 10 (Color online) The physical space is shown on the left. The host vehicle is in *blue*, the obstacles are in *magenta*, and the way-point is the *red star*. The vehicles leave a trail of their most recent positions. When the obstacles cross the dashed boundary, they are forced to turn around. On the right, the VOS of each velocity obstacle is shaded in. The maximum velocity of the robot is represented by the *black circle*. The ideal (aligned with goal, preferred speed) velocity is the *green* vector, the best new safe velocity is the *blue* vector, and the velocity before the re-plan is the *dashed black* vector. The safety guarantee is that, until the host vehicle chooses a new command input, this old velocity will still lie outside of all the VOSs when the obstacle states are updated. Just for directional reference, the nearest obstacles are sketched in dashed outlines, though they cannot be truly represented in velocity space

of only their current location and orientation. These regions in velocity space define sufficient conditions for guaranteed collision avoidance on an indefinite time scale, provided that the host robot has a speed advantage and the environment is unbounded. These velocity space constraints do not depend on the host robot dynamics, which can be arbitrarily defined and handled separately. Different dynamical models for the obstacles would require a re-derivation of a similar method based on the same principles. If the assumed conditions are met, the approach outlined here can augment a partial motion planner with rigorous infinite horizon properties. The constraints derived in this paper consider all possible future motions of the obstacles, and thus lead to potentially conservative behavior if the trajectories of the obstacles are in fact more predictable. Therefore, in practical applications, this concept would be best used as a long-term safety check in conjunction with a more sophisticated, predictive short term planner whose finite trajectories have terminal states that satisfy the time-shifted VOS conditions.

Appendix

A.1 Validity of using SCR(t) in place of CR(t)

Guaranteed collision avoidance is maintained when replacing the collision region CR(t) with SCR(t), if SCR(t) includes all of CR(t). Since $S_5(t)$ is the new lower boundary of the region, this is true if no points on the original boundary of CR(t) lie below $S_5(t)$.

$A_5(t)$ joins $A_1(-\omega t, t)$ and $A_2(\omega t, t)$, or equivalently, $B_1(0,t)$ and $B_2(0,t)$ for $\omega t \le \pi$. A_5 joins $A_1(-\pi, t)$ and $A_2(\pi, t)$ for $\omega t \ge \pi$. Proving this modified reachability set includes all of the exact reachability set equivalently shows that SCR(t) is a safe over-bound of CR(t).

To show that defining $A_5(t)$ as the new lower boundary does not exclude points from the original region, one must show that no points in the replaced boundary segments lie below $A_5(t)$. That is,

$$A_{\{1,2\},y}(\theta, t) \ge A_{5,y}(t) \quad \forall |\theta| > \pi \tag{51}$$

and

$$B_{\{1,2\},y}(\psi, t) \ge A_{5,y}(t) \quad \forall \psi \in [0, \psi^\star]. \tag{52}$$

Equation (51) is only a concern for $\omega t > \pi$ (because $\theta \le \omega t$), and within this domain, $A_{5,y}(\theta, t) = A_{2,y}(\pi, t)$. Using the first and second derivatives of $A_{2,y}(\theta, t)$ (see (2)) with respect to t, it is easy to show that $A_{2,y}$ has local minimums at $\theta = \{\pi, 3\pi, \dots\}$. Equation (2) evaluated at these points yields

$$A_{2,y}(\theta, t) = -(vt - \rho\theta), \tag{53}$$

which increases with θ. Clearly, the global minimum occurs at $\theta = \pi$. Hence, (51) is satisfied.

Starting with (4),

$$B_{2,y}(\psi, t) = \rho(2\sin\psi - \sin(2\psi - \omega t)), \tag{54}$$

$$\frac{dB_{2,y}}{d\psi} = \rho(2\cos\psi - 2\cos(2\psi - \omega t)) \tag{55}$$

$$= B_{2,x}(\psi, t) + (1 - \cos(2\psi - \omega t)) \tag{56}$$

$$\ge B_{2,x}(\psi, t). \tag{57}$$

From Sect. 3.2 in Cockayne and Hall (1975), $B_{2,x}(\psi, t) \ge 0$ for the entire domain of ψ. Therefore,

$$\frac{dB_{2,y}}{d\psi} \ge 0 \quad \forall \psi \in [0, \psi^\star]. \tag{58}$$

This is a sufficient condition for (52), since B_y is shown to be an increasing function, and $A_{5,y}(t) = B_{2,y}(0, t)$.

Therefore, the modified reachability set is a valid over-bound of the exact reachability set, so the corresponding collision region SCR(t) is a safe over-bound of the exact collision region CR(t).

References

Bekris, K. E. (2010). Avoiding inevitable collision states: Safety and computational efficiency in replanning with sampling-based algorithms. In *Workshop on "guaranteeing safe navigation in dynamic environments", international conference on robotics and automation* (ICRA-10), Anchorage, AK, May 2010.

Borowko, P., Rzymowski, W., & Stachura, A. (1988). Evasion from many pursuers in the simple motion case. *Journal of Mathematical Analysis and Applications*, *135*(1), 75–80.

Chitsaz, H., & LaValle, S. (2007). Time-optimal paths for a Dubins airplane. In *46th IEEE conference on decision and control* (pp. 2379–2384). New York: IEEE.

Chodun, W. (1989). Differential games of evasion with many pursuers. *Journal of Mathematical Analysis and Applications*, *142*(2), 370–389.

Cockayne, E. J., & Hall, G. W. C. (1975). Plane motion of a particle subject to curvature constraints. *SIAM Journal on Control*, *13*, 197.

Dubins, L. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, *79*(3), 497–516.

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, *17*(7), 760.

Fraichard, T. (2007). A short paper about motion safety. In *IEEE international conference on robotics and automation (ICRA)* (pp. 1140–1145).

Fraichard, T., & Asama, H. (2004). Inevitable collision states—a step towards safer robots? *Advanced Robotics*, *18*(10), 1001–1024.

Frazzoli, E., Dahleh, M., & Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, *25*(1), 116–129.

Friachard, T., & Bouraine, S. (2011). Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. In *Workshop on "guaranteeing motion safety for robots", robotics: science and systems conference* (RSS-11), Los Angeles, CA, June 2011.

Gal, O., Shiller, Z., & Rimon, E. (2009). Efficient and safe on-line motion planning in dynamic environments. In *IEEE international conference on robotics and automation (ICRA)* (pp. 88–93).

Kuwata, Y., Fiore, G. A., Teo, J., Frazzoli, E., & How, J. P. (2008). Motion planning for urban driving using RRT. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1681–1686). 22–26 Septembet 2008.

Large, F., Laugier, C., & Shiller, Z. (2005). Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles. *Autonomous Robots*, *19*(2), 159–171.

L'Esperance, B., Kazemi, M., & Gupta, K. (2011). Analyzing safety for mobile robots in partially known dynamic indoor environments. In *Workshop on "guaranteeing motion safety for robots" at the robotics: science and systems conference* (RSS-11), Los Angeles, CA, June 2011. http://safety2011.inrialpes.fr/.

Mitchell, I., Bayen, A., & Tomlin, C. (2005). A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, *50*(7), 947–957.

Pashkov, A., & Sinitsyn, A. (1995). Construction of the value function in a pursuit-evasion game with three pursuers and one evader. *Journal of Applied Mathematics and Mechanics*, *59*(6), 941–949.

Petti, S., & Fraichard, T. (2005). Safe motion planning in dynamic environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2210–2215). New York: IEEE.

Qu, Z., Wang, J., & Plaisted, C. (2004). A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *IEEE Transactions on Robotics*, *20*(6), 978–993.

Rzymowski, W. (1986). Evasion along each trajectory in differential games with many pursuers. *Journal of Differential Equations*, *62*(3), 334–356.

Schouwenaars, T., How, J., & Feron, E. (2004). Receding horizon path planning with implicit safety guarantees. In *Proceedings of the 2004 American control conference* (Vol. 6, pp. 5576–5581). New York: IEEE.

Shiller, Z., Large, F., & Sekhavat, S. (2005). Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *IEEE international conference on robotics and automation* (vol. 4, pp. 3716–3721). New York: IEEE.

Sinitsyn, A. V. (1993). Construction of the value function in a game of approach with several pursuers. *Journal of Applied Mathematics and Mechanics*, *57*(1), 59–65.

Tomlin, C., Pappas, G., & Sastry, S. (1998). Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, *43*, 509–521.

Van Den Berg, J., Ferguson, D., & Kuffner, J. (2006). Anytime path planning and replanning in dynamic environments. In *IEEE international conference on robotics and automation (ICRA)* (pp. 2366–2371).

Van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE international conference on robotics and automation (ICRA)* (pp. 1928–1935).

Vatcha, R., & Xiao, J. (2009). Perceiving guaranteed continuously collision-free robot trajectories in an unknown and unpredictable environment. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1433–1438). New York: IEEE.

Vendittelli, M., Laumond, J., & Nissoux, C. (1999). Obstacle distance for car-like robots. *IEEE Transactions on Robotics and Automation*, *15*(4), 678–691.

Wilkie, D., van den Berg, J., & Manocha, D. (2009). Generalized velocity obstacles. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5573–5578). New York: IEEE.

Wu, A. (2011). *Guaranteed avoidance of unpredictable, dynamically constrained obstacles using velocity obstacle sets*. Master's thesis, MIT Department of Aeronautics and Astronautics, May 2011. Available online at http://acl.mit.edu/papers/WuSM.pdf.

Albert Wu received the B.S. degree from the California Institute of Technology in mechanical engineering in 2009 and the S.M. degree from MIT in Aeronautics and Astronautics in 2011. He is currently pursuing the Ph.D. Degree in Robotics at CMU. His research interests are in the area of multi-vehicle planning.

Jonathan P. How is the Richard Cockburn Maclaurin Professor of Aeronautics and Astronautics at the Massachusetts Institute of Technology. He received a B.A.Sc. from the University of Toronto in 1987 and his S.M. and Ph.D. in Aeronautics and Astronautics from MIT in 1990 and 1993, respectively. He then studied for two years at MIT as a postdoctoral associate for the Middeck Active Control Experiment (MACE) that flew on-board the Space Shuttle Endeavour in March 1995. Prior to joining MIT in 2000, he was an Assistant Professor in the Department of Aeronautics and Astronautics at Stanford University. Research interests include: Design and implementation of distributed robust planning algorithms to coordinate multiple autonomous vehicles in dynamic uncertain environments; Spacecraft navigation, control, and autonomy, including GPS sensing for formation-flying vehicles; and Adaptive flight control to enable autonomous agile flight and aerobatics. He was the planning and control lead for the MIT DARPA Urban Challenge team that placed fourth in the recent race at Victorville, CA. He was the recipient of the 2002 Institute of Navigation Burka Award, a recipient of a Boeing Special Invention award in 2008, is an Associate Fellow of AIAA, and a senior member of IEEE.