Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

A New Local Path Planner for Nonholonomic Mobile Robot Navigation in Cluttered Environments

Gabriel Ramírez and Saïd Zeghloul
Laboratoire de Mécanique des Solides - UMR 6610 CNRS, Université de Poitiers
SP2MI, Bd. Pierre et Marie Curie, Téléport 2, BP 179, 86360 Futuroscope CEDEX, France

Abstract

This paper presents a new local path planner for mobile robots in an environment with obstacles, where the nonholonomic constraints are considered. This planner uses only the distance information between the robot and the obstacles, thus it is well adapted for a robot equipped with embarked sensors, such as ultrasonic sensors. The obstacles are mapped as linear constraints into the velocity space of the robot ; since the obstacles constraints are linear, they form a convex subset which represents the velocities that the robot can use without collision with the objects. We call this convex subset the « feasible velocities polygon » (FVP). The planner is composed by two modules ; the first one is based on an optimization problem, which is transformed into a minimal distance calculation problem, in the velocity space, between the FVP and a point, which represents the reference velocity obtained by considering the nonholonomic constraint; the second module, which is used when a dead-lock situation occurs, uses the FVP representation to find, locally, the best velocity to escape from the dead-lock. 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 navigation problem of wheeled mobile robots in cluttered environments has been addressed from two points of view:

(a) In completely known environments, the works have been focused to global path planning, using different methods: grid of visits [1], numeric potential fields [2], graphs (of visibility [3], of tangents [4], Voronoï [5]) all using an A* algorithm, behavior-based models [6] and genetic algorithms [7], among others. The main disadvantage of all these methods is the very expensive calculation time, which makes them prohibitive for use in real-time applications.

(b) The local path planning is used for navigation in known as well as in unknown environments or in dynamic environments. The short calculation time allows the robot to react in real-time. Examples of this approach are the potential fields [8], the Bug algorithm [9], the behavior-based models [10], the probabilistic models (VFH [11], certainty grids [6]), the fuzzy logic [12][13], etc. For the most of these methods, the robot is considered as a point with holonomic properties, thus it can move in all directions without reorientation. Recently, the nonholonomic properties of mobile robots have been included into the path planning problem.

In this paper we present a local path planning method where the obstacles and the robot are modeled by convex polygons, and the robot is considered as a system with nonholonomic properties. This work has been focused to the differential mobile robots (robots equipped with two parallel driving wheels and no steering wheel) and constitutes an extension of the work presented in [14], in order to solve the dead-lock situations.

The proposed method uses two different modules. The first module is a control law and a velocity optimizer, in order to approach the robot to the goal following a stable trajectory while avoiding the obstacles. The objective of the control law is to calculate a reference velocity, which stabilizes the robot on a fixed position in a free environment, despite the nonholonomic property. The velocity optimizer is used to avoid obstacles, and uses a formulation similar to the one proposed by Faverjon et Tournassaud in [15]. The obstacles are mapped as a linear constraints over the robot's velocity to form the *feasible velocities polygon.* The module, via distance calculation, finds the closest velocity on the polygon to the reference velocity. Finally, the second module uses the *feasible velocity polygon* to escape from the dead-lock situations that could appear over the trajectory.

2. Kinematic model of a differential mobile robot

The configuration of a mobile robot in a plan can be completely defined by the following state vector

$$\mathbf{q} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$$

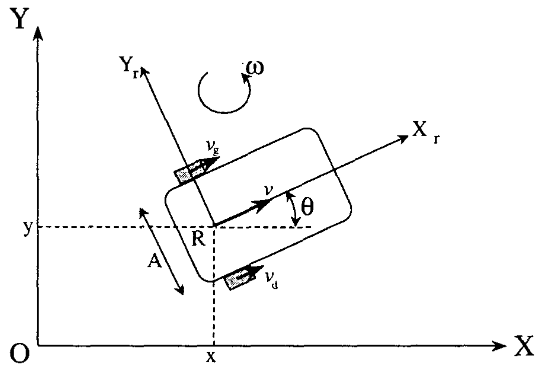


Figure 1. A differential mobile robot

where (x, y) is the position of a fixed point on the robot and θ the orientation of the frame linked to the object 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 v and the angular velocity ω of the robot are given by:

$$v = \frac{v_1 + v_2}{2} \quad , \quad \omega = \frac{v_1 - v_2}{A}$$

where v_1 and v_2 are respectively the right wheel and the left wheel velocities. The kinematic model of the mobile robot can be written as follows:

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

where $\mathbf{J}(\mathbf{q})$ is the Jacobean matrix of the robot, and $\mathbf{u}=[v,\omega]^T$ 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 the recent years. These systems cannot be stabilized to the equilibrium point using a time-invariant, smooth (or even continuous) feedback control law [16]. 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, in [17], have developed a similar control law in which the final orientation is considered.

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. Using the polar coordinates (a, α) of the goal with respect to the robot's frame, we define the error vector $\mathbf{z}(t)$ as follows (see Figure 2):

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

where:

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

$$\alpha = \text{atan2}(y_e, x_e) - \theta \quad \alpha \in [-\pi, \pi]$$

The dynamics of vector $\mathbf{z}$ is given by (cf. [14]):

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

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

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

$$v = k_1\, a\cos\alpha$$

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

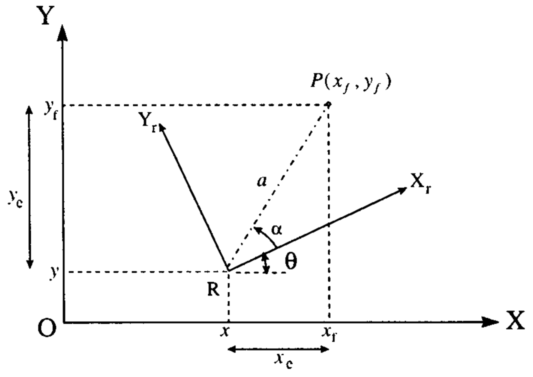


Figure 2. Error definition

with k1, k2 > 0, thus the closed loop system becomes:

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

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

In order to prove that the origin (which represents the goal) is globally exponentially stable, we can use the following Lyapunov function:

$$\mathbf{V}(\mathbf{z}) = \frac{1}{2}a^2 + \frac{1}{2}\alpha^2 \geq 0 \quad \forall \mathbf{z}$$

such that:

$$\dot{\mathbf{V}}(\mathbf{z}) = -k_1 a^2\cos^2\alpha - k_2\alpha^2 < 0 \quad \forall \mathbf{z} \neq 0$$

Because such Lyapunov function exists, the stable convergence to 0 of (a, α) is guaranteed, and a=0 implies that the robot has reached the final position.

4. Navigation in a free environment

Using the control law described above, we obtain the path shown in Figure 3. The initial position is the point $\mathbf{q}_i=[6, 3, \pi/4]^T$ and the goal position is located at the origin $\mathbf{q}_f=[0, 0, 0]^T$. For this example, the coefficients k_1, k_2 are set to 0.6.

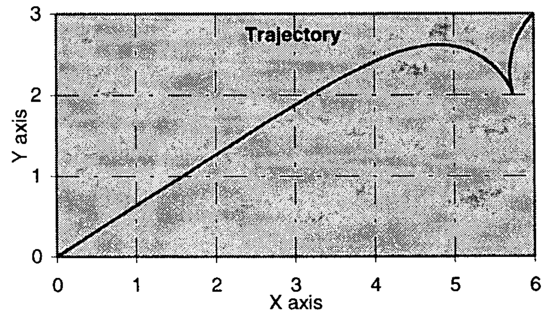
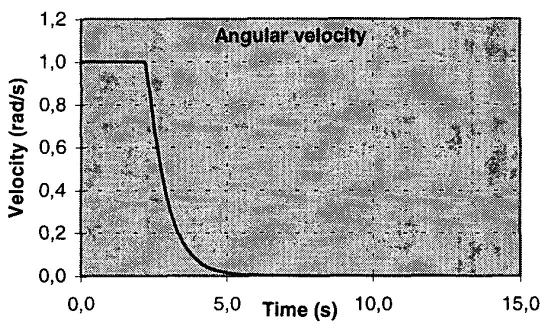
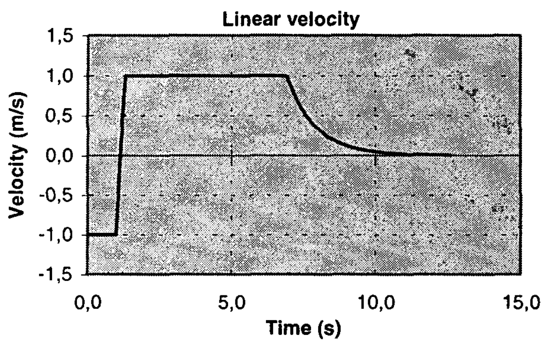


Figure 3. Robot's trajectory

The Figure 4 shows the robot's velocities evolution. The velocity limits are set to v_{max}=1.0 m/s and ω_{max}=1.0 rad/s. We can observe that the velocities are continuous everywhere, except in t=0.



(a) Angular velocity



(b) Linear velocity
Figure 4. Robot's velocities

5. « Reaching the goal » module

In [15], Faverjon and Tournassaud have proposed a substitute of potential field method to define the trajectory in cluttered environment, for a manipulator robot with a high number of degrees of freedom. This approach is called *constraints method*, because the obstacles are mapped as linear constraints over the robot's joint velocities. 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 $\dot{\mathbf{q}}$ is the joint velocities vector which minimizes the following function:

$$f = \frac{1}{2}\left\|\dot{\mathbf{q}} - \dot{\mathbf{q}}_{goal}\right\|^2 \quad \text{with} \quad \dot{\mathbf{q}}_{goal} = \frac{\mathbf{q} - \mathbf{q}_f}{\left\|\mathbf{q} - \mathbf{q}_f\right\|}$$

subject to 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, and ξ represents a coefficient in order to adapt the convergence speed.

The first module of our method uses a similar approach. The optimization problem can be stated as follows: find a vector $\mathbf{u}$ that minimizes the following function

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

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 proposed in section 3.

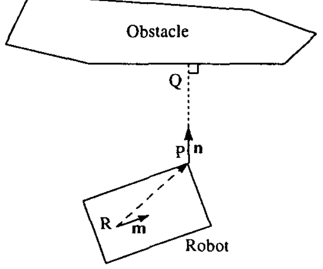


Figure 5. Establishing the obstacle constraints

For any obstacle at a distance d less than d_i, we can express the *velocity damper* constraint as a function of the control vector $\mathbf{u}$:

$$(v\mathbf{m} + \omega\hat{k} \times \overline{RP}) \cdot \mathbf{n} \leq \xi\frac{d - d_s}{d_i - d_s}$$

where $\mathbf{m}$ is the unit vector along the robot axis X_r and $\mathbf{n}$ is the unit vector along the segment PQ of minimal distance between the robot and the obstacle.

The set of obstacle constraints and the velocity limit constraints are linear in (v, ω), thus they define a convex subset in the (v, ω) space. Because this subset defines the bi-dimensional set of velocities 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 6(a). 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. The resulting FVP, in the (v, ω) space, is shown in Figure 6(b), where the velocity limit constraints and the two obstacle constraints define the convex subset which guarantees that the robot can move without collision. The velocity to reach the goal in a free environment is represented by the point $\mathbf{u}_{goal}$.

We can observe that the minimization problem

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

subject to the obstacle constraints

$$(v\mathbf{m} + \omega\hat{k} \times \overline{RP}) \cdot \mathbf{n} \leq \xi\frac{d - d_s}{d_i - d_s}$$

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 $\mathbf{u}_{goal}$. To solve this problem, we use a fast minimal distance calculation algorithm, developed by S. Zeghloul and P. Rambaud [18].

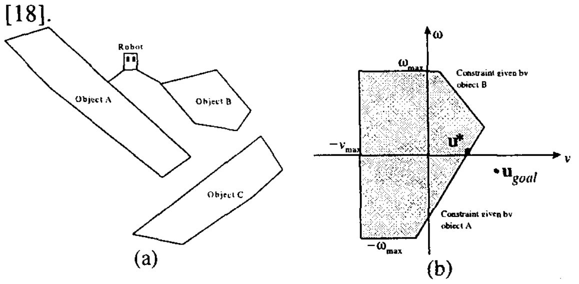


Figure 6. (a) Obstacle configuration, (b) Feasible velocities polygon

The variation of the configuration vector is then calculated using the solution vector, **u***, as follows:

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

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

(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 ;
(iii) solve the optimization problem by minimal distance calculation ;
(iv) apply of vector **u***.

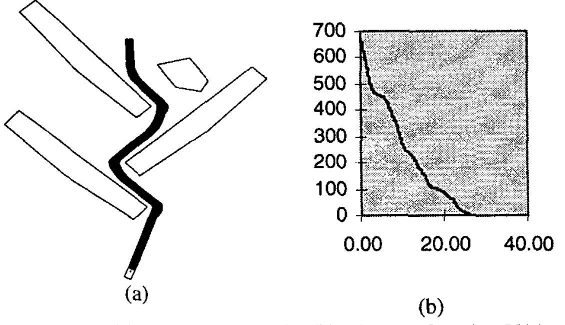


Figure 7. (a) A simple example, (b) Distance function V(**z**)

This module allows the robot to solve simple situations where the obstacles configuration does not create local minima points. An example is shown in Figure 7. While the robot uses the solution **u***, the "distance" function:

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

is strictly decreasing, since the robot is continuously approaching to the goal. This property is used in the second module to solve the dead-lock situations.

When the solution of the optimization problem, **u***, is located at the origin of the plan (v, ω), the robot cannot continue to move using the *reaching the goal* module. This condition corresponds to a dead-lock situation. In this circumstances, the second module, called *boundary following*, is applied.

6. « Boundary following » module

The "boundary-following" module and its interaction with the *reaching the goal* module have been inspired by the Bug algorithm [9]. The module uses the FVP representation and the value of the distance function to allow the robot to follow the obstacle boundary in order to escape from the dead-lock.

According to the location of **u*** on the FVP, there are two kinds of dead-lock situations: the *single-obstacle dead-lock*, which occurs when **u*** is located over one constraint of the FVP, and the *two-obstacles dead-lock*, when **u*** is located at a vertex of the FVP, as described in Figure 8(a) and Figure 8(b) respectively.

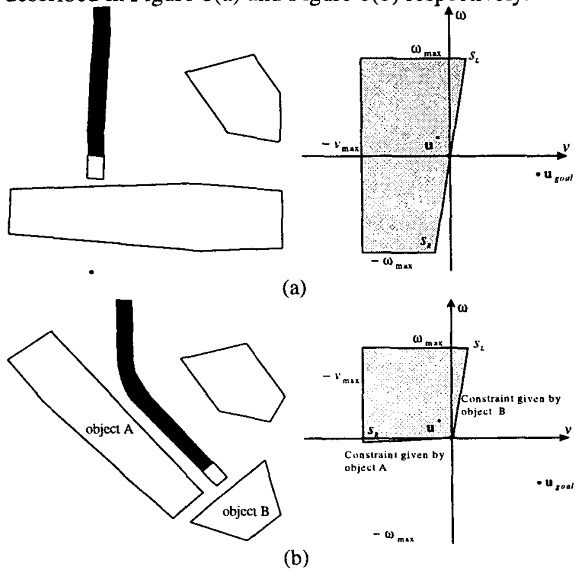


Figure 8. (a) Single-obstacle dead-lock, (b) Two-obstacles dead-lock

Regardless of the number of obstacles involved in blocking situation, the solving process is the same for both cases. When a dead-lock situation is detected (**u***=**0**), the current value of the *distance* function, V_{block}, is recorded and the two adjacent vertices to **u***, s_R and s_L, are identified. The position of the blocking obstacles, with respect to the robot's current position, defines which obstacle constraint will be chosen to surround them, and which of the vertices s_R and s_L will be used.

Thereafter, the method tracks the evolution of the chosen constraint on the FVP and applies the velocities represented by the chosen vertex. This allows the robot to follow the boundary of the blocking obstacle at a distance greater than or equal to d_s.

While the robot is surrounding the obstacles, the value of the distance function is continuously compared with V_{block}. Because the distance function is a measure of « distance » between the robot and the obstacle, when the value of the distance function is lower than V_{block}, the robot has found a point over the convex obstacle boundary that is closer to the goal than the dead-lock point. At this moment, the robot leaves the *boundary following* module in order to continue to move to the goal using *reaching the goal* module.

It is possible that, while the robot is following the obstacle boundary, the chosen vertex converges to the origin of the (v, ω) plan. This means that a new obstacle prevents the robot from following the obstacle boundary. When this occurs, the method simply switches from the followed constraint to the new one. Therefore, the robot will surround the new obstacle.

To illustrate the two modules and their interaction, let's consider the situation shown in Figure 9, with an initial distance value V(**z**)=299.08:

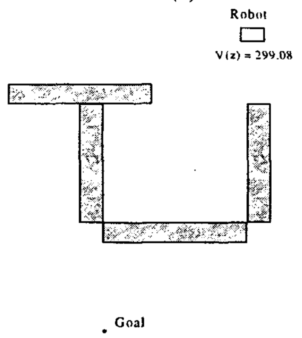


Figure 9. A "dead-end" example

As shown in Figure 10, the robot, using *reaching the goal* module, will move towards the final position following a stable trajectory, until this trajectory is blocked by an obstacle. This situation is detected as a dead-lock condition (**u***=**0**) and the current value of the distance function, V_{block}=52.43, is recorded:

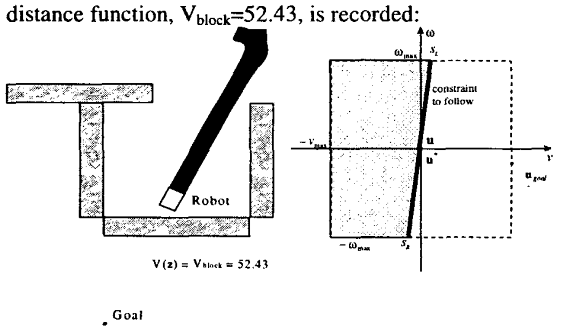


Figure 10. Dead-lock situation

Thereafter, the vertices s_R and s_L are identified. Because the blocking obstacle is located at the left of the robot, the chosen vertex is s_R since it allows the robot to turn to the right. The method will track the evolution of the followed constraint and will apply the velocities represented by s_R.

In the situation shown in Figure 11, the followed vertex s_R has converged to the origin of the (v, ω) plan, thus the robot has found a new blocking obstacle. The method switches the followed constraint, and continues to surround the new object, using the new vertex s_R'. Every time that the chosen vertex converges to zero, the method switches the followed constraint, so the robot can surround the blocking obstacles, as shown in the Figure 12.

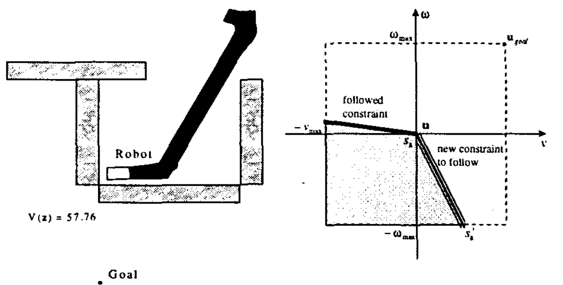


Figure 11. Switching followed constraint

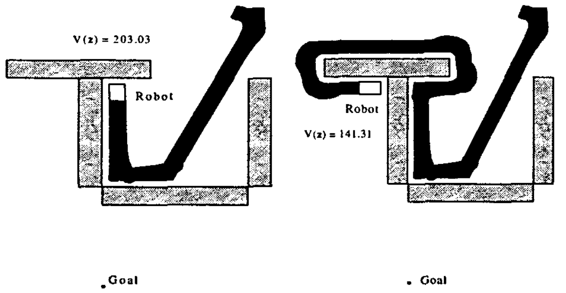


Figure 12. Surrounding obstacles

When the current value of the distance function is lower than the dead-lock value, V_{block}, the planner leaves the *boundary following* module and uses the *reaching the goal* module, until the goal is reached (cf. Figure 13).

The Figure 14 shows the evolution of the distance function for the whole trajectory. We can observe that, when the robot is using *reaching the goal* module, the distance function is a stable exponential curve as consequence of the exponential control law. We can also observe that the switching point from *reaching the goal* module to *boundary following* module is a local minimum in the evolution of the distance function. The other local minima do not represent dead-lock situations since in the *boundary following* module the vector $\mathbf{u}_{goal}$ is not considered to determinate the robot's velocity.

The resulting trajectory is a sequence of *reaching the goal* and *boundary following* intervals. By construction, for every *boundary following* interval the final value of the distance function is lower than the initial one (V_{block}). Furthermore, the distance function is strictly decreasing in the *reaching the goal* intervals. If the *boundary following* intervals are finite, we can affirm that the distance function converges to zero. Thus, the robot reaches the final position.

It is important to note that the two modules are based on the *feasible velocities polygon* only, which is built from the distance information between the robot and the obstacles, thus the method is well adapted for 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.

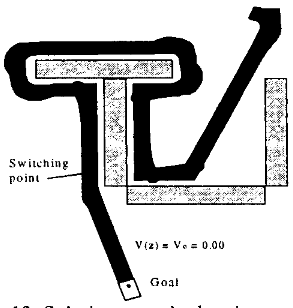


Figure 13. Solution to path planning problem

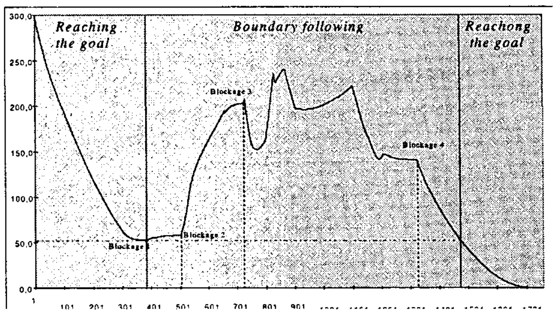


Figure 14. Distance function

7. Conclusion

In this paper, a new collision free path planner for nonholonomic mobile robots has been presented. This planner is composed by two modules. The first module allows the robot to continuously approach the goal position while avoiding the obstacles, following a stable reference trajectory obtained from an exponential control law which considers the nonholonomic property. In order to avoid the collisions, a similar approach to the one proposed in [15] is used: the objects in the influence zone are mapped as linear constraints over the robot's velocity space, forming the ***Feasible Velocities Polygon***. The collision free trajectory is obtained by minimizing the deviation of the current robot's trajectory from the reference trajectory, 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 second module is activated when a dead-lock situation is detected by the first module. The module uses the FVP representation to escape from the blocking condition, by following the obstacle boundary and 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 computing 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 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*, pp. 678-685, 1993.

[2] J. Barraquand et al., "Numerical potential field techniques for robot path planning," *Procs. Inter. Conf. Adv. Robotics*, pp. 1012-1017, Pisa, Italie, 1991.

[3] F. Garcia and R. Mampey, "Mobile robot planning by reasoning both at itinerary and path levels," *IEEE Int. Conf. Adv. Robotics*, pp. 1074-1080, Pisa, Italie, 1991.

[4] Y. H. Liu and S. Arimoto, "Proposal of tangent graph and extended tangent graph for path planning of mobile robots," *Procs. Robotics and Automation*, pp. 312-317, 1991.

[5] S. S. Iyengar et al., 'Robot navigation algorithms using learned spatial graphs," *Robotica*, pp. 93-100, 1986.

[6] D. W. Cho et al., "Experimental investigation of mapping and navigation based on certainty grids using sonar sensors," *Robotica*, pp. 7-17, 1993, vol. 11.

[7] T. Shibata and T. Fukuda, "Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system," pp. 760-765, *IEEE*, 1993.

[8] W. Tianmiao and Z. Bo, "Time-varing potential field based « perception-action » behaviors of mobile robot," *Procs. IEEE Int. Conf. Rob. Autom.*, pp. 2549-2554, 1992.

[9] T. Skewis and V. Lumelsky, "Experiments with a mobile robot operating in a cluttered unknown environment," *Procs. Int. Conf. Rob. Autom.*, pp. 1482-1487, 1992.

[10] M. Wolfensberger and D. Wright, "Synthesis of reflexive algorithms with intelligence for effective robot path planning in unknown environments," *SPIE Mobile Robots*, pp. 70-81, 1993.

[11] J. Borenstein and Y. Koren, "The vector field histogram - Fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, pp. 278-288, 1991, vol. 7.

[12] Y. Maeda, "Collision avoidance control among moving obstacles for a mobile robot on the fuzzy reasoning," 1990.

[13] B. Beaufrere and S. Zeghloul, "A mobile robot navigation method using fuzzy logic approach," Robotica, pp. 437-448, 1995.

[14] G. Ramirez and S. Zeghloul, "Path planning for a nonholonomic wheeled mobile robot in cluttered environments," *Proc. of the 4th Japan-France Congress on Mechatronics*, volume 1, pp. 337-342, 1998.

[15] B. Faverjon and P. Tournassoud, "A local based apporach for path planning of manipualtors with high number of dregrees of freedom," *IEEE Procs. Int. Conf. Robot. Autom.*, pp. 1152-1159, 1987.

[16] R. W. Brockett, "Asymptotic stability and feedback stabilization," *Differential Geometric Control Theory*, pp. 181-208, Birkhauser, 1983.

[17] O. J. Sordalen and C. Canudas, "Exponential control law for a mobile robot: extension to path following," *Procs. IEEE Int Conf. Rob. Autom.*, pp. 2158-2163, 1992.

[18] S. Zeghloul and P. Rambeaud, "A fast algorithm for distance calculation between convex objects using the optimization approach," *Robotica*, pp. 355-363, 1996.