The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon

Zvi Shiller, Oren Gal, and Thierry Fraichard

***Abstract*— This paper addresses the issue of motion safety in dynamic environments using Velocity Obstacles. Specifically, we propose the minimum time horizon for the velocity obstacle to ensure that the boundary of the velocity obstacle approximates conservatively that boundary of the set of inevitable collision states. Thus, using the velocity obstacle to select potential avoidance maneuvers would ensure that only safe maneuvers are being selected. The computation of the minimum time horizon is formulated as a minimum time problem, which is solved numerically for each static or moving obstacle. The "safe" velocity obstacles are used in an on-line planner that generates near-time optimal trajectories to the goal. The planner is demonstrated for on-line motion planning in very crowded static and dynamic environments.**

I. Introduction

The main challenge in motion planning in dynamic environments is reaching the goal while selecting on-line among only safe avoidance maneuvers. While reaching the goal cannot be guaranteed with an on-line planner, one can reduce the state space search to only safe states, i.e. states from which at least one other safe state is reachable. Some local (reactive) planners exist [9], [21], [15], [16], but most do not guarantee safety as they are too slow and hence their ability to look-ahead and avoid states of inevitable collision [12] is very limited. Recently, iterative planners [10], [4], [1], [14], [6], [20] were developed that compute several steps at a time, subject to the available computation time, but those too do not address the issue of safety. A promising approach to safe motion planning in dynamic environment is the consideration of "regions of inevitable collision," first introduced in [18] and later extended in [11], [17], [12], [3].

We address the issue of safety for an on-line local planner in dynamic environments using velocity obstacles. Safety is guaranteed by ensuring that the robot's velocity does not penetrate the velocity obstacle, which is generated for a carefully selected time horizon. This time horizon, if properly selected, ensures that the boundary of the velocity obstacle conservatively approximates the boundary of the set of inevitable collision states. Repelling the robot's velocity from entering the inevitable collision states ensures (if a solution exists) that the robot does not crash into any static or moving obstacle. The computation of the safe time horizon, which is obstacle specific, is formulated as a minimum time problem that minimizes the time for the robot velocity to exit the velocity obstacle, subject to the robot dynamic constraints. The solution is shown to be along an extremal, generated on the boundary of the control constraints.

Determining the safe time horizon is computationally efficient and it does not require a prior mapping of inevitable collision states. Since the time horizon is obstacle specific, motion safety is guaranteed if obstacles can be avoided individually or if the state-space between the current position and the goal state stays connected. The velocity obstacles, truncated at the minimum time horizon, are used for an on-line planner that generates near-time optimal trajectories by minimizing at each time step the time-to-go to the goal. The planner is demonstrated for on-line motion planning in very crowded static and dynamic environments.

This work was performed at the Paslin Robotics Research Laboratory at the Ariel University Center, Israel.

Zvi Shiller is with the Department of Mechanical Engineering, Ariel University Center, Israel (shiller@ariel.ac.il)

Oren Gal is a graduate student at the Dept. of Mechanical Engineering, Technion, Israel.

Thierry Fraichard is with Inria Rhône-Alpes, Montbonnot, France.

II. The Velocity Obstacle

The velocity obstacle represents the set of all colliding velocities of the robot with each of the neighboring obstacles [7], [19], [13]. It maps static and moving obstacles into the robot's velocity space. The velocity obstacle of a planar obstacle, B, that is moving along a general known trajectory, $c(t)$, is a warped cone in the velocity space of the point robot A. It is called a nonlinear velocity obstacle (*NLVO*) since it accounts for a general (nonlinear) trajectory of the obstacle. Selecting a *single* velocity at time $t = t_0$ outside the *NLVO* guarantees no collision at all times, as long as the obstacle stays on its current trajectory.

The non-linear v-obstacle is constructed as a union of its temporal elements, $NLVO(t)$, which is the set of all absolute velocities of A, v_a, that would result in collision at a specific time t. Referring to Figure 1, v_a that would result in collision with point $p \in B(t)$ at time $t > t_0$, expressed in a frame centered at $A(t_0)$, is simply

$$v_a = \frac{c(t)+r}{t-t_0}, \tag{1}$$

where r is the vector to point p in the obstacle's fixed frame. It is a homothety transformation [5], centered at $A(t_0)$ and scaled by $k = \frac{1}{t-t_0}$:

$$v_a = H_{A,k}(c(t)+r); k = \frac{1}{t-t_0}. \tag{2}$$

The set, $NLVO(t)$ of all absolute velocities of A that would result in collision with any point in $B(t)$ at time $t > t_0$ is thus:

$$NLVO(t) = H_{A,k}(B(t)); k = \frac{1}{t-t_0}. \tag{3}$$

Integrating (3) over $t = [t_0, \infty)$ yields the non-linear v-obstacle, $NLVO^{\infty}_{t_0}$, representing the set of all linear velocities

of A that would collide with $B(t)$ at time $t = (t_0, \infty)$:

$$NLVO_{t_0}^{\infty} = \bigcup_t H_{A,k}(B(t)); k = \frac{1}{t - t_0}; t = (t_0, \infty). \quad (4)$$

The non-linear v-obstacle is a warped cone, as shown schematically in Figure 2. If $c(t)$ is bounded over $t = (t_0, \infty)$, then the apex of this cone is at $A(t_0)$. The boundaries of the *NLVO* represent velocities that would result in A grazing B. The smallest safe time horizon is the one that allows sufficient time to avoid or escape collision as discussed next.

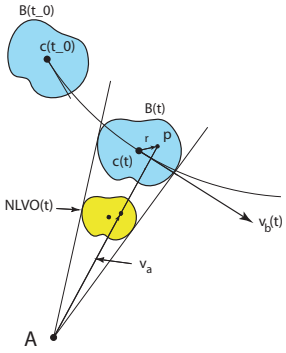

Fig. 1. A temporal element of the non-linear v-obstacle.

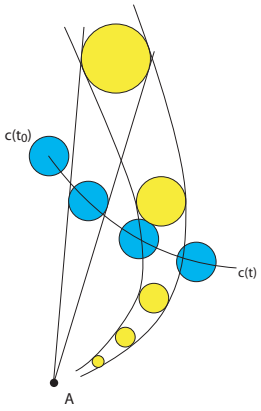

Fig. 2. A non-linear velocity obstacle

III. Time Horizon

The nonlinear velocity obstacle consists of the union of temporal velocity obstacles from time t_0 to infinity. It thus accounts for colliding velocities that would result in collisions at all times. To account for collisions that occur only within a specified time horizon, t_h, the *NLVO* is truncated by setting a time limit in (4):

$$NLVO_{t_0}^{t_h} = \bigcup_t H_{A,k}(B(t)); k = \frac{1}{t - t_0}; t = (t_0, t_h). \quad (5)$$

Truncating *NLVO* to reflect specific time limits allows to focus the motion planning problem on potential collisions occurring within a specified time interval, such as imminent collisions occurring within some given time horizon. This permits avoidance maneuvers that are potentially risky, but at times that are practically insignificant at the decision time t_0. Setting the time horizon too high would be too prohibitive, as it would mark as dangerous maneuvers that might result in collision at a distant time; selecting it too small would permit dangerous maneuvers that are too close and at too high speeds to avoid the obstacle. We wish to select the smallest time horizon that would mark as dangerous only those velocities at which the obstacle is unavoidable given robot dynamics and its dynamic and kinematic constraints. In other words, the velocity obstacle that is truncated by the smallest time horizon should consist only of velocities that belong to the inevitable collisions states (ICS). Conversely, any velocity that does not penetrate this velocity obstacle should allow sufficient time under the given control authority to avoid collision. The smallest safe time horizon thus allows sufficient time for the robot to avoid the obstacle, or equivalently, bring its velocity vector outside the velocity obstacle.

We define the smallest safe time horizon as the minimum time to exit the velocity obstacle from a given state, subject to robot dynamics and actuator constraints. It is the solution of the minimization problem:

$$min \int_{t_0}^{t_h} 1 dt, \quad (6)$$

with the initial condition

$$x(t_0), \dot{x}(t_0) \quad (7)$$

the final condition

$$\dot{x}(t_h) \notin NLVO_{th}^{\infty} \quad (8)$$

satisfying system dynamics

$$\ddot{x} = f(x, \dot{x}, u) \quad (9)$$

and control constraints

$$u \in U, \quad (10)$$

where $NLVO_{th}^{\infty}$ denotes a nonlinear velocity obstacle generated at time t_h for an infinite time horizon.

It is easy to show that the solution, t_h, to problem (6) is bang-bang, or obtained by an extremal trajectory that is generated on the boundary of the control constraints [2]. For a point mass model, there are four such trajectories, generated by the four corners of the set of admissible control U. Figure 3 shows a point robot A, a static obstacle B, and the initial velocity v_a that is pointing towards B. Obviously, v_a is inside the velocity obstacle of B, or $v_a \in VO_B$. Applying the controls associated with the four corners of the set of admissible control U results in four extremal trajectory denoted $x_1 - x_4$. We wish to compute the minimum time at which the velocity along at least one of the extremals points outside of the velocity obstacle, or equivalently, the velocity v_a is tangent to B.

For the case shown, three of the four extremals penetrate the obstacle and one does not. The minimum time t_h that is the solution to problem (6) is the time where the tangent line at $x_1(t_h)$ is tangent to B, as shown in Figure 3. The case

shown in Figure 3 is for a static obstacle. The analysis for a moving obstacle is similar except that v_a is replaced with the relative velocity $v_{a/b}$ and v_b is subtracted from the extremals.

The smallest time horizon t_h is obstacle specific, and it depends on the size of the obstacle, its velocity, the robot's velocity and its dynamic constraints. Using the smallest time horizon allows to detect if the robot approaches an inevitable collision state (by determining how far is v_a from the velocity obstacle) and to plan a safe avoidance maneuver. It is important to note that by integrating the extremals to compute t_h, one determines if the current state is safe or not without resorting to the velocity obstacles. However, knowing that a specific state may lead to a collision does not help in selecting a safe maneuver. This is where the velocity obstacles, generated for the smallest time horizon, become most useful.

The smallest time horizon can be easily computed by expressing all four extremal trajectories analytically and searching for the nearest (in time) tangency point of the common tangent between B and every extremal.

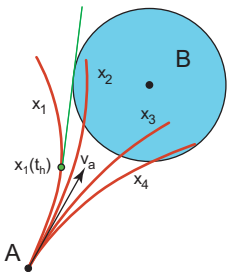

Fig. 3. The robot and obstacle on a collision course

Figure 4 shows a static obstacle and the trajectory generated by the on-line planner discussed later. The robot starts from rest at point $(0,-4)$ at the bottom, and moves in near-minimum time to the goal at $(0,2)$. The minimum time horizon along this path is shown in Figure 5. It grows from zero since the robot starts from rest, then it decreases as the robot moves tangentially to the obstacle. The time horizon is zero at the tangency point since the robot velocity is not pointing towards the obstacle. For the same reason, the time horizon is zero when the robot moves away from the obstacle.

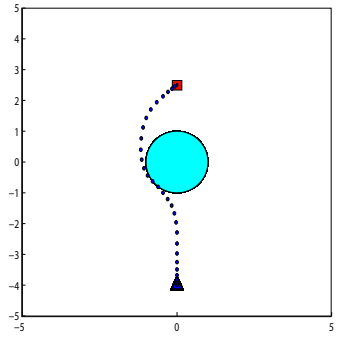

Fig. 4. Dynamic avoidance of a static obstacle

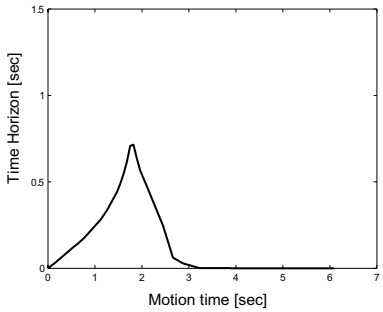

Fig. 5. The minimum time horizons along the path

In addition to focusing the avoidance on imminent collisions, the time horizon allows the consideration of large static obstacles, such as surrounding walls and highway barriers, whose v-obstacles, without a time horizon, would cover the entire velocity space. Consider a point robot A at the center of a closed room, as shown on the left in Figure 6. The velocity obstacle of the room's walls with an infinite time horizon covers the entire space, except the origin A, since any velocity other than zero would eventually result in collision with the walls. The boundary of the velocity obstacle of each wall is determined by computing the velocity v_h that would traverse the distance d to the wall in some set time horizon t_h:

$$v_h = \frac{d}{t_h}. \tag{11}$$

Any velocity larger than v_h would penetrate the velocity obstacle since it would result in collision at a shorter time than the time horizon t_h, whereas any velocity smaller than v_h is safe for the duration $t \leq t_h$. The inner boundary of the velocity obstacle of the closed room is a scaled shape of the room's walls, as shown on the right in Figure 6; the shorter the time horizon, the larger the "free" space.

In the case of a closed space, the smallest safe time horizon is the minimum time to reach zero speed, since zero speed is the only velocity outside the infinite time velocity obstacle of the closed room. It can be approximated by computing the minimum stopping time t_s in the direction of motion, which is simply (for a point mass robot):

$$t_s = \frac{v}{u_{min}}, \tag{12}$$

where v is the robots current speed, and u_min is its maximum deceleration.

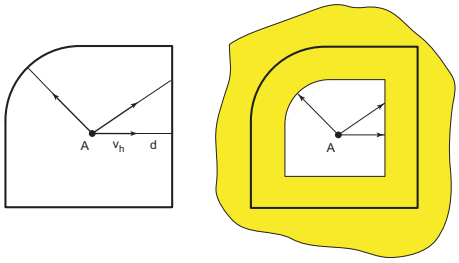

Fig. 6. The velocity obstacle of static walls and boundaries

IV. The Planner

The efficient representation of static and moving obstacles by velocity obstacles allows us to efficiently plan safe trajectories in dynamic environments. We assume knowledge of the positions and velocities of the neighboring obstacles.

We distinguish between local and global planners. The local planner generates one, or a few steps at every time step, whereas the global planner uses a global search to the goal over a time spanned tree. The local planner cannot ensure convergence to the goal and in some cases may lead to inevitable collision states [17]. The global planner, on the other hand, is complete, i.e. capable of finding a solution if one exists. Our planner is local as it generates one move at every time step.

The proper choice of the time horizon ensures survival of the robot, i.e. not entering inevitable collision states (*ICS*). For one obstacle, this guarantees convergence to the goal. For many obstacles, a solution cannot be guaranteed due to the changing nature of the environment: it is possible that during the local search, the state space around the robot becomes disconnected from the goal even though the global search might escape such a trap. The off-line planner computes a solution by exploring a tree of attainable states from the start node until the goal node is reached. The tree can be expanded using any efficient heuristics, such as a depth-first search or A*. This search can be drastically reduced by considering only "safe" attainable states that satisfy system dynamics and are out of the *ICS*. This planner is new in its on-line minimization of the time-to-go, combined with the use of velocity obstacles and the minimum time horizon to guide the tree search.

A. System Dynamics

The planner was implemented for a simple planar point mass model. This is necessary for computational reasons, and is in no way a limitation of this approach.

We consider the following point mass model:

$$\ddot{x} = u_1; |u_1| \leq 1 \tag{13}$$

$$\ddot{y} = u_2; |u_2| \leq 1 \tag{14}$$

where $(x,y)^T \in R^2$ represents the robot's position in a Cartesian coordinate frame and $(u_1,u_2)^T \in R^2$ represents the robot's controls.

Since the planning is done in the velocity space, we wish to compute the set of attainable Cartesian velocities (*ACV*) of the maneuvering robot that can be reached over a given time interval, Δt [8]. This set contains the avoidance maneuvers that are dynamically feasible from a given state. Denoting $v(t) = (\dot{x}(t), \dot{y}(t))^T$, the set of attainable *Cartesian* velocities, $ACV(t)$ is obtained by integrating the admissible controls $u = (u_1,u_2) \in U$ from the current state $(x,y,\dot{x},\dot{y})$:

$$ACV(t) = \{av | av = v(t) + u\Delta t, u \in U\}. \tag{15}$$

Each element in $ACV(t)$ represents an attainable velocity measured from the origin A. The geometric shape of $ACV(t)$ depends on the specific system dynamics. For a point mass model, with constant control constraints, it spans a rectangle around $v(t)$ that is proportional to the shape of the set of admissible controls U, as shown in Figure 7. The size of this rectangle depends on the time step Δt.

It is assumed that all velocities in $ACV(t)$ are reached at time t, when in fact they are reached at time $t + \Delta t$. This discrepancy introduces a small error that can be easily quantified.

Figure 8 shows the actual path followed by the robot from some initial velocity $v_a(t)$ under control $u = (1,-1)$ over the time interval Δt. The actual path is compared to the assumed straight line path, also shown in Figure 8. It is easy to show that the deviation between the two paths is bounded by the distance e between the end points:

$$e = \frac{\Delta t^2}{2}\sqrt{u_x^2 + u_y^2}. \tag{16}$$

This error can be made negligible by selecting Δt arbitrarily small. It is possible to account for this error by enlarging the obstacles by e. This error does not accumulate since the actual positions of robot and obstacles are used at each time step to generate the current velocity obstacles.

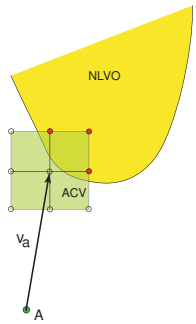

Fig. 7. Attainable Cartesian Velocities

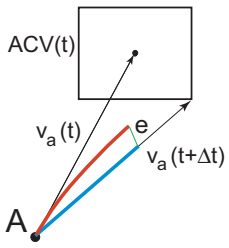

Fig. 8. Actual and assumed path over Δt.

B. Tree Search

The planner uses a depth first A* search over a tree that expands over time to the goal. Each node contains the current position and velocity of the robot at the current time step. At each state, the planner computes the set of admissible velocities *ACV*, which is then tessellated by a uniform grid, as shown in Figure 7. To test the safety of the nodes on the grid, a set of temporal velocity obstacles $NLVO(t), t \in (0,t_h)$ are computed at specified time intervals (the temporal velocity obstacles are computed in reverse order, starting

from $t = t_h$). In Figure 7, only $NLVO(t_h)$ is shown, where nodes inside $NLVO(t_h)$, marked red, are inadmissible. Nodes out of $NLVO(t_h)$ are further evaluated by computing from each the unconstrained (no obstacles) minimum time-to-go [13]. The node with the lowest time is then explored to the next time step. This is repeated until reaching the goal. For one obstacle, this planner is guaranteed to reach the goal in the near minimum time. For many moving obstacles, it may not, and a global search may be required.

V. Examples

The on-line planner was implemented and tested for crowded static and dynamic environments. Figure 9, shows four snapshots of the robot avoiding 70 moving obstacles. It starts from the bottom center and moves to the target at the top right. Despite the very challenging environment, the robot succeeds in reaching the goal while planning only one step at a time. This is largely due to the use of the minimum time horizon to truncate the velocity obstacles. This in turn drastically reduced the number of open nodes compared to the global search. Typical reductions have been between 0.1 to 0.2 of the number of nodes for the global search, a significant reduction that does not compromise safety. A video clip of the full run is available in www.ariel.ac.il/me/pf/shiller/oren.

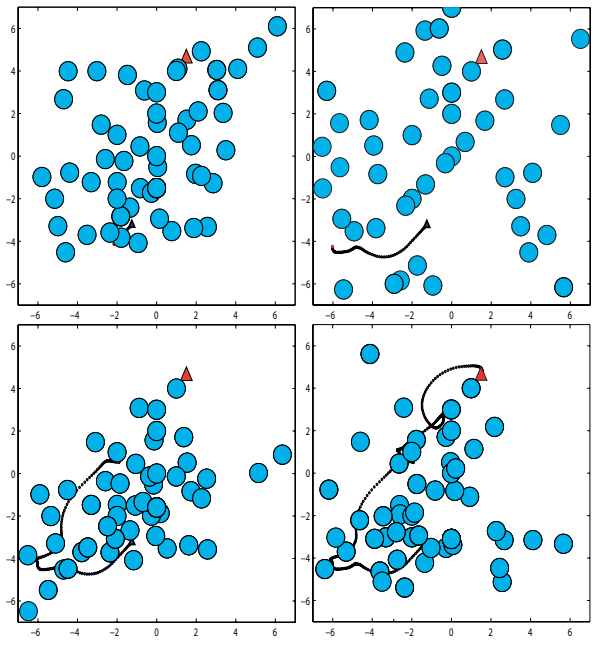

Fig. 9. Avoiding 70 moving obstacles

VI. Conclusions

A new time horizon for on-line planning in dynamic environments using velocity obstacles was presented. This time horizon is selected for each obstacle, static or moving, as the minimum time to bring the robot velocity outside the velocity obstacle from the current state. Keeping the robot's velocity vector out of the velocity obstacle ensures that the robot does not enter unsafe states from which avoidance cannot be guaranteed. Recognizing unsafe states using the velocity obstacles is not only safe but also very efficient as it drastically reduces the search tree. The planner generates near time-optimal trajectories, using the minimum time-to-go to guide the tree search. The planner was demonstrated for a point mass dynamic model. Other robot models can be used with minor modifications. The planner was successfully tested for crowded static and dynamic environments. It is suitable for real time generation of high speed trajectories in crowded static and dynamic environments.

References

[1] O. Brock and O. Khatib. Real time replanning in high- dimensional configuration spaces using sets of homotopic paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[2] A.E. Bryson and Y.C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., New York, NY, 1975.

[3] N. Chan and J. Kuffner M. Zucker. Improved motion planning speed and safety using region of inevitable collision. In *ROMANSY*, pages 103–114, July 2008.

[4] J-C. Latombe D. Hsu, R. Kindel and S. Rock. Randomized kino-dynamic motion planning with moving obstacles. *Algorithmics and Computational Robotics*, 4:247–264, 2000.

[5] C. W. Dodge. *Euclidean Geometry and Transformations*. Dover Publications, 2004.

[6] E. Feron and E. Frazzoli M.A. Daleh. Real time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance Control and Dynamics*, 25:116–129, 2002.

[7] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE International Conference of Automation and Robotics*, volume 1, pages 560–566, May 1993.

[8] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[9] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4:23–33, 1997.

[10] T. Fraichard. Planning in dynamic workspace: a state-time space approach. *Advanced Robotics*, 13:75–94, 1999.

[11] T. Fraichard. A short paper about safety. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[12] T. Fraichard and H. Asama. Inevitable collision state-a step towards safer robots? *Advanced Robotics*, 18:1001–1024, 2004.

[13] Shiller Z. Gal O. and Rimon E. Efficient safe motion planning in dynamic environments. In *IEEE Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[14] N. Simeon J. Minguez, L. Montano and R. Alami. Global nearest diagram navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[15] N.Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. In *International Conference on Intelligence Robots and Systems*, 1998.

[16] J. Minguez and L. Montano. Nearest diagram navigation. a new real-time collision avoidance approach. In *International Conference on Intelligence Robots and Systems*, 2000.

[17] S. Petti and T. Fraichard. Safe motion planning in dynamic environment. In *International Conference on Intelligence Robots and Systems*, 2005.

[18] J. Kuffner S. LaValle. Randomize kinodynamic planning. *International Journal of Robotics Research*, 20:378–400, 2001.

[19] F. Shiller, Z. Large and S. Sekhavat. Motion planning in dynamic environments: Obstacle moving along arbitrary trajectories. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2001.

[20] C. Stachniss and W.Burgard. An integrated approach to goal-directed obstacles avoidance under dynamic constrains for dynamic environment. In *International Conference on Intelligence Robots and Systems*, 2002.

[21] L. Ulrich and J. Borenstien. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.