2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Efficient and Safe On-Line Motion Planning in Dynamic Environments

Oren Gal, Zvi Shiller and Elon Rimon

Abstract— **This paper presents a new on-line planner for dynamic environments that is based on the concept of Velocity Obstacles (VO). It addresses the issue of motion safety, i.e. avoiding states of inevitable collision, by selecting a proper time horizon for the velocity obstacle. The proper choice of the time horizon ensures that the boundary of the velocity obstacle coincides with the boundary of the set of inevitable collision states. This time horizon is determined by the minimum time it would take the robot to avoid collision, either by stopping or by passing the respective obstacle. The planner generates a near-time optimal trajectory to the goal by selecting at each time step the velocity that minimizes the time-to-go and is out of the velocity obstacle. The planner takes into account the shape, velocity, and path curvature of the obstacle's trajectory. It is demonstrated for on-line motion planning in very crowded static and dynamic environments.**

I. INTRODUCTION

Most of the work on motion planning over the past twenty years has focused on static obstacles, with a few exceptions. 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. Examples of local (reactive) planners are [3], [17], [8], [11], but most do not guarantee safety as they are too slow and hence their ability to look-ahead and avoid states of inevitable collision is very limited. Recently, iterative planners [5], [7], [1], [12], [10], [15] were developed that compute several steps at a time, subject to the available computation time. The trajectory is generated incrementally by exploring a search-tree and choosing the best branch. These planners too do not address the issue of safety.

Only a few works have addressed the safety issue in dynamic environments, which is crucial for partial (local) planning. One approach is to use braking policies [18]; another is to ensure local avoidance for a limited time [10]. However, neither considers the dynamics of the moving robot. A promising approach to safe motion planning in dynamic environment is the consideration of "regions of inevitable collision," first introduced in [9] and later extended in [6], [14], [4], [2].

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 ensures that the boundary of the velocity obstacle overlaps 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 safe time horizon, which is obstacle specific, is determined by computing the minimum time it would take the robot to avoid collision, either by stopping or by passing the respective obstacle. 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.

In addition to addressing the safety issue, our planner generates near-time optimal trajectories by selecting at each time step a safe velocity that minimizes the time-to-go. The planner is demonstrated for on-line motion planning in very crowded static and dynamic environments.

II. THE VELOCITY OBSTACLE

The velocity obstacle represents the set of all colliding velocities of the robot with each of the neighboring obstacles. It maps static and moving obstacles into the robot's velocity space. The velocity obstacle (VO) of a planar circular obstacle, B, that is moving at a constant velocity v_b, is a cone in the velocity space of robot A, reduced to a point by enlarging respectively obstacle B, as shown in Figure 1. Each point in VO represents a velocity vector that originates at A. Any velocity of A that penetrates VO is a colliding velocity that would result in collision between A and B at some future time. Figure 1 shows two velocities of A: one that penetrates VO, and is hence a colliding velocity, and one that does not. All velocities of A that are outside of VO are safe as long as B stays on its current course. The velocity obstacle thus allows determining if a given velocity is potentially dangerous, and suggesting possible changes to this velocity to avert collision. If B is known to move along a curved trajectory or at varying speeds, it would be best represented by the nonlinear velocity obstacle discussed next.

The nonlinear velocity obstacle (NLVO) accounts for a general trajectory of the obstacle, while assuming a constant velocity of the robot. It applies to the scenario shown in Figure 2, where, at time t_0, a point A attempts to avoid an obstacle, B, that is following a general known trajectory, $c(t)$, and at time t_0 is located at $c(t_0)$. The $NLVO$ consists of all velocities of A at t_0 that would result in collision with

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

Oren Gal is a graduate student at the Department of Mechanical Engineering, Technion, Israel. (orengal@tx.technion.ac.il)

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

Elon Rimon is with the Department of Mechanical Engineering, Technion, Israel (rimon@technion.ac.il)

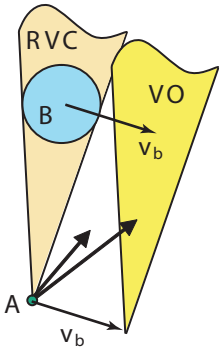

Fig. 1. A linear velocity obstacle

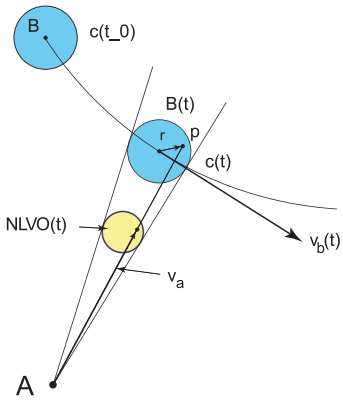

Fig. 2. A non-linear velocity obstacle

the obstacle at any time $t > t_0$. Selecting a single velocity, v_a, at time $t = t_0$ outside the $NLVO$ thus guarantees that A avoids collision at all times. It 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 2, v_a that would result in collision with point p in B 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. The set, $NLVO(t)$ of all absolute velocities of A that would result in collision with any point in B at time $t > t_0$ is obtained by replacing r with the set of all points in B:

$$NLVO(t) = \frac{c(t) \oplus \mathscr{B}}{t - t_0}. \tag{2}$$

where $\mathscr{B}$ represents the set of all points in the grown obstacle B, defined relative to some center that follows the curve $c(t)$, and $\oplus$ represents the Minkowski sum. Clearly, $NLVO(t)$ is a scaled $\mathscr{B}$, located at a distance from A that is inversely proportional to the collision time t. The entire $NLVO$ is the union of its temporal subsets from t_0, the current time, to some set time horizon t_h:

$$NLVO_{t_0}^{t_h} = \bigcup_{t_h > t > t_0} \frac{c(t) \oplus \mathscr{B}}{t - t_0}. \tag{3}$$

The non-linear v-obstacle is a warped cone. 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.

III. Time Horizon

The time horizon plays an important role in selecting feasible avoidance maneuvers. It allows considering only those maneuvers that would result in a collision within a specified time interval. Setting the time horizon too high would be too prohibitive, as it would mark as dangerous maneuvers resulting in collision at a distant time; selecting a too small time horizon would permit dangerous maneuvers that are too close and at too high speeds to avoid the obstacle. It is essential that the proper time horizon be selected to ensure that a safe maneuver, even if temporarily pointing toward the obstacle, is selected. We define a *safe state* as a state at which the robot can avoid the obstacle either by stopping or by passing. Obviously from any safe state there exists at least one *safe maneuver* (leading to another safe state). The smallest safe time horizon is the one that allows sufficient time for the robot to avoid the obstacle either by stopping or by passing. It depends on the size of the obstacle, its velocity, and the robot's dynamic constraints.

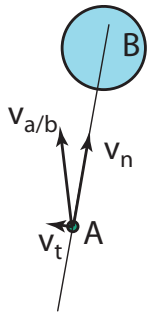

Fig. 3. The robot and obstacle on a collision course

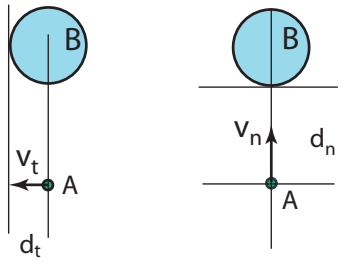

Fig. 4. Stopping and passing maneuvers

Consider a robot A and an obstacle B, each moving at some constant velocity. The time horizon is relevant only if the two are on a collision course, i.e. the velocity v_a penetrates the velocity obstacle of B. To determine the proper time horizon for this case, we first transform the problem into the avoidance of a static obstacle by considering the relative

velocity $v_{a/b}$, as shown in Figure 3. The relative velocity $v_{a/b}$ is then projected into two components, v_n and v_t that are parallel and normal to the line connecting A and the center of B, as shown in Figure 3.

The robot can avoid collision by either stopping before hitting the obstacle, or by passing it on either side. To stop, the robot's longitudinal velocity v_n must decelerate to zero before traversing the distance d_n, as shown in Figure 4; to pass, the robot must traverse the lateral distance d_t faster than it would traverse the longitudinal distance d_n. We select the time horizon such that when the robot's velocity first penetrates the velocity obstacle, it still has sufficient time to avoid collision either by stopping *or* by passing. To this end, we wish to determine the minimum time required for each maneuver (stopping and passing) to select the smallest safe time horizon. For simplicity, we decouple the two maneuvers, assuming that each is executed by a single control effort. The minimum times for the stopping and passing maneuvers thus depend on the initial velocity, distance, and the control constraint in each direction. The smallest safe time horizon is then the smallest of the minimum times for stopping and passing.

A. Stopping time

The minimum stopping time is the time it would take the robot to decelerate to a stop from its current normal velocity v_n, using the maximum deceleration. Assuming a constant longitudinal deceleration, $u_n < 0$, the stopping time is

$$t_{stop} = \frac{v_n}{-u_n}. \tag{4}$$

Since the VO assumes collision at a constant speed, whereas t_{stop} assumes a constant deceleration, using t_{stop} as the time horizon would alert the robot too early of a potential collision. Taking into account the decelerating velocity allows us to use a shorter time horizon. To determine how short, we compare the distance traveled over t_{stop} at a constant speed and at a constant deceleration.

The distance traveled at a constant velocity v_n over the stopping time t_{stop} is:

$$d_{const} = v_n t_{stop}. \tag{5}$$

The distance traveled at a constant deceleration u_n from the initial velocity v_n to a stop is:

$$d_{dec} = v_n t_{stop} + \frac{1}{2} u_n t_{stop}^2. \tag{6}$$

Substituting $v_n = -u_n t_{stop}$ into (6) yields:

$$d_{dec} = v_n t_{stop} - \frac{1}{2} v_n t_{stop} = \frac{1}{2} v_n t_{stop} = \frac{1}{2} d_{const}. \tag{7}$$

Since the distance traveled at a constant deceleration is half the distance traveled at a constant v_n over the stopping time t_{stop}, the moving robot should start decelerating when the time to collision at a constant speed drops to half the stopping time (4). The smallest time horizon t_s for the stopping maneuver is therefore half the stopping time t_{stop}:

$$t_s = \frac{1}{2} t_{stop} = \frac{v_n}{-2u_n}. \tag{8}$$

B. Passing time

The minimum time for passing, t_p, is the solution to the minimum time problem of traversing the distance d_t, given an initial velocity v_t and an unspecified final velocity. The solution to this problem is an extremal control that either accelerates or decelerates, depending on the signs of d_t and v_t.

The velocity v_f developed by accelerating at u_t over t_p until traversing d_t satisfies:

$$v_f = v_t + u_t t_p \tag{9}$$

$$v_f^2 = v_t^2 + 2u_t d_t. \tag{10}$$

The minimum time, t_p, to traverse the distance d_t is thus the smallest positive solution:

$$t_p = min \frac{-v_t \pm \sqrt{v_t^2 + 2u_t d_t}}{u_t} \tag{11}$$

Note that there are two such solutions, one for passing on the right and one on the left. Obviously, the smallest of the two is selected.

Selecting the time horizon as the smallest of the two times

$$t_h = min\{t_s, t_p\} \tag{12}$$

ensures that when the robot's velocity touches the boundary of the velocity obstacle, there remains sufficient time to avoid the obstacle either by stopping *or* by passing. Penetrating the velocity obstacle would leave no time for a safe avoidance maneuver, which implies that the boundary of the velocity obstacle, generated for $t \leq t_h$, represents states on the boundary of the *ICS*. The time horizon is computed individually for each obstacle, using the relative velocity between the robot and obstacle.

We can summarize these results in the following statements:

Statement 1: Assume a robot A that is modeled by a point mass, and is moving at some velocity v_a in the vicinity of obstacle B. B is moving at a constant known velocity, and is represented by its velocity obstacle $NLVO_{t_0}^{t_h}$ for $t \in (t_0, t_h)$, where t_h is the minimum of the stopping and passing times. If $v_a \notin NLVO_{t_0}^{t_h}$, then 1) A will not collide with B within the time $t \in (t_0, t_h)$; 2) A has sufficient time and control effort to avoid colliding with B either by stopping or by passing.

Statement 2: For the scenario in Statement 1, if $v_a \in NLVO_{t_0}^{t_h}$, then A cannot escape collision with B.

Statement 3: The boundary of $NLVO_{t_0}^{t_h}$ represents states that overlap with the boundary of the *ICS*. This follows from Statements 1 and 2 since every point inside $NLVO_{t_0}^{t_h}$ is also in *ICS*, and every point outside of $NLVO_{t_0}^{t_h}$ is not in *ICS*.

Figure 5 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 stopping and passing times along this path are shown in Figure 6, of which the smallest is used in determining the safe maneuver at each time step. As the stopping time depends only on the current

longitudinal velocity, it grows from zero, then decreases as the robot moves tangentially to the obstacle. The stopping time is zero at the tangency point since the longitudinal velocity that is pointing toward the obstacle is zero at this point. The stopping time is set to zero when the longitudinal velocity is negative, as is the case when the robot moves away from the obstacle.

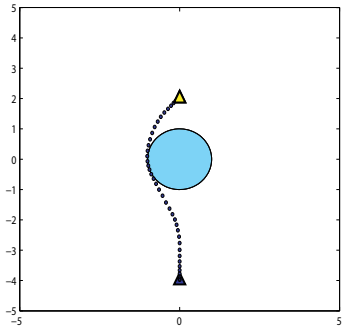

Fig. 5. Dynamic avoidance of a static obstacle

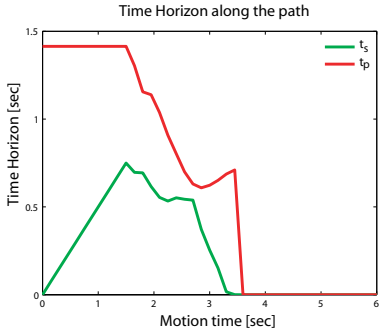

Fig. 6. The stopping and passing time horizons along the path

The passing time stays constant while the robot moves toward the center of the obstacle (the passing distance stays constant and the lateral velocity is zero all this time). The passing time decreases as the lateral velocity v_t increases. In this case, the passing time is higher than the stopping time along the entire path because of the relatively large size of the obstacle. The time horizon along the entire path is therefore the stopping time. Using the minimum time horizon between the stopping and passing ensures that at no time the robot enters inevitable collision states.

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 [14]. 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. Consequently, the success of the on-line planner should be measured by its ability to achieve a hierarchical set of goals, with survival being the first, reaching the goal being second, and other objectives being of lower priority. 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 proper time horizon to guide the tree search.

A. System Dynamics

For simplicity, the robot is assumed a planar point mass. 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.

Given the robot's dynamics, 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 [13]. This set contains the avoidance maneuvers that are dynamically feasible from a given state. The attainable *Cartesian* velocities, $ACV(t+\Delta t)$ are integrated from the current state $(x,v) = (x,y,\dot{x},\dot{y})$ by applying all admissible controls $u = (u_1,u_2) \in U$:

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

The geometric shape of $ACV(t+\Delta t)$ depends on the specific system dynamics. For a point mass model, with constant control constraints, it is a rectangle, similar in shape to the set of admissible controls U, as shown in Figure 7.

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

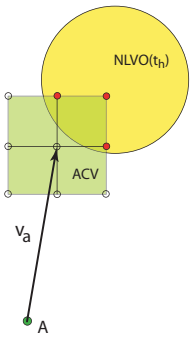

Fig. 7. Attainable Cartesian Velocities

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, as discussed next. 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.

C. Cost Function

Our search is guided by a minimum time-to-go cost function to produce near-time optimal trajectories to the goal. The time horizon selected in Section III guarantees safety by ensuring that the robot stays out of the *ICS*. Combining minimum time with the safe time horizon produces high speed, yet safe trajectories.

The cost function for each node is the minimum time to go to the goal from that node. It is determined by first computing the minimum time to the goal $w(x, \dot{x}, x_f, \dot{x}_f)$ from the current state $(x, \dot{x})$ to the goal $(x_f, \dot{x}_f)$ for each axis [19], [16]:

$$\begin{aligned} w(x,\dot{x},x_f,\dot{x}_f) \quad &= \\ -\dot{x}-\dot{x}_f \quad &+ \quad 2\sqrt{-x+x_f+\frac{\dot{x}^2}{2}+\frac{\dot{x}_f^2}{2}}, if x \in R \\ \dot{x}+\dot{x}_f \quad &+ \quad 2\sqrt{x-x_f+\frac{\dot{x}^2}{2}+\frac{\dot{x}_f^2}{2}} otherwise \end{aligned} \quad (16)$$

where R is the region below and above the switching curve in the state space:

$$\begin{aligned} R(x,\dot{x}) = \quad & \{\dot{x}^2 - 2(x - x_f + \frac{\dot{x}_f^2}{2} > 0, \\ & \dot{x}^2 + 2(x - x_f - \frac{\dot{x}_f^2}{2} < 0\} \end{aligned} \quad (17)$$

Considering both axes, the minimum time to the goal used in the cost function is the largest of the times computed for both axes [19]. This cost function produces time-optimal trajectories with no obstacles, and near-time optimal trajectories with obstacles.

V. Examples

The on-line planner was implemented and tested for obstacle-free, and crowded static and dynamic environments. In the first example, shown in Figure 8, the robot, represented by a point, starts near point $(0.25, -1)$ at zero speed, attempting to reach the goal at point $(0.25, 2)$ (marked by a red triangle) at zero speed, while avoiding two obstacles, one static and one moving (to the right). The trajectory is shown in six snapshots, starting from the top left, and ending at the bottom right of Figure 8. In each snapshot, the two obstacles are shown in blue, together with their temporal velocity obstacles $NLVO(t_h)$, shown each at the respective time horizon. Also shown is the robot trajectory up to that point from the start, with the velocity marked at the current point. Note that as the time horizon decreases, the velocity obstacle increases. At first, the robot turns left to avoid penetrating the velocity obstacle. This turn to the left occurs before the robot reaches the obstacle. After passing the static obstacle on the left, it turns right, to avoid the moving obstacle. At some point, the robot grazes the obstacle on the right, after which it is enclosed by the second velocity obstacle. This does not indicate a collision since its velocity points outside of the velocity obstacle. At that point, the relative velocity of the robot relative to the obstacle is tangent to the obstacle, as it should for the two to be sliding relative to each other. After avoiding the moving obstacle, the robot turns to the left to reach the goal. The second example,

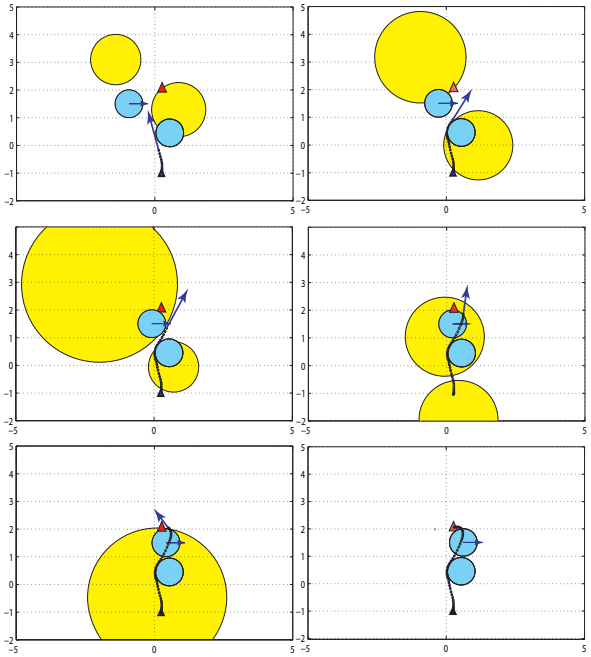

Fig. 8. Avoiding a static and a moving obstacle. Obstacles are shown in blue, and their respective velocity obstacles shown in yellow. The velocity vector is guided not to penetrate the velocity obstacles.

in Figure 9, shows the robot avoiding 70 static obstacles, starting from the bottom left and moving to three different goals. For each goal, two trajectories are shown: the local trajectory in blue dots, and the global trajectory in red. The dots are spaced at constant time intervals, thus indicating the changing speeds as the robot accelerates and slows down through narrow passages toward the goal. The local and global trajectories are quite similar for all three cases. The travel time along the local trajectories were about 10% longer than the time along global solutions. The third example, in

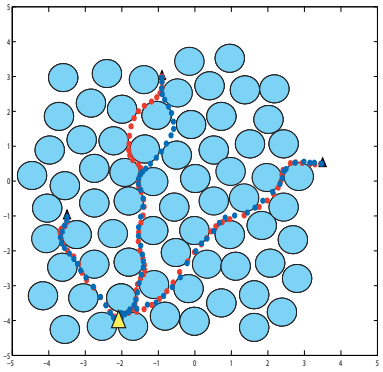

Fig. 9. Local (blue) and global (red) trajectories avoiding 70 static obstacles

Figure 10, 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. A video clip of the full run is available in www.ariel.ac.il/me/pf/shiller/oren.

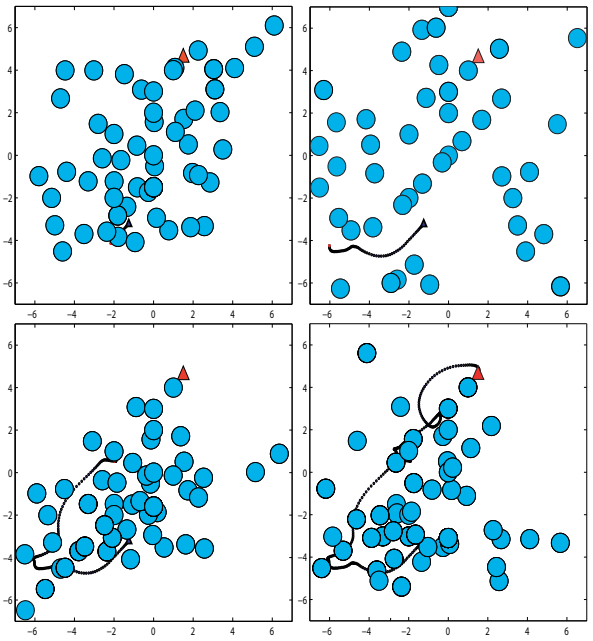

Fig. 10. Avoiding 70 moving obstacles

VI. Conclusions

An on-line planner for dynamic environments was presented. It ensures safety by using velocity obstacles that are truncated at a carefully selected time horizon. This time horizon is selected for each obstacle, static or moving, as the smallest of the minimum stopping and minimum passing times 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 *Proc. of the IEEE International Conference on Robotics and Automation*, 2000.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[16] S. Sundar. *Near-Time Optimal Feedback control of Robotic Manipulators*. PhD thesis, UCLA, 1995.

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

[18] M.S. Wikman, T.S. Branicky and W.S. Newman. Reflexive collision avoidance: a generalized approach. In *Proc. of the IEEE International Conference on Robotics and Automation*, 1993.

[19] Sundar S. Shiller Z. Time optimal obstacle avoidance. In *IEEE International Conference on Robotics and Automation*, pages 3075–3080, 1995.