**Seventh International Conference on Control, Automation, Robotics And Vision (ICARCV'02), Dec 2002, Singapore**

# Using Non-Linear Velocity Obstacles to Plan Motions in a Dynamic Environment

Frederic Large[1], Sepanta Sekhavat[1], Zvi Shiller[2] and Christian Laugier[1]

[1] *INRIA Rhône-Alpes, 38334 Saint Ismier cedex, France, (firstname.lastname)@inrialpes.fr*
[2] *Faculty of Engineering, College of Judea and Samaria, Ariel, Israel 44837.*

**Abstract**

*This paper focuses on real-time motion planning in a dynamic environment. Most of the global existing approaches cannot satisfy real-time due to heavy computation, while local methods don't guarantee reaching the goal.*

*In this paper we present a novel global approach based on the* Non-Linear Velocity Obstacle Concept. *We use the rich information on the velocities admissible for the robot to build a complete autonomous navigation module, composed of a local obstacle-avoidance system coupled with an incremental global motion planner. Real-time computation issues are discussed. Results obtained in simulation for dynamic environments are presented.*

## 1 Introduction

This paper focuses on real-time global motion planning in a dynamic environment. The aim is to find a trajectory to a goal among moving obstacles. Three constrains must be usually verified: 1) the robot must be kept safe 2) the trajectory must be feasible (i.e. must be compatible with the kinematics and dynamics of the robot), and 3) global considerations must be taken into account to guarantee that the goal will be reached (no local minima). Recent approaches based on reactive avoidance methods already address these conditions. We propose to add another one: We want the robot to anticipate its future environment, in order to generate more stable trajectories, or avoid obstacles faster than him. In this context, existing approaches that satisfy this critera are mostly either too slow for real-time, or too simplified, considering only subsets of the solutions or simple problems. The Non-Linear Velocity Obstacle Concept (NLVO) instead is particularly well suited for this. The method presented in [1] maps the positions of the obstacles and their known or estimated trajectories directly into the space of the robot velocities. Hence kinematic and dynamic constrains are also considered. We propose to use this information provided to build a navigation module, consisting of a local avoidance system, and an online global planner that can handle dynamic changes in real-time. This paper is organized as follows: we first look at related works, and then review our concept of NLVO and its extension like risk notion and Long-NLVO. Several implementations are proposed to achieve real-time realisations. Then, we present our navigation method and the results obtained in simulation.

## 2 Related works

Motion planning in a dynamic environment is usually done 1) globally with simplifications on the problem, or 2) more accurately but with local planning using a reactive obstacle method.

### 2.1 Global Planning

The first approach uses classical planning methods for static environments [7], [8] extended with a time dimension, [10], [11], [12], [13]. Such methods take into account the dynamics of the environment, but due to the complexity of the computation involved [9], they are not suitable for online planning and are essentially interesting either from the theoretical point of view, or to compute an initial reference trajectory off-line. However, few real time implementations exist. In this case, only relatively basic environments are considered or the problem is simplified by considering only a subset of the feasible trajectories [16], [17] or by using random trees [18] for faster computation. The Elastic Band method [19], [20] does not follow this scheme. A global free path to the goal is first computed. Bubbles are placed along this path to map the free space. Attractive internal forces are used between each pair of bubbles to minimise the length of the resulting band, while repulsive external forces from obstacles tend to maintain its safety. The combination of these forces lead to a dynamically updated free trajectory. This method is the most promising since it has been validated even for high degree workspaces. Recent extensions take

kinematics and mobile obstacles [14], [15] into account, although there is no anticipation on future obstacle trajectories.

### 2.2 Local Planning

The second approach consists of first planning a collision free path and to use a local reactive avoidance method to follow it. When there are obstacles, the solution is either to adapt the velocity of the robot [21], or to compute a local avoidance trajectory. These methods rely heavily on the initial path and when it has to be replanned, computation time is be critical. There have been methods to improve online path planning using probabilistic roadmaps such as the Ariadne's clew Algorithm [22], which expands incrementally the searched trajectory space. However, last works [23] [24], [25] still build the roadmap offline.

Using reactive methods and heuristic algorithms appear to be a better choice: A reactive obstacle avoidance method [27] [28] [30] [2] [3] is first used to compute free velocities from perception data. These methods alone are sufficient to keep the system safe, nevertheless, they are very sensitive to local minima. Hence, almost all recent works propose to add look-ahead capabilities to reduce this drawback [26] [4]. The common approach is to incrementally build an occupancy grid, which assumes static obstacles and is updated locally from sensor data.

These approaches use their reactivity to deal with dynamic environments but they usually do not make use of the obstacles trajectories to anticipate free space. As a result, the selected trajectories may change very fast in an evolving environment. Another essential problem is the risk of collision with obstacles that are faster than the robot.

The concept of Velocity Obstcale (VO) proposed in [5] takes into account the linear velocities of obstacles. The idea is to represent the obstacles directly in the velocity space of the robot, in order to compute geometrically the set of velocities leading to collisions on a time horizon, and conversely. The main drawback lies in the assumption that robot and obstacles have straight constant trajectories and infinite accelerations. However, it is not possible to satisfy these assumptions in real applications. All these limitations were compensated in [6] by an analytical optimization module. In [1], we reconsidered the approach of VO and extended it to arbitary obstacle trajectories. This concept also provids information on time to collision and how far obstacles would pass to the robot. Notions of uncertainty and basic dynamics are also considered.

## 3 Review of Non-Linear Velocity Obstacle

**General principle.** The concept of the Non-Linear Velocity Obstacle is derivated from the concept of *Velocity Obstacle* introduced in [5]. A short introduction is provided here and readers may refer to [1] for more details on NLVO.

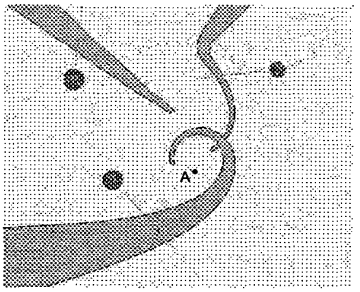



***Figure 1:** Non-linear Velocity Obstacles*

First we consider a time horizon $th$. The robot and the obstacle are approximated by a set of circles (We make the assumption here that any shape can be approximated by circles). The obstacle is grown by the radius of the robot. Hence, the problem is simplified to a point robot $A$ avoiding an obstacle $B$. We assume that the obstacle trajectories are known or can be estimated. We note *NLVO(t)* as all the velocities which would induce a collision with B at time $t$. The NLVO represents the union of all *NLVO(t)*, i.e. the velocities which would induce a collision with B before $th$ (with $t_0 < t < th$).

$$NVO = \cup NLVO(t) \quad (1)$$

$$= \cup_t((v_A(t_0)|A + v_A(t_0).t) \in B(t)) \quad (2)$$

where $v_A(t_0)$ is a linear robot velocity originating at $A$ at $t_0$ and $B(t)$ is the set of points occupied at time $t$ by $B$.

If $c(t)$ is the position of the center of B along its trajectory and $\beta$ describes the shape of B, this gives the general expression:

$$NLVO(t) = \frac{c(t) + \beta}{t - t_0}, t \subset ]t_0, th[ \quad (3)$$

**Computing the boundaries.** The general expression of the NLVO is parametrised by time. This places a big dependance on the discretisation of time (Fig 2a) in the implementation.

One possible approximation is to compute the convex hull of the NLVO(t) (Cf. Fig 2b), however there is much calculation involved.

For convenience, an analytical expression of the boundaries of the NLVO discretised by time would be more appropriate. Let us call L(t) and R(t) respectively the left and right boundaries. The algorithm to geometrically compute L(t) and R(t) has been shown in [1] (Fig. 2c). The resolution of this expression may be complicated and a simpler approximation illustrated by the figure 2d) is prefered. This solution is generally sufficient. One exception is when obstacles are too close to the robot, then the approach is no more conservative. In this case, an exact checking is performed.

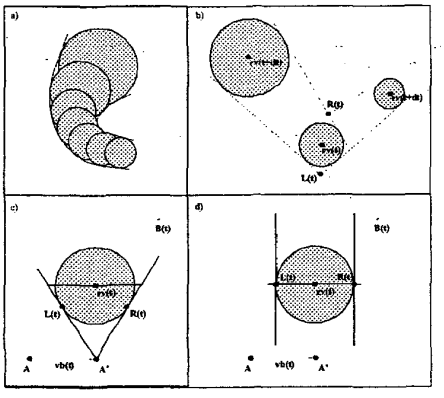


*Figure 2: Construction of the approximate NLVO*

Now considering a circular obstacle $B$ following a trajectory $c(t)$ in a frame centered at A. The expression of the center line of NLVO is noted $c_v(t) = c(t)/t$. $v_b(t)$ is the linear velocity of the obstacle B at time t. Using these notations, the expression of the left and right boundaries are discretised by time respectively, $vo_l(t)$ and $vo_r(t)$, becomes :

$$vo_r(t) = c_v(t) + i\frac{r}{t}(c_v(t) - v_b(t)) \qquad (4)$$

$$vo_l(t) = c_v(t) - i\frac{r}{t}(c_v(t) - v_b(t)) \qquad (5)$$

In practice, the time is kept as a third dimension added to the boundaries coordinates, leading to a 3D map. It is used to inform on the time to collision and then estimate a risk associated with a velocity (Cf. section 3).

**Long Obstacles.** Many environments have obstacles with long forms like walls, and we proposed in [31] an optimization method to compute the VO corresponding to such obstacles is used.

We consider long rectangular obstacles having a length much greater than the width. A simple way to map such shapes with disks is to make overlapping $n = length/width$ circles of radius $r = \frac{width}{\sqrt{2}}$ along the longer dimension of the rectangle. This gives rise to many useless virtual obstacles, hence many useless NLVO. We propose instead to keep only the two extreme virtual circular obstacles and to link them by two tangents defining a band (Cf. fig. 3). This construction allows much faster computation of the corresponding VO by taking extreme left and extreme right boundaries only. It is particulary useful to represent indoor environments like corridors and treat them as any obstacle in a more efficient way.

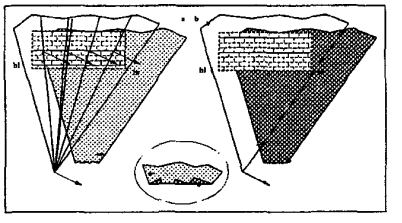


*Figure 3: Using n circles (a) or 2 linked circles (b)*

**Notion of Risk.** We propose to provide a way of distinguishing a collision free velocity from another by quantifying a risk associated to each velocity. Two observations motivated a definition of this risk function : Velocities that induce immediate collisions are considered more dangerous than those that give collisions later. Also, velocities which induce trajectories passing closer to obstacles are more dangerous. Computing such a function is not satisfactory from an implementation point of view. Instead, we propose to use an approximation.

The first method is a grid based approach, while the second uses triangulation techniques. Grid based techniques requires the velocity space to be decomposed into rectangular cells. The grid has one axis as the x component of the velocity, $v_x$, and the other axis is the y component, $v_y$. The content of a cell($v_x$,$v_y$) is a normalized value reflecting the risk of the velocity v($v_x$,$v_y$). This risk is defined by $v_y$.

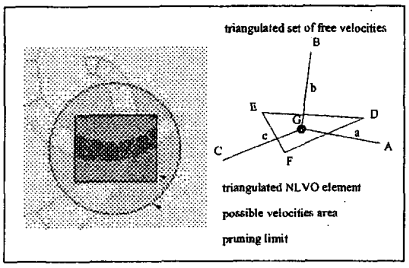
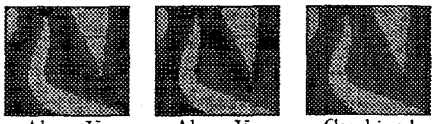



*Figure 4: Risk approximation using triangulation*



Along Vy Along Vx Combined

**Figure 4b**: Gradient Functions

Figure 4b shows sweeping cuts made across the grid in along $v_x$ and $v_y$ directions, and their combined grid. The darker colours mean that the velocities are safer. The main inconvenience of this method is its sensivity to the discretisation used. A coarse discretisation will result in poor separation of free velocities, this means that free velocities may be hidden by dangerous ones if both are mixed together in the same cell. A fine discretisation will give better results, however it causes heavy computation again. During our experiments, we used an average grid size less than 32X32 cells.

Hence, a second method which uses Delaunay triangulation is preferred. The intersection between the boundaries of the NLVO and the velocities possible for the robot can be stored as a set of polygons. Instead of using rectangular cells, triangulation is done on this mesh to group the velocities which have about the same risk value. Hence, each triangle contains a set of velocities which have a similar risk factor (Each triangle can be refined if needed). They can be easily sorted to the group of safe and dangerous velocities. Each of the 3 vertices of this triangle is weighted according to its risk factor, and we find the point lying in this triangle similar to the barycenter. However, the position of this point takes into account not only the distance but the weights of each vertex. To find this point, refering to figure 4, we find D such as $\frac{risk(A)}{risk(B)} = \frac{DA}{DB}$. We proceed the same for E and F. G is the barycenter of the triangle DEF. $risk(G)$ is then defined as $\frac{d_{max}}{3}(\frac{risk(A)}{a} + \frac{risk(B)}{b} + \frac{risk(C)}{c})$ where $d_{max}$ is an upper bound for any a, b, and c. The main advantage of this method is that no additional discretisation is needed. After pruning based on squared distances, only few triangles need to be computed, allowing real-time computation of "valleys".

None of the previous approximations allow the computation of a real risk, nevertheless, they are sufficient to sort the free velocities in a coherent way.

## 4 Local goal-oriented obstacle avoidance

Our approach is based on a cost function associated with each velocity, according to the current knowledge of the environment. Choosing the best velocity corresponds to finding the velocity with the highest cost. The novelty of this method consists of using the rich information provided by the NLVO such as the time to collision combined with the previously described notion of risk. The chosen local trajectory takes into account the current environment, its evolution (obstacle trajectory) and most importantly, a notion of risk as a safeguard against errors during the exection of the manoeuvres or wrong estimation of obstacle position and trajectories. Our cost function $C(v)$ was first expressed as a weighted sum of 3 subfunctions returning values between 0 and 1. The first one noted $V(v)$ favours higher velocities, the second $H(v)$ favours heading to the goal, and the last $F(v)$ favours safer velocities (far from obstacles).

$$V(v) = (v/v_{max}) \tag{6}$$

where $v_{max}$ is the maximum velocity

$$H(v) = 1 - \left|\frac{\theta_v - \theta_g}{\pi}\right| \tag{7}$$

where $\theta_v$ is the velocity orientation and $\theta_g$ the orientation to the goal

$$F(v) = \begin{cases} 1 - \frac{R(v)}{2} if no\ collision \\ \frac{t_c(v)}{2t_{max}} otherwise \end{cases} \tag{8}$$

where $t_c(v)$ is the time to collision, $R(v)$ the risk function, and $t_{max}$ the maximum time horizon

$$C(v) = \alpha.V(v) + \beta.H(v) + \gamma.F(v) \tag{9}$$

where $\alpha$, $\beta$ and $\gamma$ are tuning weights to retrieve the expected behaviour.

The cost function can deal locally with obstacles, nevertheless, its convergence to the goal is difficult to obtain depending on $\alpha$, $\beta$ and $\gamma$. In an improved version, the cost function $C(v)$ is instead defined as

the euclidean distance between V and an optimal velocity $V(opt)$. $V(opt)$ is the next velocity the robot would apply to follow an optimal trajectory to the goal if there were no obstacle. The essential kinematics are considered for the robot to maintain simplicity in this method. The safe velocities are first computed using NLVO and the function of risk. The cost function selects the the most appropriate among these.

This method has been implemented in simulation. We have achieved good results even with 200 000 randomly placed obstacles (100 000 circular and 100 000 long) with continuous trajectories, a refresh rate of 50Hz, a time division of 10ms and a time horizon of 10 seconds on a classical PII-433MHz (with the GUI turned off).

## 5 Towards global motion planning

The method described earlier is a local approach. When the optimal trajectory to the goal is not collision-free, we cannot guarantee that the robot will reach the goal later. This problem can be solved by using graphs. A heuristic based on the previous cost function determines how the tree expands, the value of each leaf in this tree corresponds to the lower limit of the time necessary to reach the goal following the trajectory passing by this node. The estimation of the time is obtained by summing the time necessary to reach the node and the time necessary to reach the goal from this node if there were no obstacles. This heuristic is compatible with the popular A* algorithm [29]. The leaves with the lower weights are explored first, leading to optimal solutions in the graph. Since NLVO is able to anticipate the future situations, the number of branches in the graph is reduced. Only the safe trajectories are kept with the help of the risk function. The predictive property of the method also ensures that the tree will be valid on several controller iterations. This is the key to incrementally build the global trajectory.

The challenge consists of replanning a new global trajectory while the robot is moving and keeping the system safe at the same time. Our current approach consists on running two modules in parallel to perform the task. The first one is the local avoidance module described in section 4. Its purpose is to keep the system safe at any time by checking the validity of the current trajectory. The second module is based on the incremental building of the tree. At each iteration, if the currently followed trajectory is valid and time permits, we expand the search tree until the goal is reached.

The deeper we progress into the graph, the more we choose risky branches, but it is also more optimized.

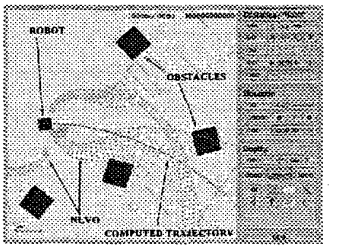



***Figure 5:*** *Snapshot of our Simulator*

We choose this heuristic because we want the *base* of the graph, i.e. the next closest velocities, to remain valid as long as possible. Velocities of the future may need to be recomputed as it is likely that they will be invalidated before they are actually applied. This choice makes the system converge faster to a solution. Ongoing experiments (Cf. figure 5) are aimed to characterising when the system should follow safe or optimal solutions.

## 6 Conclusion

The real-time computation of a collision-free trajectory to a goal in a dynamic environment requires an efficient, i.e. fast, computation process. Global planners based on robot and obstacles positions need too much time to integrate all this information when the number of obstacles increases but give good trajectories to the goal. Methods based on reactive approaches with look-ahead capabilities are very fast instead, but do not anticipate the evolution of the environment. In this paper, we proposed a solution based on the *Non-Linear Velocity Obstacle* concept. We used the rich information provided by this method such as a notion of risk for each velocity. Implementation for real-time were discussed. Based on the NLVO, we proposed a novel local goal-oriented obstacle avoidance method which takes into account the known or estimated obstacles trajectories on a given time horizon leading to more stable trajectories. Learning from existing planification methods, we then proposed to combine our results with graph-expansion techniques, leading to an incremental global motion planner in a dynamic environment.

## Acknowledgments

This work was supported by the LaRA (La Route Automatisée) and the IMARA (Informatique, Mathématique et Automatique pour la Route Automatisée) programs of INRIA on intelligent road transportation). The first author would like to thank, Priscilla Pek who has spent much effort to correct this paper.

## References


[1] Shiller, Large and Sekhavat *Motion Planning in Dynamic Environments: Obstacles Moving Along Arbitrary Trajectories.*In Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 3716-3721, Seoul (KR) (May 2001).

[2] J.Borenstein, Y.Koren. The Vector Field Histogram (VHF) Fast Mobile Robots. IEEE International Transactions on Robotics and Automation. Vol7. 1991.

[3] I.Ulrich, J.Borenstein. VHF : Reliable Obstacle Avoidance for Fast Mobile Robots. IEEE International Conference on Robotics and Automation. pp1572. Leuven, Belgium, 1998.

[4] I.Ulrich, J.Borenstein. VHF* : Local Obstacle Avoidance with Look-Ahead Verification. IEEE International Conference on Robotics and Automation. pp2505, San Francisco, USA. 2000.

[5] Fiorini, Shiller *Motion Planning in Dynamic Environments Using the Relative Velocity Paradigm.* In IEEE International Conference on Robotics and Automation, Atlanta GA, May 1993.

[6] Fiorini, Shiller *Time Optimal trajectory planning in dynamic environments.* In Proc. of the IEEE Int. Conf. on Robotics and Automation, volume 2, pages 1553-1558, Minneapolis, Minnesota, April 1996.

[7] Latombe *Robot Motion Planning.* Standford University, 1990.

[8] Hwang, Ahuja *Gross Motion Planning A Survey Advanced Robotics Redundancy and Optimisation, ACM Computing Surveys, Addision Wesley Publishing Company volume 24, No. 3, September 1992.*

[9] Canny *On the complexity of kinodynamic planning.* In Proc. of the IEEE Symp. on the Foundations of Computer Sciences, pages 306-316, White Plains, NY (USA), November 1988.

[10] Shiller, Dubowsky *Robot path planning with obstacles, actuator, gripper and payload constraints.* In Int. Journal of Robotics Research, 8(6):3-18, December 1989.

[11] Sahar, Hollerbach *Planning of mininum time trajectories for robot arms.* In Proc. of the IEEE Int. Conf. on Robotics and Automation, pages 751-758, St Louis, MI (USA), March 1985.

[12] Donald, Xavier *Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open-chain manipulators.* In Proc. of the ACM Symp. on Computational Geometry, pages 290-300, Berkeley, CA(USA), 1990.

[13] Fraichard *Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach.* In Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, volume 2, pages 1394-1400, Yokohama (JP), July 1993.

[14] Brock. Generating Robot Motion: The Integration of Planning and Execution. November, 1999.

[15] Brock, Khatib. real-Time Replanning in High Dimensional Configuration Spaces Using Sets of Homotopic Paths. pp2328. San Francisco. USA, 1993.

[16] Zhu, Latombe *New Heuristic Algorithms for Efficient Hierarchical Path Planning.* In Artificial Intelligence, North-Holland, Vol. 27, No. 1, pages 9-20, February 1991

[17] Fraichard *Trajectory Planning in A Dynamic Workspace : a 'State-Time Space' Approach.* In Advanced Robotics, 13(1):75-94, 1999.

[18] LaValle *Rapidly-Exploring Random Trees: A New Tool for Path Planning.* Technical Report No. 98-11, Dept. of Computer Science, Iowa State University, Oct. 1998.

[19] Quinlan, Khatib. Elastic Bands: Connecting Path Planning and Control. Vol 2, pp 802-807. Atlanta, USA, 1993.

[20] Quinlan. Real-Time Modification of Collision Free Paths. December 1994.

[21] Kant, Zucker *Toward efficeient trajectory planning: the path-velocity decomposition.* In Intl. Journal of Robotics Research, 5(3):72-89, Fall 1986

[22] Ahuactzin *Le Fil d'Ariane : Une Méthode de Planification Génerale. Application à la Planification Automatique de Trajectoires.* Ph.D Thesis, Thèse de doctorat. Inst. Nat. Polytechnique de Grenoble. Grenoble(FR),September 1994.

[23] Bessiere, Ahuatzin *The 'Adriane's Clew Algoritm': Global Planning with local methods.* In IEEE/RSJ Conf. on Intelligent Robots and Systems, 1993.

[24] Cherif, Vidal *Planning handling operations in changing industrial plants.* In IEEE Int. Conf. on Robotics and Automation, 1998.

[25] Ahuactzin, Gupta *The kinematic roadmap: A novel motion planning based approach for inverse kinematics of redundant manipulators.* In IEEE Transactions on Robotics and Automation, 15(5), 1999.

[26] Brock, Khatib *High Speed Navigation using the global dynamic window approach.* In Proc. Int. Conf. on the Robotics and Automation, 1999.

[27] Fox, Burgard and Thurn *A hybrid collision avoidance method for mobile robots.* In Proc. of the IEEE Intl. Conf. on Robotics and Automation, volume 2, pages 1238-43, 1998.

[28] Simmons *The curvature-Velocity Method for Local Obstacle Avoidance.* In Proc. of the IEEE Intl. Conf. on Robotics and Automation, Minneapolis, Minnesota, April 1996.

[29] Nilsson *Principles of Artificial Intelligence*, Springer, Berlin, 1982

[30] Minguez, Montano *Nearness Diagram Navigation. A New Real Time Collision Avoidance Approach*, IEEE/RSJ Int. Conf. on Intelligents Robots and Systems, Takamatsu, Japan 2000.

[31] Large, Sekhavat, Shiller and Laugier *Towards Real-Time Global Motion Planning in a Dynamic Environment Using the NLVO Concept.*In Proc. Int. Conf. on Intelligents Robots and Systems, Lausanne, 2002.