Generalized Velocity Obstacles

David Wilkie Jur van den Berg Dinesh Manocha

Abstract— **We address the problem of real-time navigation in dynamic environments for car-like robots. Our approach generalizes the concept of *velocity obstacles*, which have been used for navigation among dynamic obstacles, to take into account the constraints of a car-like robot. We present generalized velocity obstacles, which identify actions that will lead to a collision with a moving obstacle at some point in the future. We discuss using this concept to find actions that will allow collision free navigation in dynamic environments as well as the probabilistic completeness of our algorithm in finding a feasible action and in finding an action of some specified distance from a preferred action. Finally, we demonstrate the performance of our algorithm on a simulated car-like robot among moving obstacles.**

I. Introduction

The problem of computing a collision-free path for a robot moving among dynamic obstacles is important in many robotics applications, including automated transportation systems, automated factories, and applications involving robot-human interactions, such as a robotic wheelchair described in [19]. In many of these applications, the robot of interest is a car-like robot and subject to kinematic constraints.

In this paper, we extend the velocity obstacle concept. This concept was introduced by Fiorini and Shiller in [3] for navigating robots among arbitrarily moving obstacles and has been extended for multi-agent navigation and real-world applications. The velocity obstacle formulation works well for robots that can move in any specified direction, but it may not capture the movement of car-like robots well, which can only move, at any instant, with a velocity parallel to the rear wheels.

In this paper, we present an extension of the velocity obstacle concept to handle robots with kinematic constraints. We present a new algebraic formulation of the velocity obstacle in terms of the set of actions that can, at some point in the future, result in a collision. This generalized velocity obstacle is used to safely navigate a car-like robot among dynamic obstacles.

Our approach focuses on locally avoiding moving obstacles. The intended use is within *sense-plan-act* iterations, and it could be incorporated in a global planner or a *partial motion planner*, a concept introduced in [17] to compute global collision-free paths incrementally.

The rest of this paper is organized as follows. Section II discusses the related work, with an emphasis on approaches to navigation that could be applicable to car-like robots. Section III provides background information on the concept of velocity obstacles and the issues in using them for car-like robots. Section IV introduces the idea of generalized velocity obstacles. In Section V, we apply generalized velocity obstacles to a car-like robot moving among dynamic obstacles, and highlight the algorithm's performance. Section VI presents simulated scenarios of a car-like robot avoiding arbitrarily moving obstacles using our approach.

Department of Computer Science, College of Arts and Sciences, University of North Carolina, 201 South Columbia Street, Chapel Hill, NC 27599, USA. E-mail: {wilkie, berg, dm}@cs.unc.edu

For further project details, code, and videos, refer to http://gamma.cs.unc.edu/NHRVO.

II. Prior Work

In this section, we give a brief overview of related work on navigating agents and robots in dynamic environments.

Numerous motion planning algorithms have been developed for car-like robots in static environments. In [12], the notion of a *car-like robot* was formalized, and the fact that a path for a holonomic robot lying wholly in open regions of the configuration space can always be transformed into a feasible path for a nonholonomic robot was proven. [12] also provided an algorithm to generate a feasible path for a nonholonomic robot from a path found for a holonomic robot. Smooth planning for car-like robots among obstacles was first proposed in [21], and this idea was extended in [10], which uses a steering function rather than computing clothoid curves.

Approaches applicable to car-like robots have been developed for complete trajectory planning among moving obstacles, such as [7], [5], [26]. Some of these approaches decompose the problem of planning in a dynamic environment into the generation of a feasible path and planning a velocity profile to safely traverse the path, is in [8], [16], [25].

An alternative to complete planning is to plan for the robot as it acts, taking new sensor input as it arrives and planning locally. A major example of this approach uses *velocity obstacles* (VO), described in [3], [2]. The idea is to define a set of velocities that would, if used as a action for the agent, lead to a collision with an obstacle at some point in the future. In its original formulation, VO applied to agents moving along piecewise linear velocities and assumed the obstacles would be moving at constant velocities over a time interval. The idea was extended in [22], which used the idea of a *non-linear velocity obstacle* to define a VO for an obstacle moving along an arbitrary trajectory, which needs to be known in advance. In [11], the authors address the problem of predicting the motion of obstacles and apply the result to a car-like robot, but they still model the agent velocity as a piecewise linear function. Another extension of VO appeared in [24], which addressed the oscillation issue

Preprint submitted to 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 7, 2009.

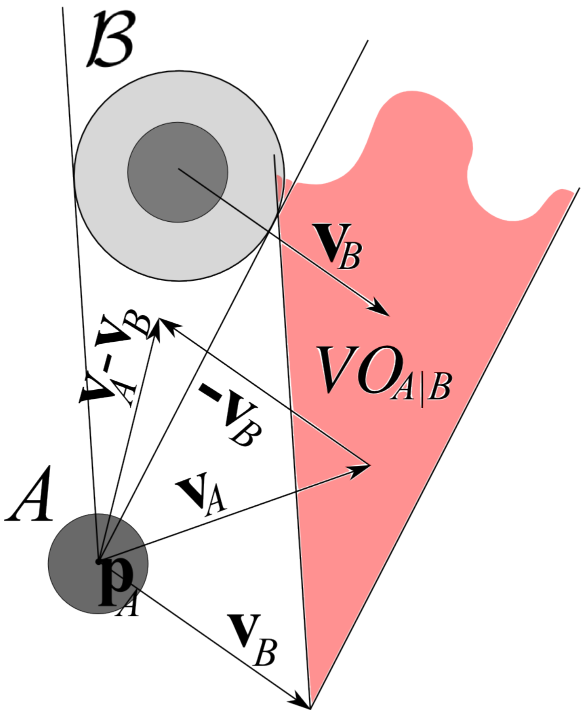

Fig. 1. The velocity obstacle $VO_{A|B}$ for robot A relative to dynamic obstacle B. The cone of velocities that would lead to a collision in the static case is translated by $\mathbf{v_B}$. Note that as the velocity $\mathbf{v_A}$ is inside the velocity obstacle, the relative velocity, $\mathbf{v_A} - \mathbf{v_B}$, leads to a collision.

that occurs in the VO formulation. The VO concept has been applied to control real robots, such as in [19], [18].

An analytical approach to finding a trajectory for a non-holonomic robot moving among dynamic obstacles is given in [20], and this approach was extended to include an rapidly-exploring random tree in [14]. Another approach to navigating non-holonomic agents in dynamic environments is presented in [15].

Some iterative approaches to planning in dynamic environments exist, such as [4], [11], [1], that seek to balance this local planning strategy with reaching a goal.

Finally, another approach to navigation among dynamic obstacles is to create potential fields for obstacles, as was done in [9], [23].

III. Velocity Obstacles

In this section, we review the velocity obstacle concept and discuss its limitations in applications to car-like robots.

A. Definition

For a disc-shaped agent A and a disc-shaped moving obstacle B with radii r_A and r_B, respectively, the velocity obstacle for A induced by B, denoted $VO_{A|B}$, is the set of velocities for A that would, at some point in the future, result in a collision with B. This set is defined geometrically. First, let $\mathbf{p}_A$ and $\mathbf{p}_B$ be the center point of A and B, respectively, and let $\mathcal{B}$ be a disc centered at $\mathbf{p}_B$ with a radius equal to the sum of A's and B's. This is generalized as a Minkowski sum. If B is static (i.e. not moving), we could define a cone, $\mathcal{C}$, of velocities for A that would lead to a collision with B as the set of rays shot from $\mathbf{p}_A$ that intersect the boundary of $\mathcal{B}$. To derive a velocity obstacle from this, we simply translate cone $\mathcal{C}$ by the velocity $\mathbf{v}_B$ of B, as shown in Figure 1. More formally:

$$VO_{A|B} = \{\mathbf{v} \,|\, \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - \mathbf{v}_B) \in \mathcal{B}\}. \quad (1)$$

B. Robots with Kinematic Constraints

For many kinematically constrained robots, such as car-like robots, the set of feasible velocities at any instant is one – the velocity in the direction of the rear wheels. One way to workaround this constraint and use velocity obstacles is to use the set of velocities that can be achieved over some time interval τ [3]. However, this approach does not guarantee collision-free navigation: consider the set of velocities, V, a car-like robot can reach after τ seconds. VOs can be constructed for the robot, and a velocity from V outside the VOs can be selected to navigate the robot. However, the robot will actually need to follow an arc to achieve the selected velocity, and the VOs provide no guarantee that this arc is collision-free. Additionally, the robot will be at a different position when it can move at the selected velocity, but the VOs are position sensitive, so the velocity is no longer guaranteed to be collision-free. To minimize these issues, we can select a small value for τ, but this has consequences for navigation. As τ decreases, the set of velocities that are being considered by the robot decreases, and the robot can miss feasible actions.

IV. Generalized Velocity Obstacles

In this section, we define a new concept called the *Generalized Velocity Obstacle*. This formulation is a generalization of the velocity obstacle concept and seeks to address the difficulty of using velocity obstacles with kinematically constrained agents, such as car-like robots.

A. Definition

Previous velocity obstacle formulations have been based on geometric relationships between an agent and linearly moving obstacles. While they allow for efficient navigation for agents capable of moving in an arbitrary direction at an instant, they may not be well suited for car-like robots.

In order to formulate a velocity obstacle for such robots, we derive a velocity obstacle algebraically, instead of geometrically. In general, given an obstacle B, let us denote the position it will have at time t by $B(t)$. Similarly, given the position of agent A at time $t = 0$ and an action u, let us denote the position agent A will have after undertaking u for time interval t by $A(t, u)$. We can now define the velocity obstacle generally as

$$VO_{A|B} = \{u \,|\, \exists t > 0 :: \|A(t, u) - B(t)\| < r_A + r_B\}. \quad (2)$$

We can derive an explicit formulation of $VO_{A|B}$ as follows. Let $t_{\min}(u)$ be the time at which the distance between the centers of A and B is minimal for a given action u for A:

$$t_{\min}(u) = \arg\min_{t>0} \|A(t, u) - B(t)\|. \quad (3)$$

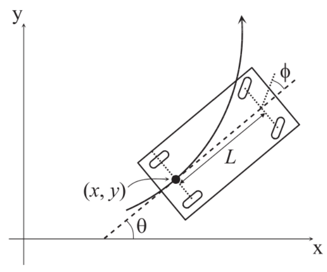

Fig. 2. The kinematic model of a simple car.

A closed expression for $t_{\min}(u)$ is obtained by solving $\frac{\partial \|A(t,u)-B(t)\|}{\partial t} = 0$ for t. It follows that an explicit equation for the velocity obstacle is found by solving $\|A(t_{\min}(u),u) - B(t_{\min}(u))\| < r_A + r_B$ for u.

B. Example: an Unconstrained Robot

Let us take as an example an agent without kinematic constraints for which the original velocity obstacle was defined. In this case, an action u for A directly corresponds to a velocity $\mathbf{v}$ for A. Let us assume, without loss of generality, that the initial position of A is at the origin. Let an obstacle B be at the initial position $\mathbf{p}_B$ and be moving with the velocity $\mathbf{v}_B$. Then we have

$$A(t,\mathbf{v}) = t\mathbf{v}, \quad (4)$$
$$B(t) = \mathbf{p}_B + t\mathbf{v}_B. \quad (5)$$

This gives the following equation for $t_{\min}(\mathbf{v})$:

$$t_{\min}(\mathbf{v}) = \frac{\mathbf{p}_B \cdot (\mathbf{v} - \mathbf{v}_B)}{\|\mathbf{v} - \mathbf{v}_B\|^2}. \quad (6)$$

Solving $\|A(t_{\min}(\mathbf{v}),\mathbf{v}) - B(t_{\min}(\mathbf{v}))\| < r_A + r_B$ for $\mathbf{v}$ then gives an explicit representation of the velocity obstacle.

This velocity obstacle is exactly identical to those derived in [3] and can be used for navigation in the same way as the geometrically derived velocity obstacle. A preferred action u^* can be computed based on a function of the current configuration and the goal configuration, and the action closest to the preferred but outside all velocity obstacles can be used to move the robot. The difference between the definitions of (1) and (2) is that we can easily generalize the latter to incorporate kinematic (and dynamic constraints), as we will show in Section V.

C. Finite Time Horizon

A modification of the velocity obstacle that is useful in practice is to limit the potential collisions to those that will happen before some arbitrary time, t_{lim}. For the traditional velocity obstacle formulation, this time horizon changes the velocity obstacle from a cone to a truncated cone with a rounded end, which can be approximated as in [6]. The time horizon can easily be incorporated in the above definitions: the clause "$t > 0$" must be replaced by "$t \in [0, t_{\text{lim}}]$" in (1), (2) and (3).

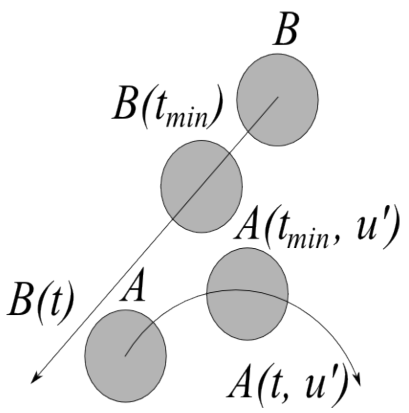

Fig. 3. Obstacle B is moving along trajectory $B(t)$, and agent A is trying to evade. A tests action u', which generates trajectory $A(t,u')$. The minimum distance between B and A occurs at t_{min}, and this distance is greater than the sum of the radii. Therefore, action u' is not in the velocity obstacle.

V. Navigating a Simple Car

A. Velocity Obstacles for Car Kinematics

Following the framework laid out above, we can formulate a velocity obstacle for a non-holonomic agent subject to the simple car kinematic constraints. Let (x, y) be the position of the robot and θ its orientation. Following [13], its kinematic constraints are given as

$$\begin{aligned} x'(t) &= u_s \cos\theta(t), \quad (7)\\ y'(t) &= u_s \sin\theta(t),\\ \theta'(t) &= u_s \frac{\tan u_\phi}{L}, \end{aligned}$$

where u_s and u_ϕ are the actions of the car, i.e. speed and steering angle, respectively, and L is the wheelbase of the car. This is shown in Figure 2.

By integrating the above equations (7), we can derive a formula for the position of a car at time t under the assumption that the actions of speed, u_s, and steering angle, u_ϕ, remain constant:

$$A(t,u) = \begin{pmatrix} \frac{1}{\tan(u_\phi)} \sin(u_s \tan(u_\phi)t) \\ -\frac{1}{\tan(u_\phi)} \cos(u_s \tan(u_\phi)t) + \frac{1}{\tan(u_\phi)} \end{pmatrix}. \quad (8)$$

This derivation assumes the car has unit wheelbase (i.e. $L = 1$). For the full derivation of these equations, please see the appendix. These positions are derived relative to the robot's frame of reference, in which the robot is at the origin and its orientation is along the positive x-axis. We will assume that there are an arbitrary number of obstacles B_i and that we can view them as moving linearly over a short time interval,

$$B_i(t) = \mathbf{p}_{B_i} + \mathbf{v}_{B_i} t \quad (9)$$

Like above, we proceed by finding an expression for the minimum distance between the robot A and a dynamic obstacle B given an action u for A. The derivative of the distance can be calculated, but this does not produce a simple analytical expression for $t_{\min}$. Therefore, we will solve for

Algorithm 1 Find best feasible action

```
for $i = 0$ to $n$ do
  $u \leftarrow$ sample actions from the set of all actions $U$
  $free \leftarrow$ true
  for all Moving Obstacles $B$ do
    let $D(t) =$ the distance between $A(t, u)$ and $B(t)$
    let $D'(t) =$ the derivative of $D(t)$
    $t_{\min} \leftarrow$ solve $D'(t) = 0$ for $t \in [0, t_{\lim}]$.
    $d \leftarrow D(t_{\min})$.
    if $d < r_A + r_B$ then
      $free \leftarrow$ false
    end if
  end for
  if $\|u - u^*\| < min$ then
    $min \leftarrow \|u - u^*\|$
    $argmin \leftarrow u$
  end if
end for
return $argmin$
```

$t_{\min}$ numerically for specific action and check whether the action is inside the velocity obstacle.

In Figure 3, we see an example of an action u' that is outside the velocity obstacle.

B. Approach

We use an optimization procedure to navigate the robot among multiple moving obstacles. Let u^* be the action the robot would select if no moving obstacles were around, for instance the action that would lead the robot most directly towards its goal. We refer to u^* as the *preferred* action.

The actual action u to give the robot is given by the solution to the problem

$$u = \underset{u' \notin \bigcup_{B_i} VO_{A|B_i}}{\arg\min} \|u^* - u'\|. \qquad (10)$$

That is, the problem of navigation among multiple dynamic obstacles can be formulated as the minimization of the distance between the optimal action, u^*, and a sampled action, u', where u' is subject to a velocity obstacle constraint for each moving obstacle. In some cases, it may be desirable to use a distance metric that favors some action variable over another in the criterion function. For example, steering in a specific direction may matter more than the speed.

An approach to solving this optimization procedure is given in Algorithm 1. At each discrete time step, t, we sample n actions. For each sampled action, we check, for each moving obstacle B_i, whether this action is inside the velocity obstacle induced by B_i. If the action is outside all of the velocity obstacles, then the action is feasible and is tested against the current best feasible action in terms of distance from the preferred action, u^*.

C. Analysis

We will now discuss the safety of navigation done using velocity obstacles, the probabilistic completeness of finding a collision-free action where one exists, and the probabilistic completeness of finding a collision-free action within some specified distance from the preferred action. In terms of safety, the general velocity obstacle concept is being analyzed. In terms of completeness, we analyze Algorithm 1.

1) Safety: As with traditional velocity obstacle navigation, the safety of the chosen action is bounded by the accuracy of the prediction of the obstacle's movement. In a sense, the agent is choosing an action based on how it guesses the obstacles will be moving in the future. If an action is chosen outside all velocity obstacles, then it is guaranteed to be safe as long as all the obstacles remain moving *as modeled*. But if an obstacle starts moving differently than predicted, there is no safety guarantee for the previously chosen action.

Theorem 1. *An action chosen outside of all velocity obstacles will be safe as long as each obstacle continues to move as modeled.*

Proof: The velocity obstacle for obstacle B consists of all the actions that will lead to a collision at some point in the future, as long as the motion model for B remains accurate. Any u not in a B's velocity obstacle will then never lead to a collision with B in the future, as long as the motion model for B remains accurate. If the chosen action is outside all the velocity obstacles, it is then not an action that will lead to a collision with some obstacle at some point in the future. Therefore, it will never lead to a collision, as long as all obstacles continue to move as modeled.

2) Probabilistic Completeness of Evasion: Our approach is for local navigation, not global planning. However, we can still talk about completeness in terms of the algorithm finding a collision-free action whenever one exists. In the following theorem, let $\mu(\mathcal{U})$ be the volume of the bounded action space and α be a real number $\in (0, 1]$.

Theorem 2. *If the volume of the feasible set of actions is at least $\alpha\mu(\mathcal{U})$, then the probability of finding no safe action when one exists converges exponentially to zero in the number of samples, n.*

Proof: If the volume of the feasible set is at least $\alpha\mu(\mathcal{U})$, then the probability of sampling an action in the feasible set is at least α. After n samples, the probability that no sample has been drawn from the feasible set is at most $(1 - \alpha)^n \leq e^{-n\alpha}$.

3) Probabilistic Completeness of Near Optimal Navigation: This theorem addresses the likelihood of finding an action that is close to the optimal action, u^*. In this theorem, γ is a real number $\in (0, 1]$.

Theorem 3. *If there exists a subset of actions, O, such that for all $u \in O$, $\|u^* - u\| \leq \beta$, and if O has a volume of at least $\gamma\mu(\mathcal{U})$, then the probability of only finding actions farther than β from the preferred action converges to zero exponentially in n, the number of samples.*

Proof: If the volume of O is at least $\gamma\mu(\mathcal{U})$, then the probability of sampling an action in O is at least γ. The probability of not finding an action in O after n samples is

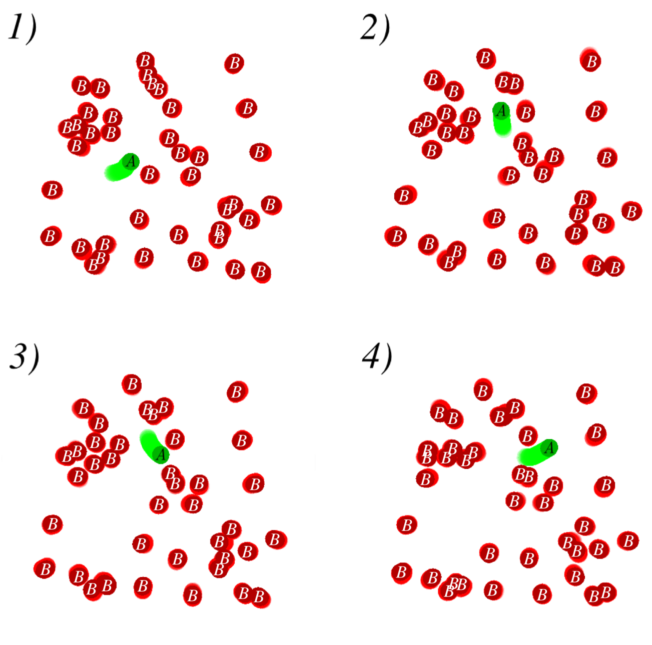

Fig. 4. Agent A needs to make its way through a crowd of obstacles, the Bs, that are each undertaking a random walk. In frame 1, A moves forward and turns left. In frame 2, A is blocked from going further in its desired direction. In frame 3, A reverses and waits by a narrow gap. And in frame 4, A clears the gap at an opportune moment and continues on.

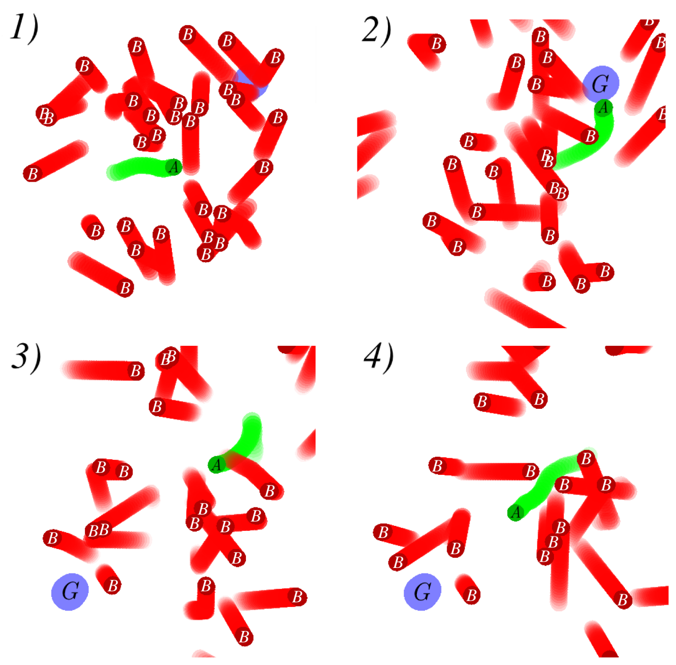

Fig. 5. Agent A takes collision-free actions through a crowd of quickly moving obstacles, the Bs, to the goal G. As A reaches the goal, the goal switches, and A must head back through the obstacle crowd.

then at most $(1-\gamma)^n \leq e^{-n\gamma}$.

In both of the above probabilistic completeness theorems, it is important to note that the existence and speed of the convergence to zero depends on the existence and size of the specified volumes in the action space. As the sizes of these volumes decrease, the speed of the convergence can become arbitrarily slow.

VI. Experiments

In this section, we present some scenarios of simulated car-like robots navigating through the use of our approach. In all figures for this section, the green circle represents the robot for whom we are navigating, the red circles are obstacles (that are allowed to collide with each other), and the trails simply indicate the recent path taken by the robot or obstacle. In all situations, the agent has the same bound on its velocity that the obstacles have: $(1.5, 1.5)$ to $(-1.5, -1.5)$. The minimal turning radius is one, and the radii of the agent and obstacles are one.

We also bias the sampling in these scenarios by explicitly adding four maneuvers to be tested. These maneuvers are moving the robot at the maximum forward speed along a circle of minimal radius (to the left and to the right) and moving the robot at the maximum reverse speed in the same manner. These maneuvers are added as emergency escape maneuvers.

A. Service Robot

In this scenario, shown in Figure 4, the robot A needs to move through a crowd of obstacles. A task such as this could be imagined for a robot delivering food or drinks. Each obstacle is undertaking a random walk, with its velocity being chosen every timestep uniformly between $(-1.5, -1.5)$ and $(1.5, 1.5)$. The Figure 4 shows agent A successfully moving through the crowd. Environments such as this are particularly difficult to move safely in as the safety guarantee no longer holds after an obstacle changes direction, which these obstacles are doing constantly.

B. Goal Directed Movement

In this scenario, shown in Figure 5, the robot needs to reach a goal region, the blue circle, while avoiding all the red obstacles. The obstacles change their direction with a low probability every time interval. This creates a nice mix of random motion and straight line trajectories. Once the agent, A, has reached a goal, G, the goal switches and the agent makes toward the new goal. Milestones of a global planner could by utilized in a similar manner.

VII. Conclusion

In this paper, we have presented a generalization of the velocity obstacle concept that can be used to navigate robots among multiple, unknown, dynamic obstacles. This generalization allows us to create a velocity obstacle in the action space for a car-like robot. We have also presented an algorithm to navigate a robot based on the generalized velocity obstacle. We discussed the safety of the actions chosen by this procedure and the convergence of the algorithm in finding a collision-free action and in finding an action of a specified quality. The algorithm presented allows for fast navigation for car-like robots among dozens of arbitrarily moving obstacles and allows the robots to test their entire action space for a feasible action, rather than a constrained subset.

While effective, this approach has some limitations. First, the current formulation for car-like robots does not include a constraint on the steering angle velocity. Thus, the paths generated are only piecewise smooth. This could cause problems for real car-like robots as the steering angle cannot be changed instantaneously. Second, the method uses a numerical means to calculate the minimum distance. The inaccuracy in the computation of the minimum time implies that a small safety buffer is required around each obstacle. And third, as this is a probabilistic algorithm, a feasible solution may not be found even if one exists.

This work has many natural extensions. Our formulation is general enough that it could be extended to consider dynamic constraints and constraints such as a maximum steering angle velocity. Incorporating these constraints would allow our method to generate smooth paths, rather than paths that are only piecewise smooth, and to handle fast moving robots. This work can also be extended to achieve multi-agent navigation.

References

[1] G. Chen and T. Fraichard. A real-time navigation architecture for automated vehicles in urban environments. In *2007 IEEE Intelligent Vehicles Symposium*, pages 1223–1228, 2007.

[2] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocityparadigm. In *1993 IEEE International Conference on Robotics and Automation, 1993. Proceedings.*, pages 560–565, 1993.

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

[4] T. Fraichard. Trajectory planning in a dynamic workspace: a'state-time space'approach. *Advanced Robotics*, 13(1):75–94, 1998.

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

[6] S. J. Guy, J. Chhugani, C. Kim, N. Satish, P. Dubey, M. Lin, and D. Manocha. Highly Parallel Collision Aviodance for Multi-Agent Simulation. Technical report, Department of Computer Science, University of North Carolina, 2009.

[7] D. Hsu, R. Kindel, J.C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233, 2002.

[8] K. Kant and S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72, 1986.

[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90, 1986.

[10] F. Lamiraux and J.P. Lammond. Smooth motion planning for car-like vehicles. *IEEE Transactions on Robotics and Automation*, 17(4):498–501, 2001.

[11] F. Large, D. Vasquez, T. Fraichard, and C. Laugier. Avoiding cars and pedestrians using velocity obstacles and motion prediction. In *IEEE Intelligent Vehicle Symposium*, pages 375–379, 2004.

[12] J.P. Laumond, PE Jacobs, M. Taix, and RM Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593, 1994.

[13] S.M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[14] K. Macek, M. Becker, and R. Siegwart. Motion planning for car-like vehicles in dynamic urban scenarios. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[15] E. Owen and L. Montano. Motion planning in dynamic environments using the velocity space. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005)*, pages 2833–2838, 2005.

[16] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research*, 24(4):295, 2005.

[17] S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005)*, pages 2210–2215, 2005.

[18] E. Prassler, J. Scholz, and P. Fiorini. Navigating a robotic wheelchair in a railway station during rush hour. *The international journal of robotics research*, 18(7):711, 1999.

[19] E. Prassler, J. Scholz, and P. Fiorini. A robotics wheelchair for crowded public environment. *IEEE Robotics & Automation Magazine*, 8(1):38–45, 2001.

[20] Z. Qu, J. Wang, and C.E. Plaisted. A New Analytical Solution to Mobile Robot Trajectory Generation in the Presence of Moving Obstacles. *IEEE Transactions on Robotics*, 20(6):978–993, 2004.

[21] A. Scheuer, T. Fraichard, and M. INRIA. Continuous-curvature path planning for car-like vehicles. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, 1997.

[22] Z. Shiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA*, volume 4, 2001.

[23] RB Tilove. Local obstacle avoidance for mobile robots based on the method ofartificial potentials. In *1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings.*, pages 566–571, 1990.

[24] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*, pages 1928–1935, 2008.

[25] J. van den Berg and M. Overmars. Kinodynamic motion planning on roadmaps in dynamic environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems-IROS*, pages 4253–4258, 2007.

[26] M. Zucker, J. Kuffner, and M. Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1603–1609, 2007.

Appendix

In this section, we derive of the generalized velocity obstacle for robots subject to the simple car kinematics. The car is modeled as a rigid body moving along a plane, and thus its configuration is (x, y, θ). In the car's frame of reference, the origin is at the midpoint of its rear axle and it is pointed in the positive x direction. In all of our derivations, we assume the wheelbase of the car is 1. To restate (7), the kinematic equations in this reference frame are given as

$$\begin{aligned} \frac{dx}{dt} &= u_s \cos\theta, \qquad (11) \\ \frac{dy}{dt} &= u_s \sin\theta, \\ \frac{d\theta}{dt} &= u_s \tan u_\phi, \end{aligned}$$

where u_s and u_ϕ are possible actions in the action space U. These are the speed and turning angle, respectively.

We can integrate as follows to derive our equation for $A(u, t)$. First, we integrate $\frac{d\theta}{dt}$,

$$\theta = u_s \tan(u_\phi) t. \qquad (12)$$

Next, we integrate $\frac{dx}{dt}$ by substituting $u_s \tan(u_\phi)t = \theta$,

$$\begin{aligned} \frac{dx}{dt} &= u_s \cos(\theta(t)), && (13) \\ &= u_s \cos(u_s \tan(u_\phi)t), && (14) \\ \int dx &= \int \frac{1}{\tan u_\phi} \cos(\theta) d\theta, && (15) \\ \text{thus,} x &= \frac{1}{\tan u_\phi} \sin(u_s \tan(u_\phi)t). && (16) \end{aligned}$$

We can make a similar argument for $\frac{dy}{dt}$, and we can determine from inspection that the integral adds the constant 1 to avoid a division by zero. This leads to the equation $y = \frac{1}{\tan u_\phi}(-\cos(u_s \tan(u_\phi)t) + 1)$. The velocity obstacle for the simple car and linearly moving obstacle uses the distance function,

$$D(u) = \sqrt{\left(\frac{\sin(u_s \tan(u_s)t)}{\tan u_\phi} - p_{B_x} - v_{B_x} t\right)^2 + \left(-\frac{\cos(u_s \tan(u_\phi)t)}{\tan u_\phi} + \frac{1}{\tan u_\phi} - p_{B_y} - v_{B_y} t\right)^2}, \tag{17}$$

and seeks to find its minimum by searching for a zero in the derivative of $D(u)^2$,

$$\begin{aligned} \frac{dD(u)^2}{dt} = 2\left(\frac{\sin(u_s \tan(u_\phi)t)}{\tan u_\phi} - p_{B_x} - v_{B_x} t\right) \\ * \left(\frac{\cos(u_s \tan(u_\phi)t) u_s \tan u_\phi}{\tan u_\phi} - v_{B_x}\right) \\ +2\left(-\frac{\cos(u_s \tan(u_\phi)t)}{\tan u_\phi} + \frac{1}{\tan u_\phi} - p_{B_y} - v_{B_y} t\right) \\ * \left(\frac{\sin(u_s \tan(u_\phi)t) u_s \tan u_\phi}{\tan u_\phi} - v_{B_y}\right). \end{aligned} \tag{18}$$

With these defined, we can proceed along the lines of Algorithm 1.