Robotica (2001) volume 19, pp. 543–555. Printed in the United Kingdom © 2001 Cambridge University Press

Collision-free path planning for nonholonomic mobile robots using a new obstacle representation in the velocity space

Gabriel Ramírez and Saïd Zeghloul

Laboratoire de Mécanique des Solides – UMR 6610 CNRS, Université de Poitiers, SP2MI, Bd. Marie et Pierre Curie, Téléport 2, BP 179, 86962 Futuroscope CEDEX (France)

(Received in Final Form: March 15, 2001)

SUMMARY

This paper presents a collision-free path planner for mobile robot navigation in an unknown environment subject to nonholonomic constraints. This planner is well adapted for use with embarked sensors, because it uses only the distance information between the robot and the obstacles. The collision-free path-planning is based on a new representation of the obstacles in the velocity space. The obstacles in the influence zone are mapped as linear constraints into the velocity space of the robot, forming a convex subset that represents the velocities that the robot can use without collision with the objects. The planner is composed by two modules, termed "*reaching the goal*" and "*boundary following*". The major advantages of this method are the very short calculation time and a continuous stable behavior of the velocities. The results presented demonstrate the capabilities of the proposed method for solving the collision-free path-planning problem.

KEYWORDS: Collision avoidance; Mobile robots; Nonholonomic systems.

1. INTRODUCTION

The autonomous navigation of indoor wheeled mobile robots has been the center of attention of the scientific community in recent years, motivated by applications such as cargo delivery, office cleaning and assistance to disabled people, among others. In order to achieve its objectives, the mobile robot must be able to plan a collision-free trajectory among the obstacles of the environment. Planning a path for a mobile robot means to find a continuous trajectory that leads the mobile robot from the initial position to the goal position.

The two main approaches for solving the path-planning problem are global and local methods. The global methods need complete information of the environment and they are based on a global world representation. Examples of this approach are the grid of visits,[1] the numeric potential fields,[2] the graphs using the A* algorithm (graphs of visibility,[3] of tangents,[4] Voronoï[5]) the behavior-based models[6] and the genetic algorithms,[7] among others. These approaches guarantee that either the goal position will be reached, thus the path planning problem has a solution, or the methods will entirely analyze the free space and conclude that the goal position is unreachable. In this latter case, the path planning problem has no solution. However, the construction and maintenance of a global model require heavy computational work, as well as a great amount of memory resources.

On the other hand, the local path planners use only the local information in a purely reactive manner. At every control cycle, the robot performs the action corresponding to the configuration of the obstacles located in the neighborhood. The local path planners are typically simpler than the global ones, since they can directly map the sensor readings to actions. Different methods are used to choose the next action according to the local environment: the potential fields,[8] the Bug algorithm,[9] the behavior-based models,[10] the probabilistic models (Virtual Field Histogram,[11] occupancy grids[6]) the fuzzy logic,[12,13] etc. Although the local planners are simpler to implement, they do not guarantee global convergence to the goal position. The mobile robot may get trapped in a local minimum, and the method will follow a divergent path or a loop while attempting to escape from the deadlock condition.

In conclusion, the global approaches suffer from practical problems for their implementation, while the local planners do not guarantee a solution to the path-planning problem. Furthermore, almost all these planners consider the mobile robot as a point with holonomic properties.

For the local path planner proposed in this paper, the robot and the obstacles are modeled by convex polygons and the nonholonomic constraints of the robot are considered. This work is focused to differential mobile robots (robots equipped with two parallel driving wheels and no steering wheel).

The proposed method consists of two different modules of motion, termed "*reaching the goal*" and "*boundary following*", and the transition conditions for switching between them in order to ensure global convergence to the goal.[14] This work is inspired by the Bug algorithms,[9,15,16] applied to point holonomic mobile robots.

Since the considered mobile robot is a nonholonomic system, it cannot be stabilized on a fixed position using a time-variant, smooth feedback control law.[17] In Section 3, we present a nonlinear exponential control law, which makes the goal position globally exponentially stable. This control law allows the robot to navigate in a free environment (Section 4), and it is used as a reference velocity in the collision-free path-planning algorithm presented in Section 5.

We introduce a new representation of the obstacles in the robot's velocity space, termed the "*Feasible Velocities Polygon*" (or FVP) in Section 5. The nearby obstacles are

mapped as linear constraints on the robot's velocity space, using a *velocity damper* model,[18] as function of the distance between the robot and the obstacle. The resulting set of linear constraints, in addition with the velocity limit constraints, defines a convex polygon in the velocity space. This convex subset, the *feasible velocities polygon*, describes the pairs of velocities (ν, ω) that allow the robot to avoid any collision with the obstacles. In addition, the use of velocities (ν, ω) leads to respecting the nonholonomic constraint of the robot. This representation is used for the two modules to ensure the collision-free navigation.

For the first module, "*reaching the goal*", we use a formulation similar to the one proposed by Faverjon and Tournassaud for manipulators with a high number of degrees of freedom.[18] This approach minimizes the deviation of the robot's current trajectory from a straight line, in the joint variable space, between the initial position and the final position, under the obstacle constraints. For the navigation problem of mobile robots, the optimization process is represented by a minimal distance calculation problem, between the *feasible velocities polygon* and the point describing the reference velocity.

Therefore, the *reaching the goal* module allows the mobile robot to continuously move towards the goal position, following the reference velocity under the obstacle constraints. Because the first module is based on a local optimization process, the robot may get trapped in a local minimum point, for example a U-shape obstacle. In this case, the resulting pair (ν, ω) is equal to zero, and the robot cannot move anymore towards the goal.

When the deadlock situation is detected, the second module, "*boundary following*" is applied. This module allows the robot to follow the obstacle boundary, using the *feasible velocities polygon*, in order to escape from the blocking point. The process is achieved by tracking the evolution of the obstacle constraints that represent the blocking obstacles.

Using the second module, the robot follows the obstacle boundary while estimating the distance to the goal. When a "closer" point than the blocking one is detected, the obstacle has been circumvented and the method switches to the first module again. A numeric example of the two behaviors is shown in Section 6.

In conclusion, the transition conditions ensure that any new deadlock situation is closer to the goal than the preceding one. Thus, the global convergence to the goal is guaranteed.

2. KINEMATIC MODEL OF A DIFFERENTIAL MOBILE ROBOT

The configuration of a mobile robot in a plan (see Figure 1) can be completely defined by the following state vector:

$$\mathbf{q} = [x \;\; y \;\; \theta]^T$$

where (x, y) is the position of a fixed point on the robot and θ the orientation of the frame linked to the robot with respect to the X-axis. We have chosen the center R of the wheel axis as the fixed point on the robot. The linear velocity ν along the X_r axis and the angular velocity ω of the robot are given by:

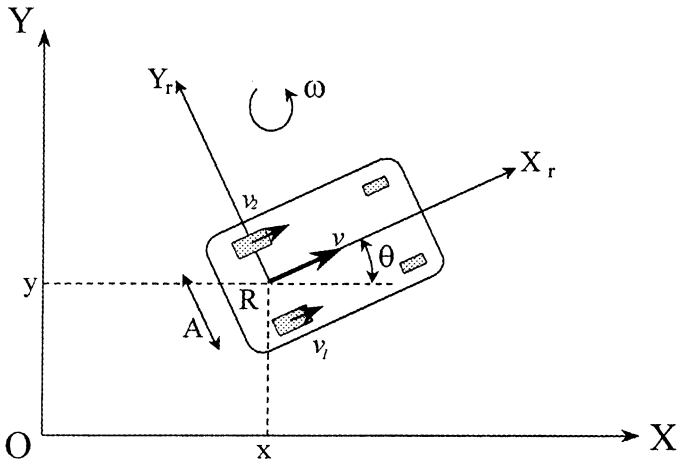


Fig. 1. A differential mobile robot.

$$\nu = \frac{\nu_1 + \nu_2}{2}, \quad \omega = \frac{\nu_1 - \nu_2}{A}$$

where ν_1 and ν_2 are respectively the right wheel and the left wheel velocities. The kinematic model of the mobile robot can be written as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix}; \qquad \text{thus } \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\mathbf{u}$$

where $\mathbf{J}(\mathbf{q})$ is the Jacobean matrix of the robot, and $\mathbf{u} = [\nu, \omega]^T$ is the input vector (or control vector). Nonholonomic systems are systems with nonintegrable velocity constraints. For this system, the nonholonomic constraint is given by:

$$\dot{y} - \dot{x}\tan\theta = 0$$

3. CONTROL LAW

The nonholonomic systems have been the center of attention in the nonlinear control community in recent years. These systems cannot be stabilized to the equilibrium point using a time-invariant, smooth (or even continuous) feedback control law.[17] Thus, most existing results from linear and nonlinear systems theory are not directly applicable in this case. Time-varying control laws can be used but such control laws may have slow rates of convergence. Fast convergence rates typically necessitate non-smooth/non-continuous control laws.

The proposed control law is a piecewise smooth state feedback, which makes the final position globally exponentially stable. The final robot's orientation is not taken into account for the construction of the control law. Sordalën and Canudas, have developed a similar control law in which the final orientation is considered.[19]

Let $\mathbf{q}_f = [x_f, y_f, 0]^T$ be the desired goal position (regardless of the final orientation) and $\mathbf{q} = [x, y, \theta]^T$ the robot's current position, as shown in Figure 2. We define the deviation vector $\mathbf{q}_e$ as follows:

$$\mathbf{q}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \mathbf{q}_f - \mathbf{q} = \begin{bmatrix} x_f - x \\ y_f - y \\ -\theta \end{bmatrix}$$

and by derivation, we obtain:

$$\dot{\mathbf{q}}_e = -\mathbf{J}(\mathbf{q})\mathbf{u}$$

We define the error vector $\mathbf{z}(t)$ as the polar coordinates (a, α) of the goal with respect to the robot's frame, as follows:

$$\mathbf{z}(t) = \begin{bmatrix} a \\ \alpha \end{bmatrix}$$

where:

$$a = \sqrt{x_e^2 + y_e^2}$$

$$\alpha = \arctan 2(y_e, x_e) + \theta_e \quad \alpha \in [-\pi, \pi]$$

Therefore, if $\alpha = 0$, the robot is pointing to the goal position, and if $a = 0$ the goal has been reached. Because a and α are continuous everywhere (except α in $x_e = 0$, $y_e = 0$), the vector $\mathbf{z}$ is derivable. The gradient of $\mathbf{z}$ is given by:

$$\mathbf{J}_z(\mathbf{q}_e) = \nabla \mathbf{z}(\mathbf{q}_e) = \begin{bmatrix} \cos(\alpha - \theta_e) & \sin(\alpha - \theta_e) & 0 \\ -\dfrac{\sin(\alpha - \theta_e)}{a} & \dfrac{\cos(\alpha - \theta_e)}{a} & 1 \end{bmatrix}$$

We have, by derivation of the vector $\mathbf{z}$:

$$\dot{\mathbf{z}} = \mathbf{J}_z(\mathbf{q}_e)\dot{\mathbf{q}}_e = -\mathbf{J}_z(\mathbf{q}_e)\mathbf{J}(\mathbf{q}_e)\mathbf{u}$$

$$\dot{\mathbf{z}} = \begin{bmatrix} -\cos\alpha & 0 \\ \dfrac{\sin\alpha}{a} & -1 \end{bmatrix} \mathbf{u}$$

So, we obtain:

$$\dot{a} = -\nu \cos\alpha$$

$$\dot{\alpha} = \frac{1}{a}\nu \sin\alpha - \omega$$

We propose the following control law, which leads to exponential convergence:

$$\nu = k_1 a \cos\alpha$$

$$\omega = k_2 \alpha + k_1 \sin\alpha \cos\alpha$$

with $k_1, k_2 > 0$. The robot's velocities (ν, ω) are continuous in the space (a, α). Furthermore, these velocities vanish only at the origin $(a = 0, \alpha = 0)$.

The closed loop system becomes:

$$\dot{a} = -(k_1 \cos^2\alpha)a$$

$$\dot{\alpha} = -k_2\alpha$$

This system has only an equilibrium point at the origin of the space (a, α). Indeed, $\dot{\alpha} = 0$ implies that $\alpha = 0$; and on the other hand, $\dot{a} = 0$ with $\alpha = 0$ implies that $a = 0$. Thus, the robot only stops at the origin.

In order to prove that the origin is globally exponentially stable, let's take the following candidate Lyapunov function:

$$\mathbf{V}(\mathbf{z}) = \frac{1}{2}a^2 + \frac{1}{2}\alpha^2$$

This function is continuous and differentiable at any point (a, α), $\mathrm{V}(\mathbf{0}) = 0$ and $\mathrm{V}(\mathbf{z}) > 0$ otherwise. The time derivative of $\mathrm{V}(\mathbf{z})$ is given by:

$$\dot{\mathbf{V}}(\mathbf{z}) = -k_1 a^2 \cos^2\alpha - k_2\alpha^2$$

Because $\dot{\mathrm{V}}(\mathbf{z}) < 0 \ \forall \mathbf{z} \neq \mathbf{0}$, the function $\mathrm{V}(\mathbf{z})$ is a Lyapunov function of the system (see appendix):

$$\dot{a} = -(k_1 \cos^2\alpha)a$$

$$\dot{\alpha} = -k_2\alpha$$

and the origin $(a = 0, \alpha = 0)$ is globally exponentially stable. The distance between the robot and the goal, $a(t)$, globally exponentially converges to zero, and $a = 0$ implies that the robot has reached the final position.

4. NAVIGATION IN A FREE ENVIRONMENT

For the initial position $\mathbf{q}_i = [4, -4, 0]^T$ and the goal position located at the origin $\mathbf{q}_f = [0, 0, 0]^T$ we obtain the path shown

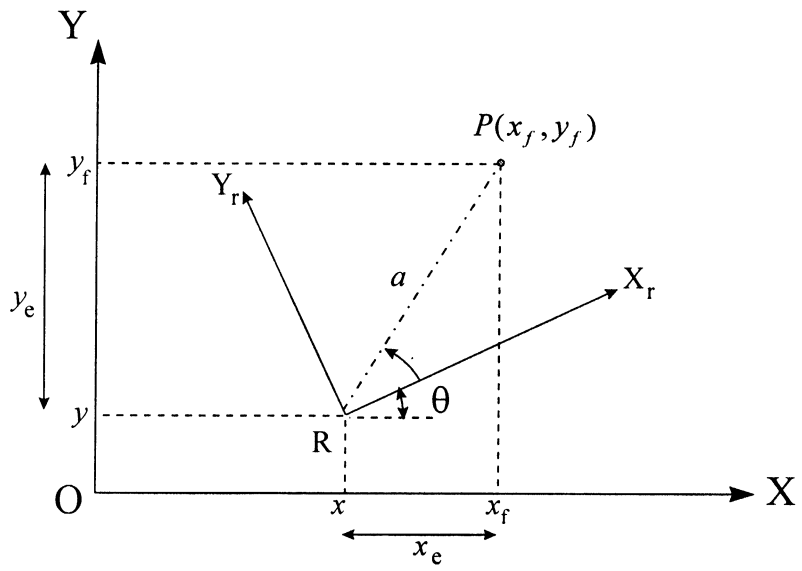


Fig. 2. Error definition.

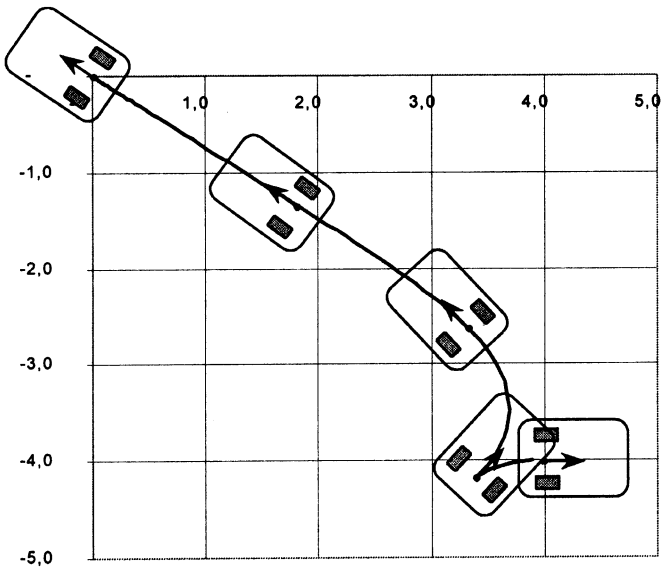


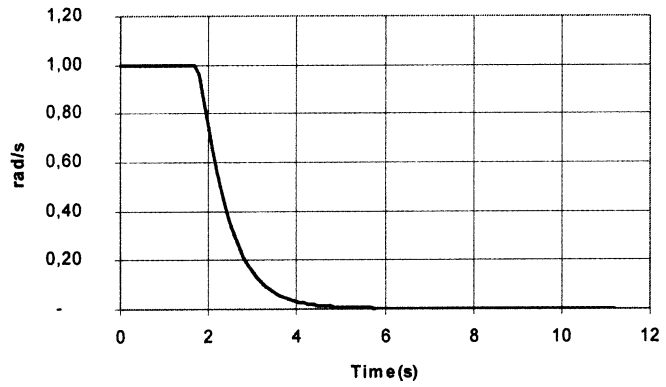
Fig. 3. Robot's trajectory.

in Figure 3, using the control law described above. For this this example, the coefficients k_1, k_2 are set to 0.6.

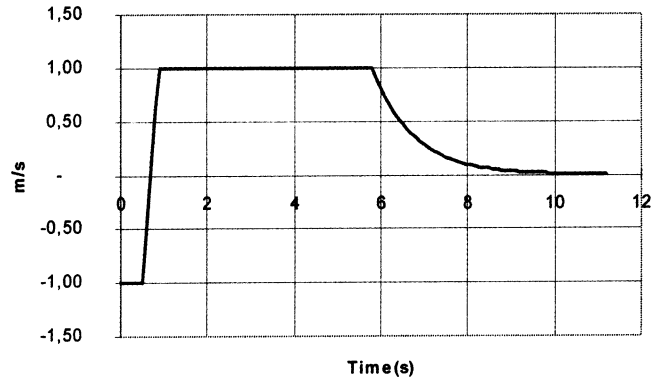
Figure 4 shows the robot's velocities evolution. The velocity limits are the following:

$$v_{\max} = 1.0 \text{ m/s}, \ \omega_{\max} = 1.0 \text{ rad/s}$$

We can observe that the velocities are continuous everywhere, except in $t = 0$.



(a) Angular velocity



(b) Linear velocity

Fig. 4. Robot's velocities.

5. COLLISION-FREE PATH PLANNING

5.1 "Reaching the goal" module

Faverjon and Tournassaud have proposed a substitute of potential field method to find a collision-free trajectory in a cluttered environment, for a manipulator robot with a high number of degrees of freedom.[18] This approach is called the "*constraints method*", because the obstacles are mapped as linear constraints over the robot's joint velocities (Figure 5). The method is based on an iterative scheme where the configuration vector is found by using the following formula:

$$\mathbf{q}_i = \mathbf{q}_{i-1} + \Delta t \, \dot{\mathbf{q}}$$

The vector $\dot{\mathbf{q}}$, the joint velocities vector, is obtained as a solution of the following optimization problem:

Minimize: $$f = \frac{1}{2} \| \dot{\mathbf{q}} - \dot{\mathbf{q}}_{goal} \|^2$$

under the obstacle constraints given by the velocity damper *model*:

$$\dot{d} \geq -\xi \frac{d - d_s}{d_i - d_s}$$

where:

- $\mathbf{q}$ and $\mathbf{q}_f$ are respectively the current and final configuration vectors,
- $\dot{\mathbf{q}}_{goal}$ is the vector of desired joint velocities, calculated in order to obtain a straight line in the joint space,
- d is the minimal distance between the robot and the considered object,
- d_i is the influence distance from which a constraint becomes active,
- d_s is the security distance that must be respected,
- ξ, represents a coefficient in order to adapt the convergence speed.

The obstacle constraint expresses that the distance between the robot and the obstacle must not decrease too fast when it is less than the influence distance d_i. It is easy to prove that if the initial distance is greater than d_s, then

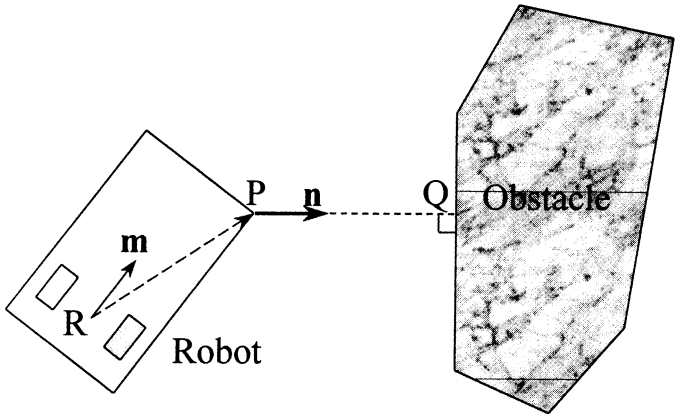


Fig. 5. Establishing the obstacle constraints.

there cannot be any collision between the robot and the object.

The first module of our method uses a similar approach. The optimization problem can be stated as follows:

Find a control vector $\mathbf{u}=[\nu, \omega]^T$ *that minimizes the following function*

$$f_r=\frac{1}{2}\|\mathbf{u}-\mathbf{u}_{goal}\|^2$$

where

$$\mathbf{u}_{goal}=\begin{bmatrix} k_1 a\cos\alpha \\ k_2\alpha+k_1\sin\alpha\cos\alpha \end{bmatrix}$$

is the velocity vector obtained from the exponential control law described in Section 3.

For any obstacle located at a distance d less than d_i, we can express the *velocity damper* constraint as function of the components ν and ω of the control vector **u**. Let **m** be the unit vector along the robot axis X_r, and **n** the unit vector along the segment PQ of minimal distance d between the robot and the involved obstacle, as shown in Figure 5. The time derivative of d is given by:

$$\dot{d}=-\mathbf{n}\cdot\mathbf{v}_P$$

where $\mathbf{v}_P$ is the robot's velocity evaluated at the point P, given by the following expression:

$$\mathbf{v}_P=\nu\mathbf{m}+\omega\hat{k}\times\overrightarrow{RP}$$

So, the obstacle constraint that represents the considered obstacle is given by the following expression:

$$(\nu\mathbf{m}+\omega\hat{k}\times\overrightarrow{RP})\cdot\mathbf{n}\le\xi\frac{d-d_s}{d_i-d_s}$$

The set of obstacle constraints and the velocity limit constraints are linear in (ν, ω), thus they define a convex subset in the (ν, ω) space. Because this subset defines the bi-dimensional set of velocities (control vectors) that the robot can use without collision with the obstacles, we call it the "*Feasible Velocities Polygon*" or FVP.

For example, let's consider the situation shown in Figure 6a. The obstacle C is at a greater distance than d_i from the robot, so only objects A and B are considered in the calculation process, i.e. they are mapped as linear constraints over the robot's velocity. If no obstacle is considered, the FVP is built only by the velocity limit constraints, as shown in Figure 6b. The object A, located to the right of the robot, is mapped as an additional constraint on the FVP. This new constraint is shown in Figure 6c, it prevents the robot to turn to the right and collide with the obstacle A. The Figure 6d shows the FVP when the two obstacles, A and B, are mapped into the velocity space. We obtain the convex subset in which any velocity used by the mobile robot ensures non-collision with the obstacles. The reference velocity to reach the goal in a free environment is represented by the point $\mathbf{u}_{goal}$.

We can observe that the minimization problem

$$\min_{\mathbf{u}} f_r=\frac{1}{2}\|\mathbf{u}-\mathbf{u}_{goal}\|^2$$

subject to the linear constraints

$$(\nu\mathbf{m}+\omega\hat{k}\times\overrightarrow{RP})\cdot\mathbf{n}\le\xi\frac{d-d_s}{d_i-d_s}$$

$$|\nu|\le\nu_{\max}$$

$$|\omega|\le\omega_{\max}$$

is a minimal distance calculation problem between the FVP and the reference point $\mathbf{u}_{goal}$. The solution to this problem is represented by the point $\mathbf{u}^*$, which is the closest point on the polygon to $\mathbf{u}_{goal}$, as shown in Figure 6d.

In order to compute the obstacle constraints, it is necessary to know the distance between the robot and the obstacles. In simulation, the distance d between the robot and any of the obstacles is calculated using a fast minimal distance calculation algorithm, developed by S. Zeghloul and P. Rambaud.[20] In the real case, the distance d is measured by the distance sensors, such as ultrasonic sensors or laser sensors.

For real and simulation cases, the optimization problem is solved as a minimal distance calculation problem between the FVP and the reference velocity. The distance calculation algorithm proposed by S. Zeghloul and P. Rambaud[20] is used to find the solution.

When the solution vector $\mathbf{u}^*$ has been calculated, the variation of the robot's configuration vector is given by:

$$\Delta\mathbf{q}_i=\Delta t\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\mathbf{u}^*$$

The first module of the path planner, called *reaching the goal*, is defined by the following iterative process:

While ($\mathbf{u}^*\neq\mathbf{0}$)

i. compute the reference velocity $\mathbf{u}_{goal}$, using the exponential control law;
ii. build the FVP, by mapping the obstacles in the influence zone as linear constraints over the robot's velocities; this process uses the distance information between the robot and the obstacles. In simulation, a fast distance calculation algorithm calculates the distance. For the real robot, the distance is measured by distance sensors;
iii. solve the optimization problem by minimal distance calculation;
iv. apply vector $\mathbf{u}^*$.

This module allows the robot to solve simple situations where the obstacle configuration does not create local minima points. An example is shown in Figure 7.

While the robot uses the *reaching the goal* module, the *distance* function:

$$\mathbf{V}(\mathbf{z})=\frac{1}{2}a^2+\frac{1}{2}\alpha^2$$

is strictly decreasing, since the robot is continuously approaching the goal – this *distance* function is the same

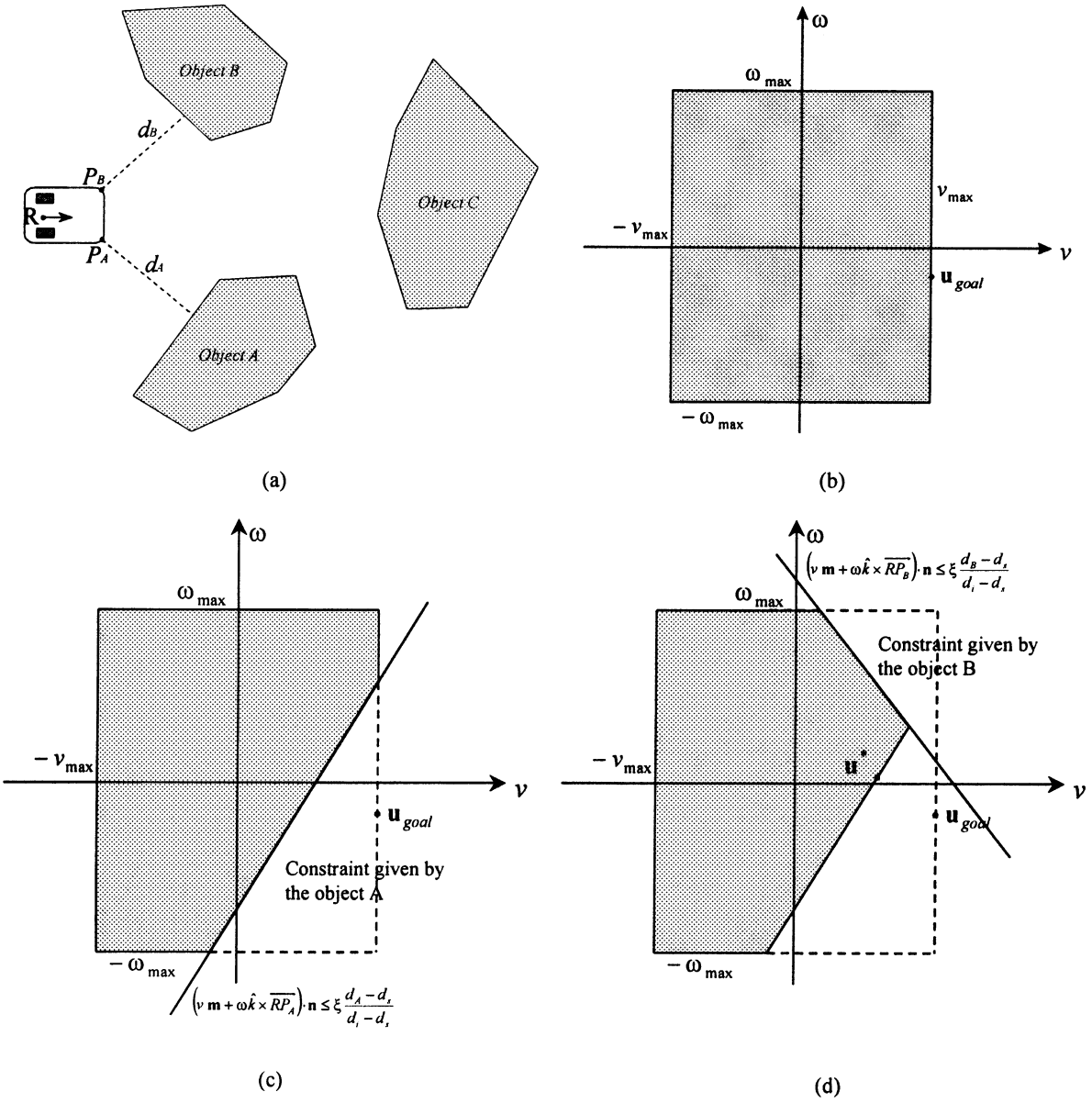


Fig. 6. (a) Obstacle configuration, (b) Velocity limit constraints, (c) mapping of the object A, (d) Feasible Velocities Polygon.

Lyapunov function used to prove the stability of the exponential control law. This property is used in the second module to solve the deadlock situations.

When the solution of the optimization problem, **u***, is located at the origin of the plan (v, ω) but the *distance* function is not zero, the robot cannot move anymore towards the goal using the *reaching the goal* module. This situation corresponds to a deadlock condition, which is a typical behavior of the local methods. In these circumstances the first module gives the null vector as solution; so the second module, termed *boundary following*, is applied in order to solve the deadlock situation.

5.2 "Boundary following" module

The *boundary following* module and its interaction with the reaching the goal module have been inspired by the Bug algorithms.[9,15,16] The module uses the FVP representation and the distance function to allow the robot to follow the obstacle boundary, in order to escape from the deadlock condition.

According to the location on the FVP of $\mathbf{u}^*=\mathbf{0}$, we can define two types of deadlock situations: the *single-obstacle deadlock*, which occurs when **u*** is located over an edge of the FVP, and the *two-obstacle deadlock*, when **u*** is located at a vertex of the FVP. These situations are described in Figure 8a and Figure 8b, respectively.

Regardless of the number of obstacles involved in the blocking situation, the solving process is the same for both cases. First, when a deadlock situation is detected ($\mathbf{u}^*=\mathbf{0}$), the current value of the *distance* function, V_{block}, is recorded. This value is used to determine when the blocking obstacle has been circumvented, and the module *reaching the goal* must be applied again.

At the same time, the configuration of the blocking obstacles is analyzed in order to find a "*search direction*" (clockwise or counterclockwise), that allows the robot to circumvent the obstacles. This parameter is established by

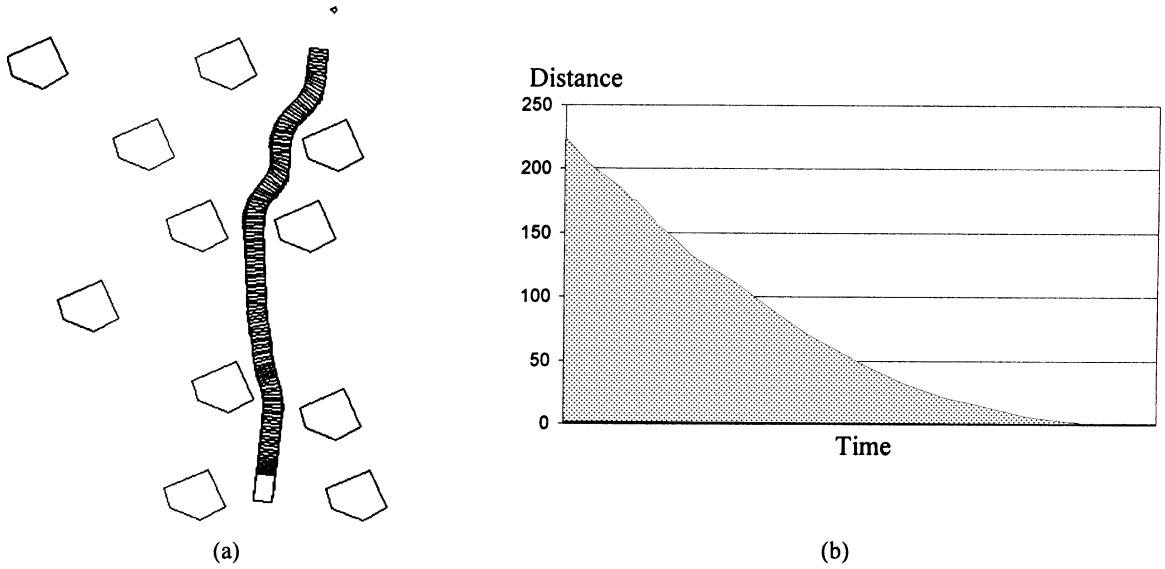


Fig. 7(a) A simple example, (b) Distance function.

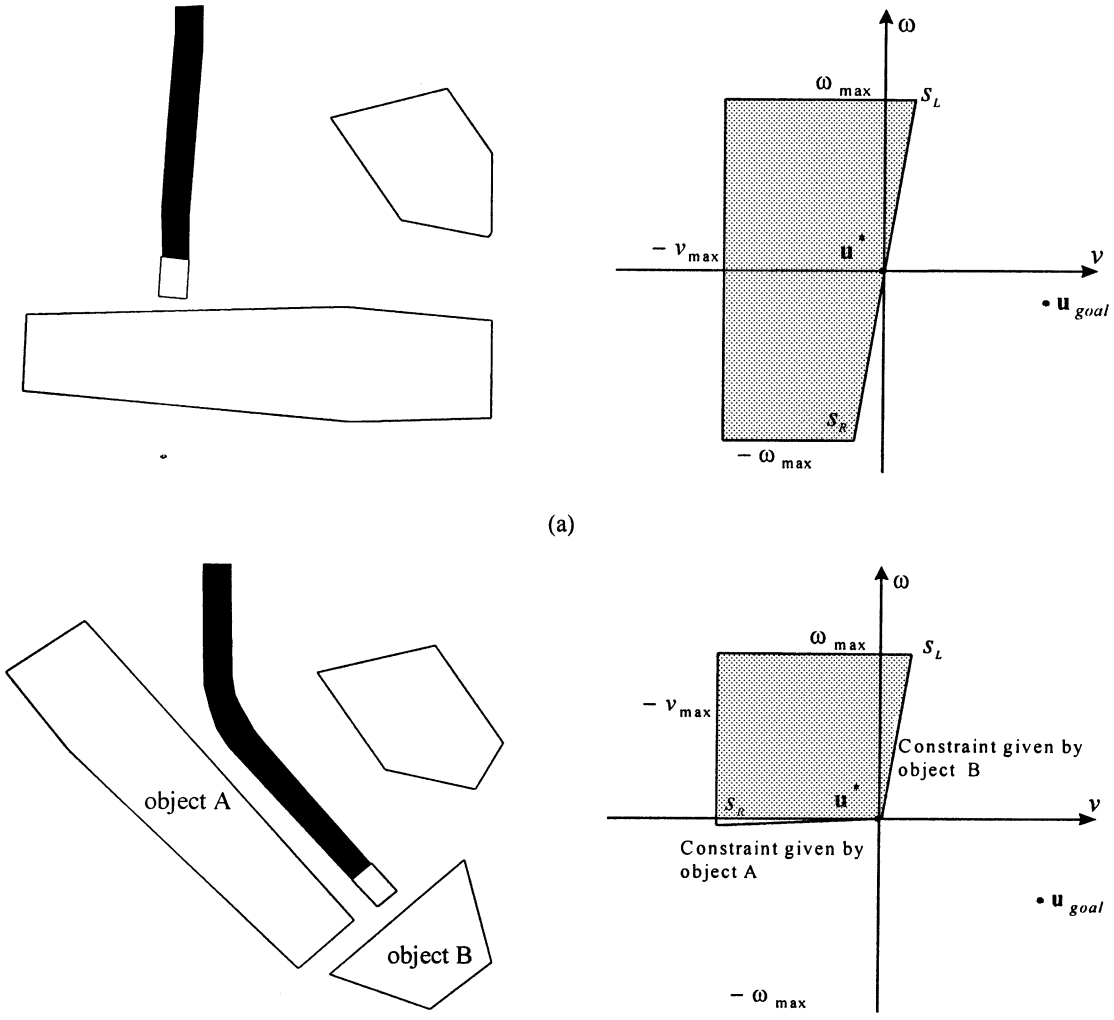


Fig. 8(a) Single-obstacle dead-lock, (b) Two-obstacles dead-lock.

the relative position of the blocking obstacles with respect to the mobile robot's position, considering that no traffic rules are defined. For example, in the single-obstacle deadlock case shown in Figure 9, the minimal distance between the robot and the obstacle is realized at the point P. Because the point P is located on the left side of the robot, locally, the best direction to circumvent the obstacle is counterclockwise, and this is the value given to the *search direction* parameter.

Also, a "*constraint to follow*" is identified on the FVP. This constraint corresponds to the obstacle to be circumvented by the robot. This parameter is needed to track the evolution of the relative position between the robot and the obstacle in order to follow the obstacle boundary.

When the "*search direction*" and "*constraint to follow*" parameters have been established, the mobile robot begins to circumvent the obstacle. The method tracks the evolution of the *constraint to follow* as long as the robot moves, by a comparative analysis of the current FVP with the previous

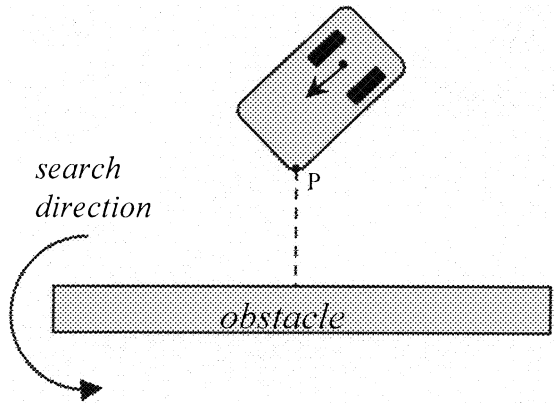


Fig. 9. Choosing the *search* direction.

one (the FVP of the preceding control cycle). The value of *search direction* defines which of the two vertices on the *constraint to follow* must be applied as the velocity vector: s_R for a clockwise direction or s_L for a counterclockwise

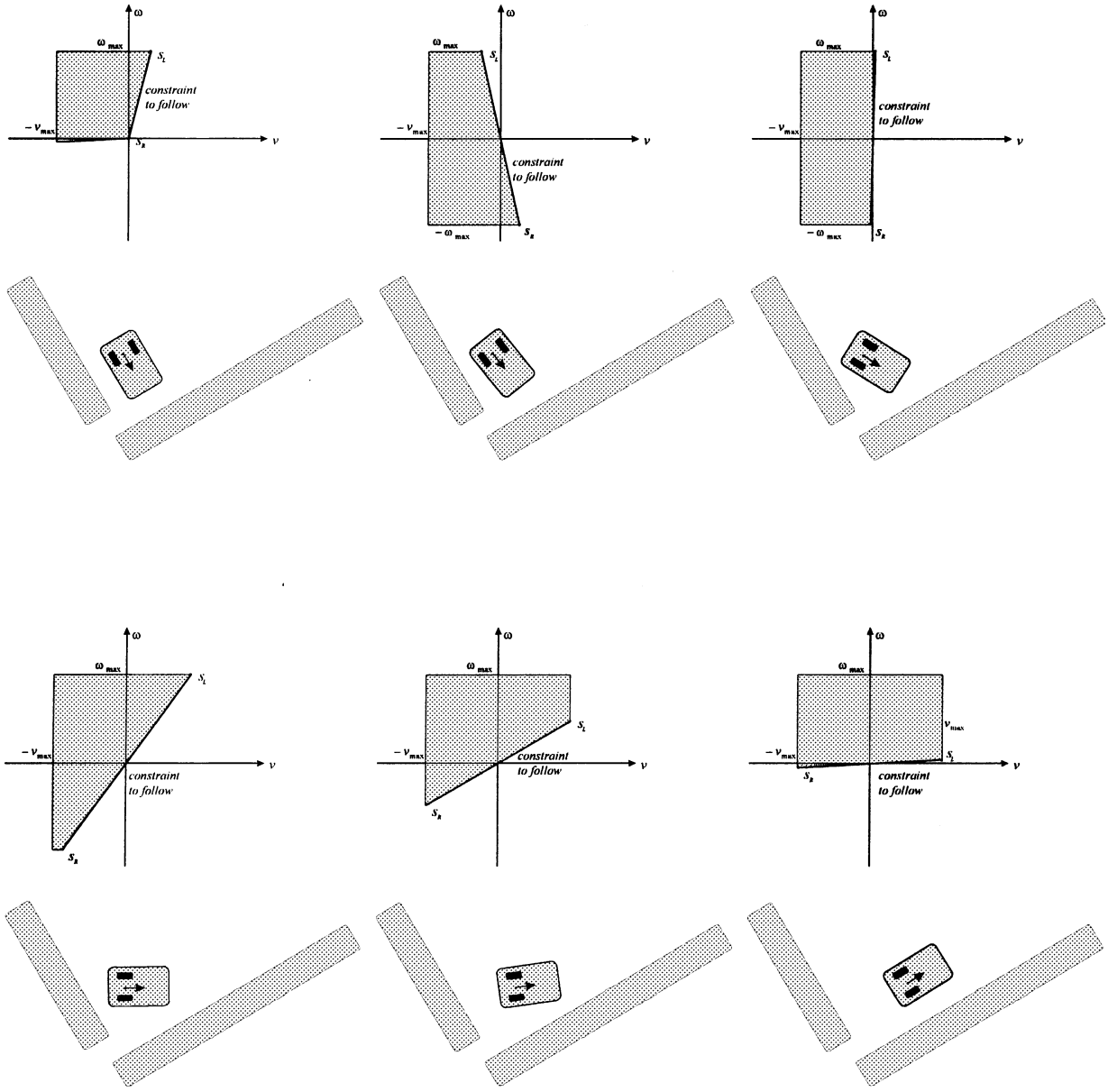


Fig. 10. *Boundary following* module.

direction. This process allows the robot to follow the obstacle boundary at a distance d_s, as shown in Figure 10.

It could happen that, while the robot is following the obstacle boundary, a new obstacle blocks its trajectory. This condition is perceived as a new constraint on the FVP that leads the chosen vertex (established by the *search direction* and *constraint to follow* parameters) to the origin of the plan (v, ω). When, under the new constraint, the chosen vertex vanishes ($\mathbf{u} = \mathbf{0}$), we can be sure that the distance between the new obstacle and the followed obstacle is not large enough to allow the robot to pass. In this situation, the robot begins to circumvent the new obstacle by switching the value of *constraint to follow* to the new constraint, without modifying the value of the *search direction* parameter.

The robot follows the boundary of the blocking obstacles until it finds a point that is closer to the goal than the deadlock point defined above. To achieve this, the value of the distance function V(**z**) is continuously compared with the value V_{block} while the robot is circumventing the obstacles. Because the function V(**z**) is a measure of distance between the robot and the goal position, when the value of the V(**z**) is less than V_{block}, the robot has found a point, over the convex obstacle boundary, that is closer to the goal than the deadlock point. When such a point is found, the blocking obstacle has been circumvented and the robot switches from the *boundary following* module to the *reaching the goal* module, in order to move towards the goal.

In conclusion, if the size of the blocking obstacles is finite, the robot will be able to circumvent them in a finite time and will find a point that is closer to the goal than the deadlock point. When the closer point is found, the blocking obstacles have been circumvented and the robot continues its way towards the goal.

6. NUMERICAL EXAMPLE

To illustrate the behavior of the two modules and their interaction, let's consider the situation shown in Figure 11, with an initial distance value V(**z**) = 250.0

As shown in Figure 11, the robot, using the *reaching the goal* module, will move towards the final position following a stable trajectory, until the robot reaches a two-obstacle deadlock situation. At this point, $\mathbf{u}^* = \mathbf{0}$ and the current value of the distance function, $V_{block} = 72.34$, is recorded.

For the application of the *boundary following* module, it is necessary to determine the value of *search direction*. For the actual obstacle configuration, the robot must circumvent the obstacles in the counterclockwise direction. Also, for this two-obstacle deadlock, the *constraint to follow* is identified.

Thereafter, the boundary following module tracks the evolution of the *constraint to follow* edge on the FVP. At every control cycle, the vertices s_R and s_L on this constraint are identified. Because the robot is circumventing the obstacles in the counterclockwise direction, the velocities represented by the vertex s_R are applied to the robot's movement. Since the control vector **u** leads to the obstacle constraint, the robot follows the boundary of the obstacle at a distance d_s .

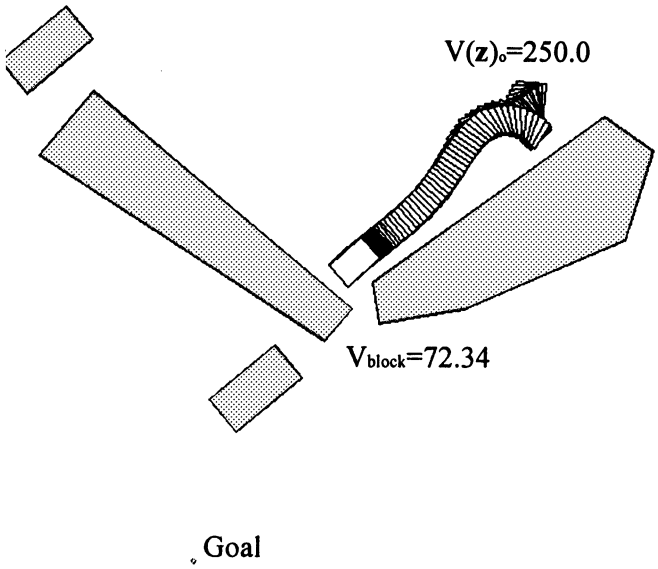


Fig. 11. A "dead-end" example.

While the robot is following the obstacle boundary, a new obstacle has entered in the influence zone; thus a new constraint is mapped into the velocities space. Because the distance between the obstacles is not large enough to allow the robot to pass, the new constraint makes that the chosen vertex, s_R, converges to the origin of the (v, ω) plan, leading the robot to a new deadlock situation, as shown in Figure 12. In order to solve this new deadlock, the method switches the followed constraint to the new one and the value of *search direction* is kept, thus the robot can circumvent the new obstacle. So, the new chosen vertex is s'_R.

Every time that a new obstacle prevents the robot from following the boundary of the circumvented object, the method switches the followed constraint, so the robot can circumvent this blocking obstacle, as shown in Figure 13a.

While the robot is circumventing the blocking obstacles, the value of the *distance* function is compared with the deadlock value, V_{block}. When the current value of V(**z**) is less than V_{block}, the algorithm has found a point of the obstacle boundary that is closer to the goal than the deadlock point (Figure 13b). At this moment, the *reaching the goal* module is activated again, and the robot continues its movement until the goal is reached, as shown in Figure 14a. The Figure 14b shows the evolution of the function V(**z**) for the whole trajectory.

7. GLOBAL CONVERGENCE TO THE GOAL

In order to show that the proposed path planner ensures the global convergence of the mobile robot to the final position, let's consider the situation shown in Figure 15a, *getting out from a labyrinth*.

The resulting trajectory is a sequence of *reaching the goal* (*RG*) and *boundary following* (*BF*) intervals. We can observe in Figure 15b that, for the *reaching the goal* intervals, the function V(**z**) is strictly decreasing following exponential shapes, because the reference vector used, $\mathbf{u}_{goal}$, is obtained from the exponential control law described in Section 3. This control law leads to a stable behavior of the robot's velocities.

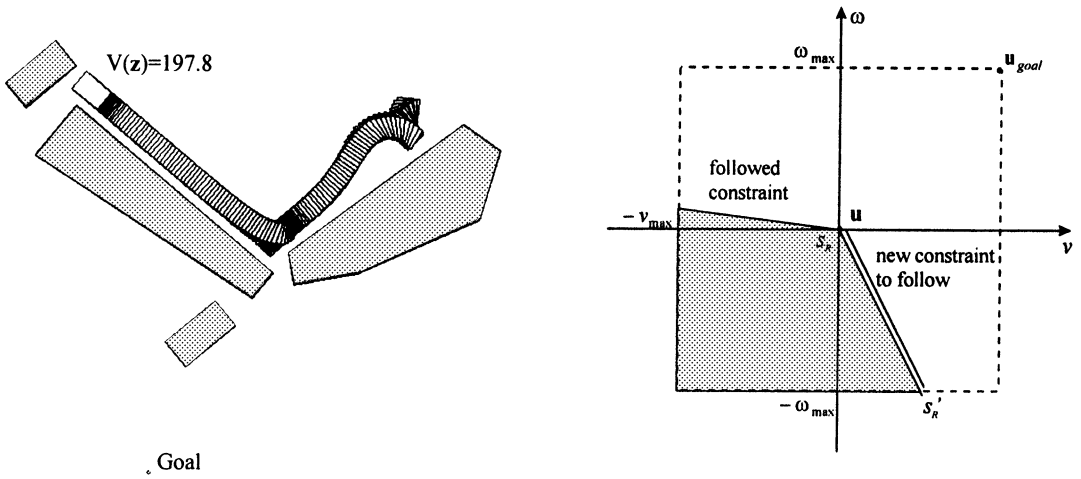


Fig. 12. Switching followed constraint.

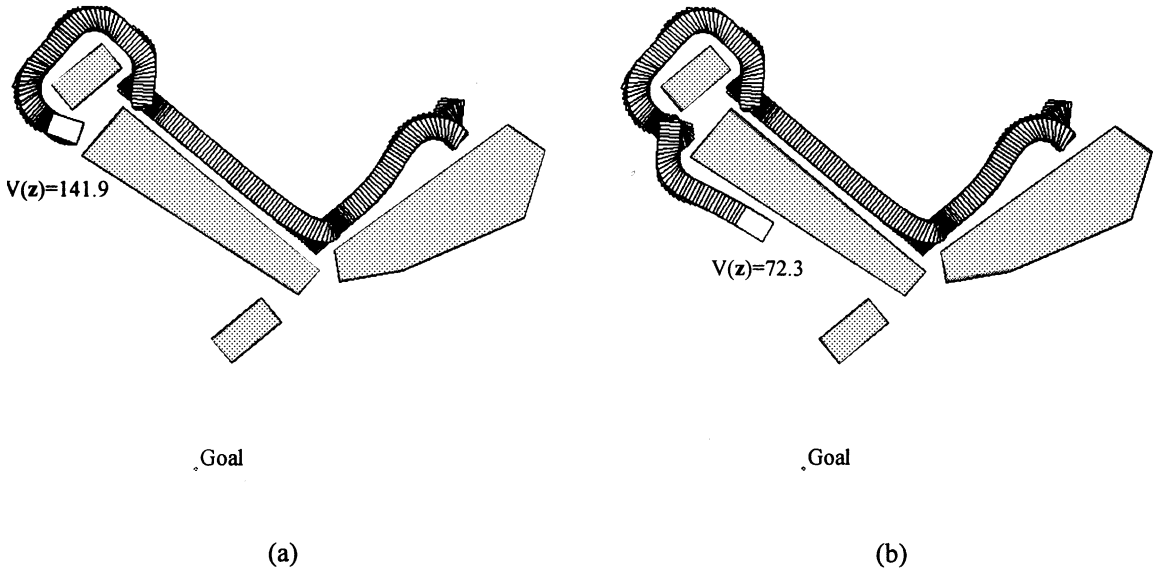


Fig. 13. Circumventing obstacles.

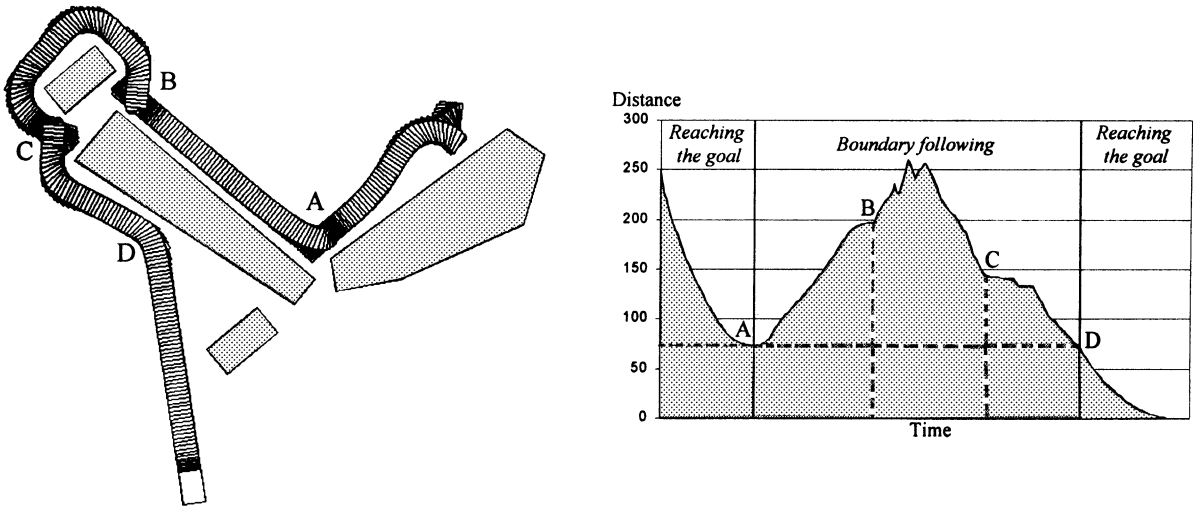


Fig. 14(a) Solution to path planning problem, (b) Distance function.

We can also observe that the transition point from a reaching the goal interval to a boundary following interval (deadlock situations, marked by '*' in Figure 15b) corresponds to a local minimum of the evolution of the function V(**z**). Because the vector $\mathbf{u}_{goal}$ is not considered in the *boundary following* process, the local minima of the evolution of V(**z**) in the *boundary following* intervals do not correspond to deadlock situations.

Since in the *reaching the goal* periods the function V(**z**) is strictly decreasing, and its initial and final values for the *boundary following* periods are equal, we can conclude that any new deadlock situation (represented by the transition *RG-BF*) is closer to the goal position than the preceding one.

If the *boundary following* intervals are bounded in time (i.e. there is no obstacle of infinite size and the goal position is reachable), then we can only consider the evolution of V(**z**) during the *reaching the goal* periods (darker graph in Figure 15b): V(**z**) is always decreasing whilst remaining greater than zero, therefore V(**z**) converges to zero or a positive value. If the goal position is unreachable, then, after the last deadlock, there is an obstacle (or a set of obstacles) that the mobile robot cannot circumvent. The value of V(**z**) at the last deadlock point is the global minimum of V(**z**) and the robot cannot move any closer to the goal position. The robot will continuously move around the blocking obstacles and the last period of the trajectory is an infinite *boundary following* period.

On the other hand, if the goal position is reachable, then all the obstacles can be circumvented and the last interval of the trajectory is a *reaching the goal* period. In this period the method uses the control law proposed in Section 3, and we have proved that, while using such control law, the system:

$$\dot{a} = -\nu \cos\alpha$$

$$\dot{\alpha} = \frac{1}{a}\nu \sin\alpha - \omega$$

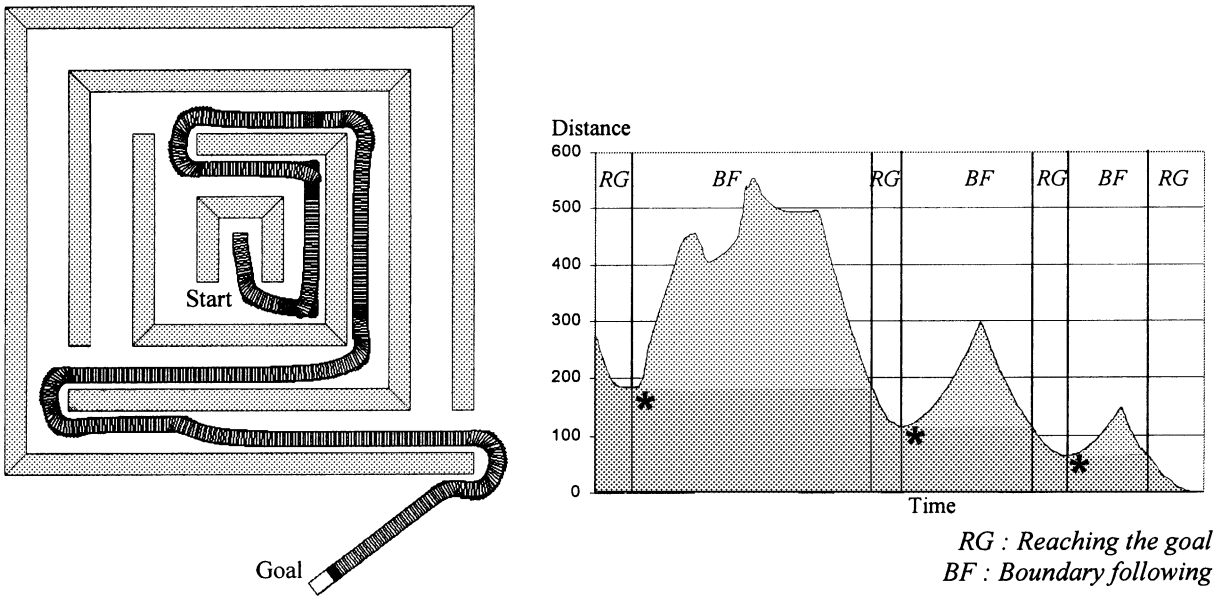
has only an equilibrium point located at the origin, and the origin is globally exponentially stable. We can conclude that *a* converges to zero in a finite time, and subsequently, the function V(**z**) converges to zero and the robot will reach the goal position.

Figure 16 shows another example, *getting into a labyrinth*, and the corresponding *distance* evolution.

It is important to notice that the two modules only use the *feasible velocities polygon*, built from the distance information between the robot and the obstacles; so, the proposed method is well adapted for implantation on a robot equipped with distance sensors, as ultrasonic sensors. The proposed path planner has been implanted on a real mobile robot (Robosoft Robuter) and the results are very satisfactory.

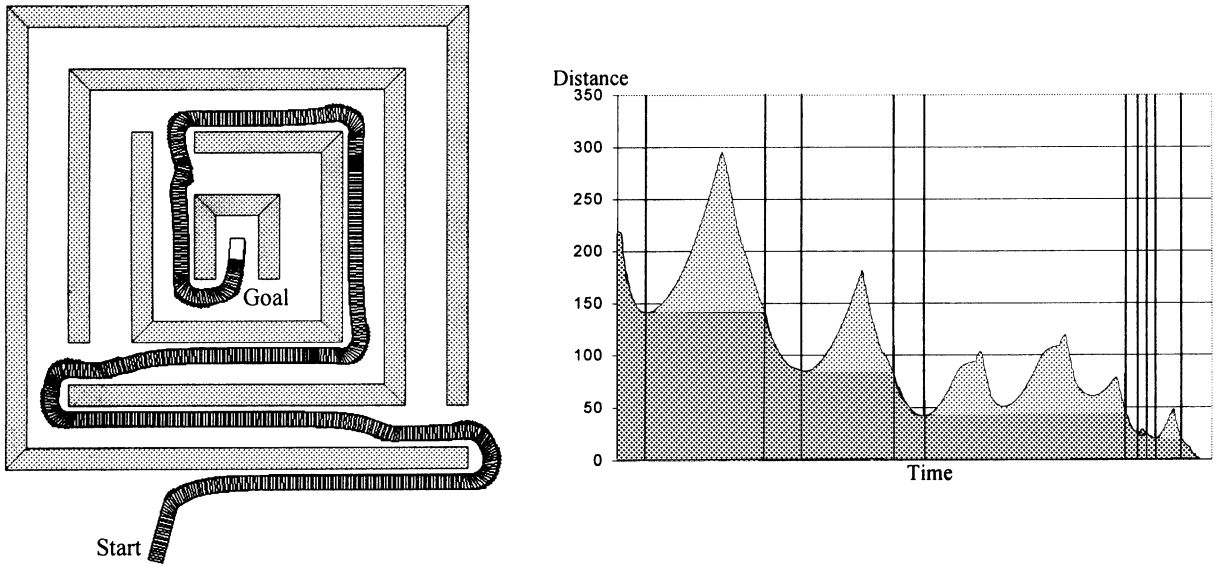
8. CONCLUSION

In this paper, a new collision-free path planner for nonholonomic mobile robots has been presented. Two modules compose this planner. The first module allows the robot to continuously approach the goal position while avoiding the obstacles, following a reference trajectory obtained from an exponential stable control law, which considers the nonholonomic property of the robot. In order to avoid the collisions, the objects in the influence zone are mapped as linear constraints over the robot's velocity space, defining the ***Feasible Velocities Polygon***. The collision-free trajectory is obtained by minimizing the deviation of the current robot's trajectory from the reference trajectory,



(a) (b) *Distance* evolution

Fig. 15. Getting out from a labyrinth.



(a) (b) *Distance* evolution

Fig. 16. Getting into a labyrinth.

under the obstacle constraints. This minimization problem is solved by a minimal distance calculation in the velocity space, between the FVP and the point describing the goal velocity. The solution is obtained using a fast algorithm for minimal distance calculation.

The second module is activated when the first module detects a deadlock situation. As well as the first module, the second module uses the FVP representation to ensure the collision-free navigation. To escape from the blocking condition, the algorithm follows the obstacle boundary while evaluating the distance to the goal.

The results show that the proposed path planner gives a solution in almost all cases where the classical local methods fail. The computational times are very short, which allows the implantation of the proposed method for real-time application. The path planner has been implanted on a commercial mobile robot, Robosoft Robuter, and experimental tests have been successfully performed.

References

1. T. Balch and R. Arkin, "Avoiding the past: a simple but effective strategy for reactive navigation", *IEEE International Conference on Robotics and Automation* (1993) pp. 678–685.
2. J. Barraquand et al., "Numerical potential field techniques for robot path planning", *Proceedings of IEEE International Conference on Advanced Robotics*, Pisa, Italy (1991) pp. 1012–1017.
3. F. Garcia and R. Mampey, "Mobile robot planning by reasoning both at itinerary and path levels", *Proceedings of IEEE International Conference on Advanced Robotics*, Pisa, Italy (1991), pp. 1074–1080.
4. Y. H. Liu and S. Arimoto, "Proposal of tangent graph and extended tangent graph for path planning of mobile robots", *Proceedings of IEEE Conference on Robotics and Automation* (1991) pp. 312–317.
5. S. S. Iyengar et al., "Robot navigation algorithms using learned spatial graphs", *Robotica* **4**, Part 2, 93–100 (1986).
6. D. W. Cho et al., "Experimental investigation of mapping and navigation based on certainty grids using sonar sensors", *Robotica* **11**, Part 1, 7–17 (1993).
7. T. Shibata and T. Fukuda, "Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system", *Proceedings of IEEE Conference on Robotics and Automation*, Atlanta, USA (1993) pp. 760–765.
8. W. Tianmiao and Z. Bo, "Time-varying potential field based "perception-action" behaviors of mobile robot", *Proceedings of IEEE International Conference on Robotics and Automation* (1992) pp. 2549–2554.
9. T. Skewis and V. Lumelsky, "Experiments with a mobile robot operating in a cluttered unknown environment", *Proceedings of IEEE International Conference on Robotics and Automation* (1992) pp. 1482–1487.
10. M. Wolfensberger and D. Wright, "Synthesis of reflexive algorithms with intelligence for effective robot path planning in unknown environments", *SPIE Mobile Robots* 70–81 (1993).
11. J. Borenstein and Y. Koren, "The vector field histogram – Fast obstacle avoidance for mobile robots", *IEEE Transactions on Robotics and Automation* **7**, 278–288 (1991).
12. Y. Maceda, "Collision avoidance control among moving obstacles for a mobile robot on the fuzzy reasoning", *Journal the Robotics Society of Japan* **6**, No. 6, 50–54 (1990).
13. B. Beaufrere and S. Zeghloul, "A mobile robot navigation method using fuzzy logic approach", *Robotica* **13**, Part 4, 437–448 (1995).
14. G. Ramirez and S. Zeghloul, "A new local path planner for nonholonomic mobile robot navigation in cluttered environments", *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, USA (2000) pp. 2058-2063.
15. V. J. Lumensky and A. A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment", *IEEE Transactions on Automatic Control* 1058–1063 (1986).
16. I. Kamon et al., "A new range-sensor based globally convergent navigation algorithm for mobile robots", *Proceedings of IEEE International Conference on Robotics and Automation* (1996) pp. 429–435.
17. R. W. Brockett, "Asymptotic stability and feedback stabilization", *Differential Geometric Control Theory* (Birkhauser, 1983) pp. 181–208.
18. B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with high number of degrees of freedom", *Proceedings of IEEE International Conference on Robotics and Automation* (1987), pp. 1152–1159.

19. O. J. Sordalen and C. Canudas, "Exponential control law for a mobile robot: extension to path following", *Proceedings of IEEE International Conference on Robotics and Automation* (1992) pp. 2158-2163.
20. S. Zeghloul and P. Rambeaud, "A fast algorithm for distance calculation between convex objects using the optimization approach", *Robotica* **14**, Part 3, 355–363 (1996).

APPENDIX

In order to demonstrate that the origin (which represents the goal) is globally exponentially stable, it is sufficient to find a function $V(\mathbf{z})$ that verifies the following theorem:

Lyapunov's theorem: Let $\mathbf{x}^* = \mathbf{0}$ be an equilibrium point of the system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$, and $V(\mathbf{x})$ an application $V(\mathbf{x}): D \rightarrow \Re$, $D \rightarrow \Re$, $D \subset \Re^n$ with the following properties:

(*i*) $V(\mathbf{x})$ is continuous and differentiable in D,
(*ii*) $V(\mathbf{0}) = 0$,
(*iii*) $\dot{V}(\mathbf{x}) > 0 \quad \forall \mathbf{x} \neq \mathbf{0}$

Under these conditions, we can conclude

(*a*) if $\dot{V}(\mathbf{x}) = \left(\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \right)^T f(\mathbf{x}, \mathbf{u}, t) \leq 0$; $\mathbf{x}^* = \mathbf{0}$ is globally stable
(*b*) if $\dot{V}(\mathbf{x}) < 0 \quad \forall \mathbf{x} \neq \mathbf{0}$; $\mathbf{x}^* = \mathbf{0}$ is globally exponentially stable.

The differential equation $\dot{\alpha} = -k_2 \alpha$ has the analytical solution $\alpha(t) = \alpha_0 e^{-t/\tau}$, with $\alpha_0 = \alpha(0)$ and $\tau = 1/k_2$. This function converges to zero and we can consider that $\alpha = 0$ at $t = 5\tau$. At $t = 5\tau$, the closed loop system becomes:

$$\dot{a} = -k_1 a$$

$$\dot{\alpha} = 0$$

The analytical solution for $a(t)$ is $a(t) = a_0 e^{-k_1(t-5\tau)}$, with $a_0 = a(5\tau)$ and $t > 5\tau$. In conclusion. both a and α converge exponentially to zero.