Motion planning in dynamic environments using the velocity space

Eduardo Owen †‡
‡*Escuela de Ingeniería Eléctrica y Electrónica*
Universidad del Valle
Calle 13 No. 100-00, Cali, Col
efowen@unizar.es

Luis Montano †
†*Instituto de Investigación en Ingeniería de Aragón*
Dep. Informática e Ingeniería de Sistemas, Universidad de Zaragoza
María de Luna 3, E-50018 Zaragoza, Spain
montano@unizar.es

***Abstract*— This paper addresses a method for robot motion planning in dynamic environments, avoiding the moving and static obstacles while the robot drives towards the goal. The method maps the dynamic environment into a velocity space, using the concept of *estimated arriving time* to compute the times to potential collision and potential escape. The problem of finding the best motion command is directly treated in the velocity space, providing the trajectory which satisfies an optimization criterium (tipically the minimum time or the shortest path). In this work the method is applied to robots which are subject to both kinematic constraints (i.e. involving the configuration parameters of the robot and their derivatives), and dynamic constraints, (i.e. the constraints imposed by the dynamics of the robot and the limits of its actuators). Some experimental results are discussed.**

***Index Terms*— Mobile robot navigation, dynamic environments, velocity space, motion planning**

I. Introduction

The collision avoidance problem has been extensively treated in the robotics literature. Roughly, two types of approaches which can be classified as global and local (reactive) have been addressed. The global methods [11] plan optimal paths towards the goals but they do not deal well with unexpected changes in the environment. In [16] a complete path planner for changing environments is presented. Unlike the preceding method, the local avoiding methods such as [17], [4], [12], [15] cope with these unexpected changes in a reactive way, but they do not utilize the kinematic or dynamic information of the environment to compute the motion commands. Moreover most of these works do not cope with the problem of the robot velocity planning using that information.

When the available velocity information of the objects obtained from the sensors is utilized, the navigation system can compute trajectories which improve the motion performance regarding other classical obstacle avoidance methods. The motion planning in dynamic environment has been studied by several researchers and different strategies have been proposed to solve it. [13] deals with this problem by searching solutions in a visibility graph in the configuration-time space. [1] discusses the problem by discretizing the configuration-time space in slices of the configuration space at time intervals. [7] proposes a cell decomposition to represent the configuration-time space and then joins empty cells to connect the free space. [8] defines the concept of collision front as the geometric locus of collision points between two objects. Using this idea they create an accessibility graph and state a minimum-time problem. [9] resolves the problem of avoiding static objects, and then they plan the speed along that path. [6] introduce the concept of state-time space to formulate problems of path planning in dynamic settings. Under certain considerations it is possible to transform the problem of finding time optimal path and find the shortest way in a directed graph in this new space [5]. Finally, in [2] and [10], the concept of velocity obstacle is used to transform the dynamic problem in several static problems. However kinematic constraints are not taken into account.

Some probabilistic approaches for navigation in dynamic environments use the Markov Decision Processes (MDPs) to find optimal control commands that allow the robot to go to the goal avoiding obstacles. [14] proposes a policy search in a high dimensional control space, aiming to find plans that could lead to better motion planning. [3] introduces a methodology for obstacle avoidance by controlling the velocity, but discretizing to three different speeds.

In order to plan optimal or near-optimal trajectories (minimum time, shortest path or both) avoiding collisions in dynamic scenarios for real robots, it is necessary to take into account the robot constraints and the kinematic or dynamic information gathered by the sensors from the environment. This way we present in this paper a technique to locally plan robot motions on a model of the environment which reflects its dynamism. We transform the complete problem from the workspace or the configuration space to the velocity space in order to make decisions about the "best" commands directly in this space. The major contribution of this work is a method for modelling and mapping multiple static and moving obstacles into the velocity space. This velocity space model reflects simultaneously the robot kinematic and dynamic constraints and the environment dynamic, so decisions about the optimal command for a near horizon can be made. A heuristic method to compute velocity commands using that model is the other contribution of this paper.

The paper is organized as follows. In section II the approach is outlined. Section III presents the method to map the configuration space of a dynamic environment to the velocity space. The problem of computing trajectories using the velocity space built is presented in section IV. Simulation results are discussed in section V and in section VI some conclusions are presented.

II. The approach

The approach presented in this paper is based on the idea of mapping the motion of the robot and the static and moving objects of the environment from the workspace to the velocity space. The information mapped in this space is computed from two concepts: times to collision and times to escape from collision. The best robot velocity command is calculated in this space, in which the static obstacles and predictions about the moving ones are directly reflected. Several criteria to choose the optimal command can be applied, i.e. the minimum time trajectories, the shortest path, passing forward the moving objects or slowing down allowing the objects pass before. This approach differs from others in several characteristics:

- The commands are computed directly in the defined velocity space (VS), not in the workspace (WS), $W = R^2$, or in the configuration space, $CS = R^2 \times S^1$. This is an advantage in order to compute optimal trajectories, reasoning in terms of velocity against reasoning in terms of paths.
- The approach takes into account some constraints that restrict the motion capabilities of the robot (i.e. kinematic constraints and dynamic constraints), and they are directly reflected in the model.
- The mapping of environment on the velocity space is obtained from analytical expressions. So it is continuous and no grids are needed as in other approaches. Moreover, the eligible velocities extend to the whole velocity space, within the physical limits, and no discretization is made in this sense, as other approaches above mentioned.
- The velocity space implicitly includes time information about further potential collisions or escape possibilities, with static or moving objects. The model allows to plan motions within a visibility horizon further than the corresponding to one sampling period. So, a look-ahead to search the best command is feasible within this model.
- Optimization techniques can be applied on this space, either using global optimization or applying heuristics to find the optimal command in real time. This last approach is presented in this work.

The method is applied iteratively. Each sampling period the Configuration Space is computed and mapped to the Velocity Space, and the *best* velocity command is calculated by reasoning in VS. The next sampling period the procedure is resumed. Thus, the whole trajectory is computed as a sequence of short trajectories that converge to the goal. We assume in this paper that a global planner computes the subgoals to be reached by the robot to go towards the objective. A planner as the presented in [16] could be used, because it takes into account the changes in the environment, and fits well with the objective of navigation in changing scenarios. Also we consider that the objects locations are provided by a sensor (i.e. a laser range-finder) and the objects velocities are computed.

First we present the technique for mapping moving obstacles in the Configuration Space (CS) to the Velocity Space (VS). Secondly we explain how to obtain the velocity commands to reach the goal.

III. Mapping moving obstacles from the CS to VS

The mapping of the dynamic environment is based on computing the robot paths and velocities (trajectories) that would provoke further collisions with the objects. The collision time is also a relevant information implicit in the mapped space. We consider here the case of non holonomic robots. Besides, some constraints are imposed in this paper in order to present the method:

- The robot can move following *straight or circular paths*. This is a common constraint imposed to non holonomic robot motions. Assuming that the linear and angular velocities are constant during a sampling period and the process is resumed the next period, this is a reasonable constraint. In this way, we take into account the kinematic constraints.
- The objects move with a constant velocity following straight paths. This seems a strong constraint, but as the process is resumed every sampling period it can consider the object motion as a sequence of short straight paths. Anyway, the method presented can be easily extended to other kind of paths, thus there is not loss of generality. For the shake of clarity we develop the method using this constraint.
- The objects are represented as polygons, square or rectangular. This approach makes easier the computation and provides a security margin to calculate the time to collision. Moreover, the robot is considered as circular, reducing the complexity of computations in the Configuration Space.

A. Computing the times of collision with obstacles

To apply the method, we use a local reference attached to the robot. All the computations are made based on this local reference. Figure 1a represents the workspace (WS) with a

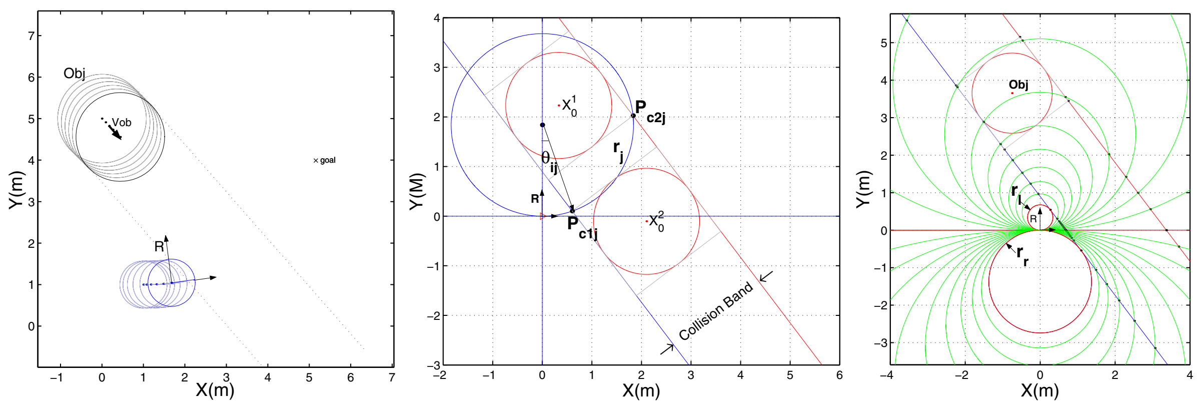


Fig. 1. (a) Workspace, (b) collision band, path r_j and collision points P_{c1j} and P_{c2j} in the Configuration Space (c) Extension to the whole range of paths potentially in collision

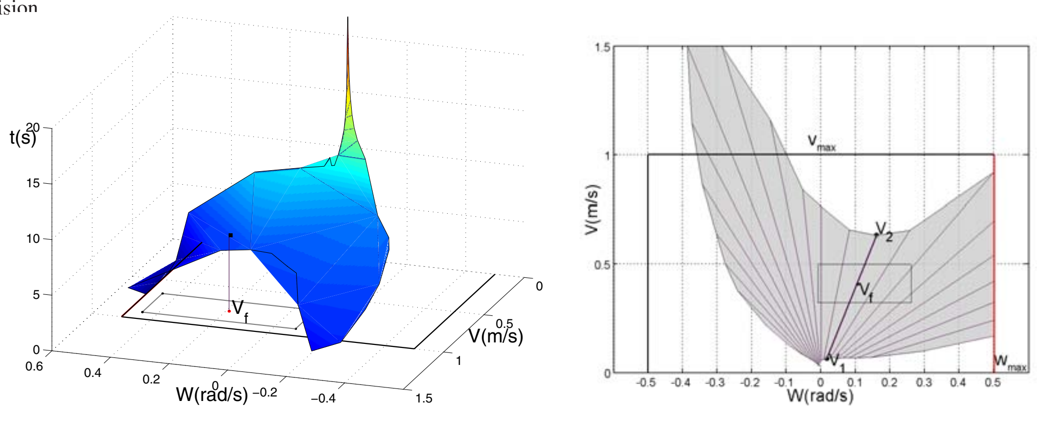


Fig. 2. (a)DOV in velocity-time space (b)projection of DOV, V_{DOV}, on the plane (v, w)

robot and a moving obstacle at constant velocity following a straight path. Figure 1b depicts the CS at instant k. It shows the Collision Band (zone swept by the object moving along a straight line) and an obstacle in two locations ($\mathbf{x_o}^1$ and $\mathbf{x_o}^2$) representing the locations in which the robot, following a trajectory r_j, arrives when the object has just passed at time t_{1j} (point P_{c1j}) or escapes from collision crossing just before the object arrives at time t_{2j} (point P_{c2j}). The computation of this pair of points and their associated times for the set of trajectories that can lead to collision is the basis of the proposed model.

From the known object location $\mathbf{x_o} = (x_o, y_o, \phi_o)$ in the robot local reference and its velocity $\mathbf{v_{ob}}$, the points of collision ($P_{c1j}(x_{1j}, y_{1j})$ and $P_{c2j}(x_{2j}, y_{2j})$) and the corresponding times t_{1j} and t_{2j} are calculated by solving the following equations for each path characterized by its curvature radius r_j and its center $(0, y_{cj})$ in the robot reference R. For each r_j, t_{ij} (i=1,2) is computed,

$$\begin{aligned} x_{ij} &= x_0 + v_0 \cos(\phi_0) t_{ij} \\ y_{ij} &= y_0 + v_0 \sin(\phi_0) t_{ij} \\ {r_j}^2 &= x_{ij}^2 + (y_{ij} - y_{cj})^2 \end{aligned}$$

obtaining two solutions for each collision point P_{cij},

$$t_{ij} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$\begin{aligned} A &= v_0^2 \\ B &= 2v_0(x_0 \cos(\phi) + y_0 \sin(\phi)) - 2v_0 \sin(\phi) y_{cj} \\ C &= x_0^2 + y_0^2 + y_{cj}^2 - 2y_0 y_{cj} - r_j^2. \end{aligned}$$

Two cases can appear. When the robot is out of the collision band we select t_i (in the sequel we will eliminate the subscript j for clarity) as the solution corresponding to the first intersection point (the lower value of t_i) for both P_{c1} and P_{c2}. From the times t_1 and t_2 the robot angular velocities are computed as $w_i = \theta_i / t_i$ being θ_i the robot angular displacement on r to reach P_{ci}. The linear velocities are computed as $v_i = r w_i$. Thus, we have computed velocities $\mathbf{v_1}$ and $\mathbf{v_2}$ and their corresponding times t_1 and t_2. When the robot is in the collision band, the strategy is to escape from the band before the object arrives, avoiding a collision. In this case only the escape point P_{c2} and velocity $\mathbf{v_2}$ are calculated, using the previous equations. A lower velocity would result in collision. Thus, $\mathbf{v_1} = \mathbf{0}$ and $t_1 = 0$ are chosen.

These calculations are extended to the whole space, considering a range of curvature radii between r_l and r_r (see Figure 1c). Figure 2a represents the velocity-time space obtained from the previous computations. The surface is built from the pairs (v_1, w_1, t_1) (v_2, w_2, t_2) belonging to the contour of the surface, obtained from every trajectory r_j. Note that the circular paths are transformed in this plane into straight lines ($r = v/w$). In this Figure, $\mathbf{v_1}$ and $\mathbf{v_2}$ are the extreme velocities computed for the path r_j. In the sequel we will name this surface Dynamic Object Velocity (DOV). The Figure 2b depicts the projection of the surface on the plane (v, w), V_{DOV}. Formally,

$$DOV = \{(v, w, t) | (v, w) \in [INT(V_{DOV}) \cup \delta V_{DOV}]\}$$

where INT denotes the set of velocities belonging to V_{DOV} zone and δ the velocities of the contour of V_{DOV}. The velocities under DOV or out of its projection are eligible, because they correspond to commands that do not lead to collision.

The contour of V_{DOV} represents the velocity limits $\mathbf{v_1}$ and $\mathbf{v_2}$ for every circular path computed as explained above. Notice that the velocities in V_{DOV} do not correspond to forbidden velocities (cannot be treated as "obstacles" in the sense of reactive navigation). In other words, $\mathbf{v_f}$ in Figures 2a,b could be chosen if the time t_f in which it is applied was lower than the time of collision on the surface DOV (that is, if it is under the surface). That is an important remark in order to compute the *best* trajectory in this space.

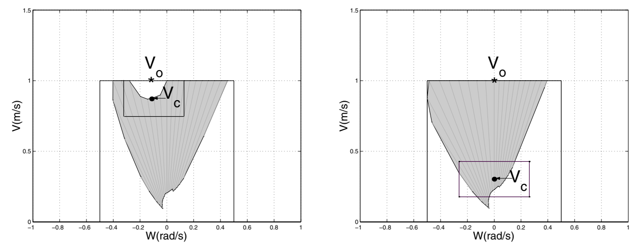
Static obstacles are computed in a similar way, but in this case all the velocities belonging to V_{DVO} for static obstacles are forbidden. It should be taken into account in the method used to compute the velocity command.

Extension to multiple objects is straightforward. The computations are repeated for each object. The union of all the zones of velocities DOV provides the Dynamic Objects Velocity Set (DOVS). The DVOS represents the velocities for which could have collision if they were maintained for some time, V_{DOVS} its projection and V_{free} the set of velocities without collision risk,

$$DOVS = \cup_{i=1}^{m}(DOV_i)$$
$$V_{free} = \{(v, w) | (v, w) \in [V_{adm} \ominus V_{DOVS}]\}$$

where V_{adm} is the set of admissible velocities limited by v_{max} and w_{max} and $\ominus$ the operation of set difference.

We will name Dynamic Velocity Space (DVS) the velocity-time space including the DOVS and V_{free} information. We can use different strategies to compute the *best* motion. Notice that the strategy can choose velocities in both zones, V_{free} and V_{DOV}. But obviously the computations have to be different in each case. In the IV section, we explain a strategy to compute the robot velocities.



(a) (b)

Fig. 3. (a) $\mathbf{v_o}$ is reachable only inside the window, (b) $\mathbf{v_o}$ is not reachable

B. Considering the Dynamic Constraints

In the previous subsection, it was assumed that any $\mathbf{v_1}$, $\mathbf{v_2}$ could be reachable in one step (sampling period T), which could be unrealistic. The maxima reachable velocities are only the ones inside the velocity window centered in the current velocity, $\mathbf{v_c}$, that is $(v_c \pm \Delta v_{max}, w_c \pm \Delta w_{max})$, which represent the dynamic constraints (acceleration constraint). Figure 3a shows that $\mathbf{v_o}$ only can be reached in one step from the velocities inside the window, due to the dynamic constraints. In Figure 3b it can be seen that $\mathbf{v_o}$ cannot be reached in one step, because it is outside the window. To deal with this problem, we compute the times needed to reach the limits $\mathbf{v_1}$ and $\mathbf{v_2}$ from $\mathbf{v_c}$ in DVS, t'_1 and t'_2 respectively. These times are used to recompute the velocity limits $(\mathbf{v'_1}, \mathbf{v'_2})$. The computation of t'_1 and t'_2 is a conservative way for considering the dynamic constraints in DVS. More formally the procedure can be expressed as follows:

1) compute how many sampling periods are needed to reach velocities $\mathbf{v_1}$ and $\mathbf{v_2}$ from $\mathbf{v_c}$. These are t'_1 and t'_2.
2) compute the new velocity limits $(\mathbf{v'_1}, \mathbf{v'_2})$ from t'_1 and t'_2, respectively, using the same equations that in previous subsection. This way the robot dynamic constraints are considered in order to have secure limits for the collision velocities.

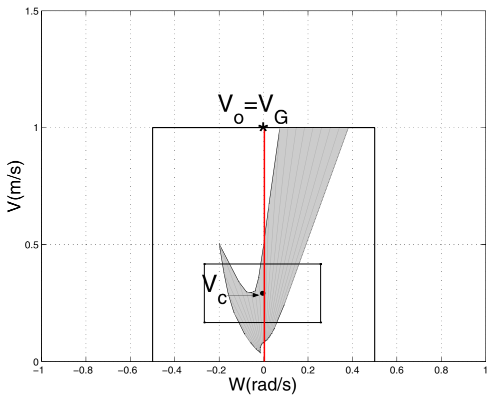
Figure 3b represents the DVS when dynamic constraints are taken into account. In this case $\mathbf{v_o}$ drives to collision, thus it cannot be chosen. This allows to augment the visibility horizon in DVS for commands eligible in more than one step.

IV. Computing trajectories using DVS

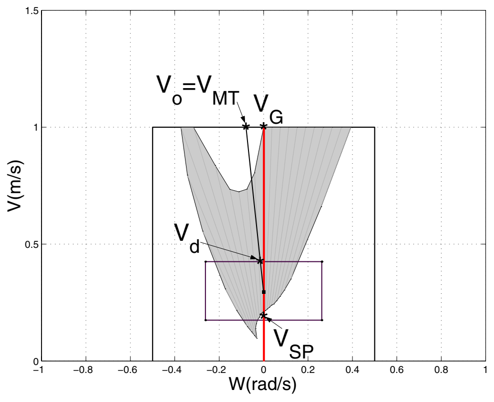
Different techniques to find the optimal commands can be applied using the model based on the DVS. We develop here a heuristic method that exploits the information included in DVS. The method is applied in three steps: first, the subgoal or the goal provided by the planner is mapped in DVS, $\mathbf{v_G}$. Second, a velocity command, $\mathbf{v_o}$, is planned in V_{free} as a function of $\mathbf{v_G}$, to drive the robot towards the goal. Third, the setpoint velocity in the sampling period is computed, regarding the dynamic constraints.

Step 1. Mapping the goal

To map the goal into DVS, we compute the angle between



(a)



(b)

Fig. 4. (a) $\mathbf{v_{MT}}$ leads to move around the obstacle at maximum linear velocity, (b) $\mathbf{v_{SP}}$ generates a rectilinear motion (the shortest path) at a lower velocity. $\mathbf{v_d}$ is the setpoint velocity

the current robot heading and the direction of the goal (ψ) in the local reference system. From this angle we obtain the angular velocity, $w = \psi/T$, to align the robot and the goal, being T the sampling period. Under the maximum linear velocity (v_{max}) and maximum turn velocity criteria, if $w \geq w_{max}$, then $w = w_{max}$. The pair (v_{max}, w) defines the velocity, $\mathbf{v_G}$. This policy tries to align the robot and the goal as fast as possible in absence of obstacles. If there are obstacles, $\mathbf{v_G}$ is only an initial guide to select the best command. Obviously only velocities compatible with the particular kinematic constraints of the vehicle (differential drive, car-like) can be selected.

Step 2. Planning the velocity command
We compute a velocity command $\mathbf{v_o}$, regarding $\mathbf{v_G}$ previously mapped. The policy will be to reach that velocity as soon as possible, so the best velocity command, $\mathbf{v_o}$, will be the nearest one to $\mathbf{v_G}$ in V_{free}. One situation is represented in Figure 4a. A clear optimal strategy is to choose $\mathbf{v_o} = \mathbf{v_G} = (v_{max}, 0)$. This strategy allows the robot to pass before the objects arrive and avoids the collision. Another situation is drawn in Figure 4b. In it DVS shows that $\mathbf{v_o} = \mathbf{v_G} = (v_{max}, 0)$, cannot be selected because it drives to collision. Thus, velocity $\mathbf{v_{MT}} = (v_{max}, w_{MT})$, the

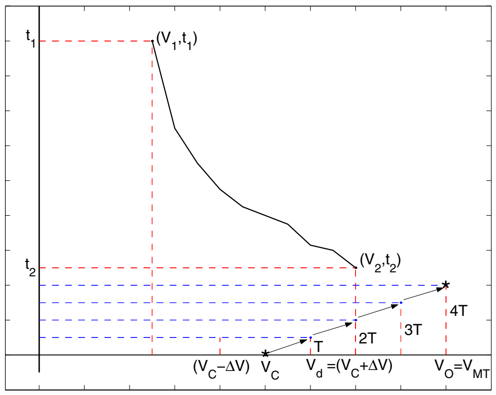


Fig. 5. The setpoint command $\mathbf{v_d}$ does not lead to collision at the current time, and allows to reach $\mathbf{v_o}$ in 4 steps ($4T$).

nearest one to $\mathbf{v_G}$, or $\mathbf{v_{SP}} = (v_{SP}, 0)$ can be chosen, being both velocities $\in V_{free}$ and $v_{SP} < v_{max}$, $w_{MT} \neq 0$. Velocity $\mathbf{v_{MT}}$ generates a circular trajectory around the moving obstacle, driving the robot at maximum linear velocity and then prioritizing minimum time. On the other hand, velocity $\mathbf{v_{SP}}$ generates a linear path, using a linear velocity lower than v_{max}. It yield a shorter path (rectilinear, waiting the object passes). The associated times t_{MT} and t_{SP}, implicitly included in DVS, can be also used to make the decision about the best velocity command.

Step 3. Computing the setpoint velocity $\mathbf{v_d}$
As explained above, the dynamic constraints limit the velocities the robot can reach in one sampling period. Thus the setpoint velocity $\mathbf{v_d}$ has to be inside the velocity window (see Figure 4b) every sampling period. Using the maximum velocity policy, $\mathbf{v_d}$ can be chosen as the velocity that quickly drives the robot to reach the velocity planned in the previous step, $\mathbf{v_o}$. Notice that $\mathbf{v_d}$ is not in V_{free}. But as the time to collision information is included in DVS, it can be verified that $\mathbf{v_d}$ is an admissible command. Figure 5 represents a slice (for $w = -0.1$) of DVS (Figure 2a). It can be seen that $\mathbf{v_d}$ is an admissible command inside the window velocity (sampling period) and $\mathbf{v_o}$ is reachable in 4 steps without collision, since both velocities are below the DOVS surface corresponding to potential collisions.

This process is resumed every sampling period, so the method allows to react to sudden changes in the environment, i.e. new obstacles and changes in the obstacles velocity (direction and module).

V. Experimental Results

We discuss in this section the results obtained by applying different strategies computed from the DVS. In the first experiment (Figure 6a) the strategy is to reach the goal at maximum velocity. In order to maintain the velocity as high as possible (the MT criterium), the system chooses commands that make the robot avoids the obstacles, moving around the obstacles at maximum linear velocity. In the second experiment (Figure 6b) the objective of the strategy

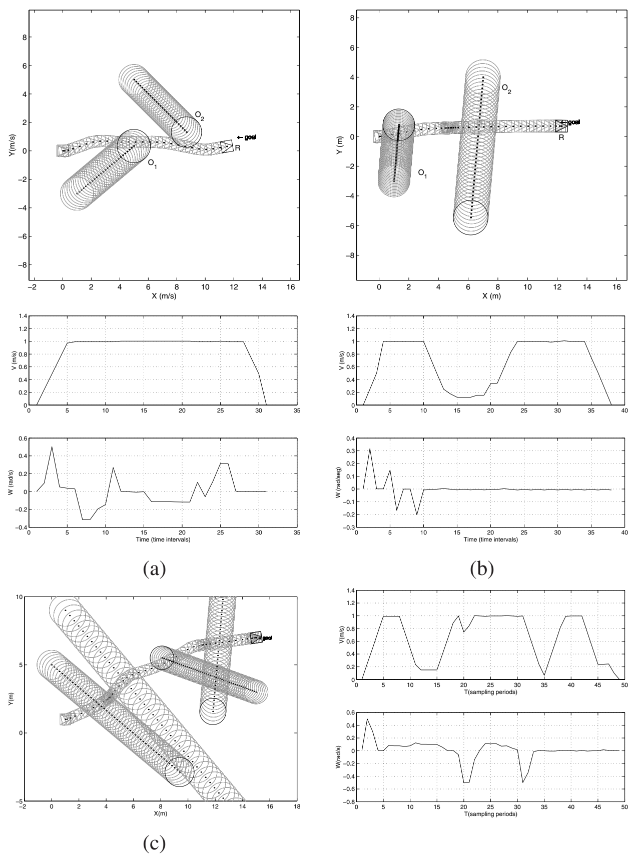


Fig. 6. (a) Experiment 1: the MT criterium and Velocity profiles (b) Experiment 2: the SP criterium and Velocity profiles (c) Experiment 3: navigation around four moving obstacles and profiles

is to follow the shortest path (near the straight line). This strategy results in a high velocity to pass before the object 1 arrives, but in a low velocity near the object 2, waiting the object 2 passes before the robot reach the collision band. As can be seen in Figure 6a the maximum linear velocity is chosen in the Velocity Space yielding a minimum time trajectory. In Figure 6b the robot follows a straight path, shorter than in previous experiment, but with a lower linear velocity.

The third experiment shows the robot behavior in a more complex scenario, with more objects moving around the robot (Figure 6c). Four moving objects cross in front of it. The robot has to manoeuver by speeding up (close to the first and third objects, from the left to the right) or slowing down (close to the second and fourth objects) when needed, in order to reach the goal without colliding and maintaining motions without oscillatory behaviors. This situation could be problematic for classical obstacle avoidance methods, producing oscillatory motions near the objects and therefore no optimal trajectories. It can be seen that to maintain the maximum linear velocity, the robot has to slightly turn to pass before the objects arrive.

VI. Conclusions

This paper presents a method for modelling the dynamism of the environment and the robot constraints, constructing the Dynamic Velocity Space (DVS). It is obtained by mapping the configuration space to a velocity space. This model allows to apply optimal strategies to compute the robot velocity commands directly in the velocity space, while avoiding the static and moving obstacles. For that objective a heuristic strategy has been presented, and simulation experiments showing the application of the method and the results of different strategies have been discussed. The design of more complex and multi-criteria strategies is an ongoing work.

Ackowlegements. This work was partially supported by MCYT DPI2003-7986 Spanish project.

References

[1] M. Erdmann and T. Lozano-Perez. On multiple moving objects. Technical report, Massachusetts Institute of Technology, 1986.

[2] P. Fiorini and Z. Shiller. Robot motion planning in dynamic environments. In *International Symposium of Robotic Research*, pages 237–248, 1995.

[3] A. Foca and P. Trahanias. Predictive control of robot velocity to avoid obstacles in dynamic environments. In *Int. Conf. on Intelligent Robots and Systems*, 2003.

[4] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997.

[5] T. Fraichard. Trajectory planning in a dynamic workspace: a state-time-space approach. *Advanced Robotics*, 13(1), 1999.

[6] T. Fraichard and C. Laugier. Kynodinamic planning in a structured and time-varing 2d workspace. *International Journal of Robotic Research*, 5(3):72–99, 1986.

[7] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles. *IEEE Trans. on Robotics and Automation*, 5(1):61–69, 1989.

[8] K. Fujimura and H. Samet. Time-minimal paths among moving obstacles. *IEEE Trans. on Robotics and Automation*, pages 1110–1115, 1989.

[9] K. Kant and S. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *International Journal of Robotic Research*, 5(3):72–99, 1986.

[10] Z. S. F. Large and S. Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *IEEE Int. Conf. on Robotics and Automation*, pages 3716–3721, Seoul, Korea, 2001.

[11] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Press, 1990.

[12] J. MInguez and L. Montano. Nearness diagram(nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Trans. on Robotics and Automation*, 20(1):45–59, 2004.

[13] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *J. ACM*, 41(4):764–790, 1994.

[14] N. Roy and S. Thrun. Motion planning through policy search. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[15] R. Simmons. The Curvature-Velocity Method for Local Obstacle Avoidance. In *IEEE Int. Conf. on Robotics and Automation*, pages 3375–3382, Minneapolis, USA, 1996.

[16] A. Stentz. Optimal and efficient path planning for partially-known environments. In *IEEE Int. Conf. on Robotics and Automation, Vol. 4*, pages 3310–3317, 1994.

[17] I. Ulrich and J. Borenstein. VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In *IEEE Int. Conf. on Robotics and Automation*, pages 2505–2511, San Francisco, USA, 2000.