Autonomous navigation of a nonholonomic mobile robot in a environment

Annemarie Kökösy, Franck-Olivier Defaux, Wilfrid Perruquetti

Autonomous navigation of a nonholonomic mobile robot in a complex environment

Annemarie Kokosy and Franck-Olivier Defaux
ISEN, LAGIS (CNRS UMR 8146)
41 boulevard Vauban
59046 Lille Cedex, FRANCE
annemarie.kokosy@isen.fr, defauxfo@hotmail.com

Wilfrid Perruquetti
EC-Lille, LAGIS (CNRS UMR 8146)
Cité Scientifique, BP48
59651 Villeneuve d'Ascq Cedex, FRANCE
wilfrid.perruquetti@ec-lille.fr

***Abstract*—This paper presents a new path planning algorithm for the autonomous navigation of a nonholonomic mobile robot. The environment in which the robot evolves is unknown and encumbered by obstacles. The goal of the robot is to move towards the arrival point (which is known) by avoiding the obstacles. The path planning algorithm recomputes a new trajectory whenever a new obstacle is detected. The planned trajectory takes account of the physical constraints of the robot (speed saturation, kinematic robot model, nonholonomic constraint). The trajectory of the robot is obtained by optimizing a problem of optimal control under constraints. The resolution of this problem is done by using the flatness property of the system, which transforms the initial optimization problem into a nonlinear dynamic programming problem. The problems of the local minima are solved by using a supervisor. Our algorithm will be compared with another algorithm of the literature in order to highlight its effectiveness. Simulation results will be presented to illustrate the good performance of the algorithm for robot navigation in a complex environment.**

Keywords: *robot navigation, non linear optimization, flatness, supervisor, nonholonomic robot, path planning, complex environment*

I. Introduction

The autonomous navigation of mobile robots in an environment with obstacles is a problem on which many works have currently been carried out and this for more than twenty years. When the robot does not know in advance the obstacles in the environment, it must use its sensors to be located and perceive the environment and to react accordingly. Two approaches can be found in the literature. In the first one, the robot uses the local sensory information in a sheer reactive way. Algorithms Bug1 and Bug2, presented in [1] only use position and contact sensors. These algorithms consist of two reactive modes of motion: moving directly towards the target and following an obstacle boundary, and a transition condition to switch between them. When the robot hits an obstacle, it switches from the mode moving towards a target to the mode following a boundary. It moves away from the obstacle boundary when a leaving condition, which ensures that the distance to the target decreases, holds. These algorithms guarantee the global convergence to the target, but their performances depend on the complexity of the map. In particular, the introduction in the environment of nonconvex obstacles can sometimes be catastrophic for the algorithms results. It also should be noted that the direction followed to make the turn around the obstacle boundary is arbitrary and it is impossible to know if it is the best choice. However, the original Bug algorithms do not make the best use of the available sensory data to produce short path. In the VisBug algorithm, presented in [2], the robot has a distance sensor enabling it to know which distance separates it from an obstacle, in all directions. The robot uses this distance information to find many shortcuts and to test a leaving condition compared to the Bug2 algorithm. One of the limits of the VisBug algorithm is that it makes only use of the data of the distance sensors with an aim at reducing the path between the initial and final points found by the Bug2 algorithm. The TangentBug algorithm, presented in [3], will more specifically exploit these data by building the local tangent graph. The convergence of this algorithm is guaranteed. Various properties of the TangentBug algorithm are shown. In particular, simple conditions to detect that the target is unattainable are given, as well as a higher limit on the total way crossed according to the perimeter of the obstacles and the number of local minima met. Various examples and simulation results highlight the interest of this algorithm compared to VisBug. The authors of the algorithm CautiousBug [4] highlight an important limitation of the various Bug methods: at the time of the meeting of an obstacle, the choice of the boundary-following direction is based on local information. This choice may be wrong in the sense that it will result in a longer path. They propose a simple method which will allow, in many cases, to guarantee better performances, as well as a factor of competitiveness to compare the results of the various algorithms (it consists of the relationship between the length of the way crossed by the studied algorithm and the length of the optimal way, obtained when the map is entirely known). The algorithm is based, following the example of TangentBug algorithm, on the local graph of the tangents. The difference will be made on the choice of the direction followed when a local minimum is detected. Instead of basing itself on local information, the robot will carry out a research in spiral. Nevertheless, this research in spiral requires to make manoeuvrers with the robot, which in the case of a point robot do not have any importance,

but which, in the case of a nonholonomic robot, can reduce the optimality of run time (difficulty in making half-turn, even impossibility in a reduced space). In conclusion, the Bug algorithms give excellent results for the completely unknown environment, because they always make it possible to converge towards the target or to find that the target is unreachable. However, the path carried out can be in many cases far from optimality. Moreover, these algorithms would require a strong theoretical development taking account of the uncertainties of the sensors and kinematics and dynamic constraints of the robots.

In the second approach, the robot will plan its trajectory in the environment. It is the case of work exploiting the deformable virtual zones (DVZ) [5], [6] which as are developed in [7]. In the DVZ algorithm, the robot to be moved will be surrounded by a virtual envelope, typically in an elliptical shape. This envelope could be deformed in two ways: either in an internal way (the variables of the robot model employed act on its form), or in an external way (when an obstacle enters the virtual zone). The goal of the robot control using the DVZ method will then be to minimize the deformation due to the obstacles. The robot model which is used for the path planning optimization in [6] is a car like. The advantages of this method are that it does not require important computing time and can be implemented in real-time. It also allows the use of a kinematic model of robot, which facilitates the application to a real system. However, it requires many parameters of adjustments and does not guarantee the convergence of the robot towards the objective. The path planning algorithm presented in [7] guarantees the autonomous navigation of a nonholonomic robot in an unknown environment with obstacles of circular shape or which can be included in circles. The algorithm takes account of the kinematics constraints of the robot as well as of its physical limitations (maximal speed). The trajectory is calculated by using the property of flatness of the system and can be implemented in real-time, because the optimal problem is calculated on a reducing horizon. However, it does not allow to take into account complex shapes of obstacles, since only the circular obstacles can be taken into account.

The outline of this paper is as follows. The problem statement is given in Section II. Section III gives the main results. Simulation results are detailed in Section IV.

II. Problem statement

The robot is represented by a disc located at the mass center (x, y) and of radius r (Fig. 1). $q = [x, y, \theta]^T$ and $U = [v, w]^T$ respectively denote the state variables and the control inputs (linear and angular velocities). The kinematic equations of the system under the nonholonomic constraint of pure rolling and no slipping can be written as follows:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= w \end{aligned} \tag{1}$$

The robot must evolve in an autonomous way of an initial given point to a final known point while avoiding collision with the unknown obstacles met on its way. It is therefore necessary to construct a robot control able of guaranteeing its navigation without collisions while taking into account the robot physical limitations.

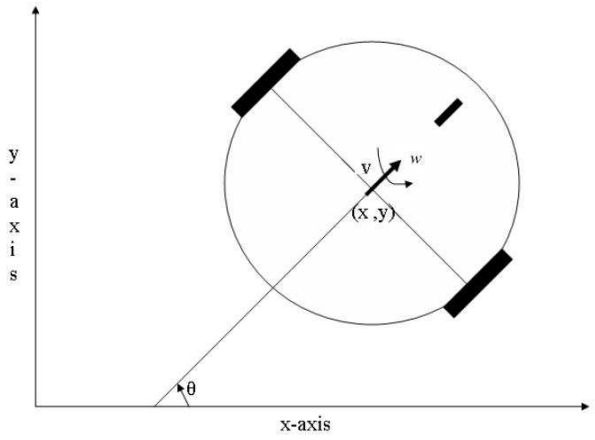

Fig. 1. The car like robot

III. Main results

A. Motion planning [7]

The goal is to generate a feasible quasi-minimum time trajectory that satisfies the environmental constraints: obstacle avoidance and also the physical constraints due to the limitations on the velocities. The optimal control problem is to find the control inputs which minimize:

$$J = \int_{\tau_i}^{t_f} dt, \tag{2}$$

where the initial time is $\tau_0 = 0$, $\tau_i = t_i + \delta t$ and t_f is the unknown final time. The trajectory must join the known states $q(\tau_i)$, $q(t_f)$ and satisfy the constraints, $\forall t \in [\tau_i, t_f]$:

$C1$ the optimal trajectory and the optimal control are solutions of the cinematic model of the robot (1).

$C2$ the control bounds:

$$|v| \leq v_{max} - \epsilon_v, \quad |w| \leq w_{max} - \epsilon_w,$$

where ϵ_v and ϵ_w are positive control parameters. The inclusion of these constants in the constraints of the motion planning generator guarantees that there is sufficient control authority to track the trajectory.

$C3$ the collision avoidance with the N_o detected obstacles which can be included in a circle:

$$\forall m \in \{1, \ldots, N_o\}, \quad \sqrt{(x - x_m^o)^2 + (y - y_m^o)^2} \geq r + r_m^o$$

with $r_m^o > 0$.

B. Taking into account of obstacles with general shape

In order to take into account obstacles of more general forms in the problem of path planning, a modeling of the obstacles in the form of polygons was used. This modeling has the advantage, compared with modeling by circles [7],

of being able to be used for all shapes of obstacles which can be included in a polygone, by respecting its contour as well as possible, which limits the unused space for the robot displacement. One can notice that the more the number of segments used to model the obstacle will be important, the more this obstacle will be represented accurately. An obstacle O_i defines an inaccessible zone for the robot in its environment. The complex contour of the obstacle can be approximated by a succession of N_i segments S_{ij}. Each one of these segments is represented by its two ends points A_{ij} and B_{ij}. These points have respectively $(x_{A_{ij}}, y_{A_{ij}})$ and $(x_{B_{ij}}, y_{B_{ij}})$ as coordinates.

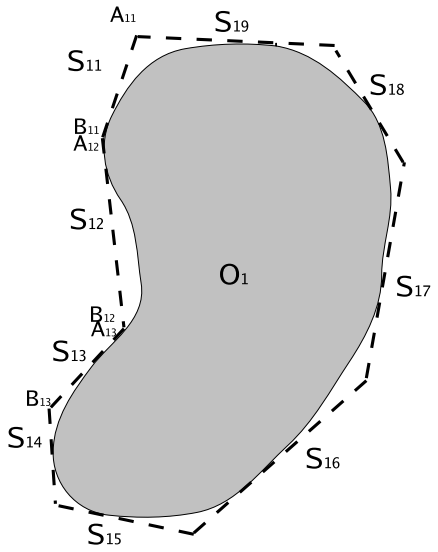

Fig. 2. Approximation of obstacles with complex shape

It is possible to define obstacle O_i like the meeting between its contour o_i and its interior O_i^0:

$$O_i = o_i \cup O_i^0,\ o_i = \cup_{j=1}^{j=N_i} S_{ij}, \tag{3}$$

We note N_p the number of obstacles other than circular ones in the sensor range of the robot. The whole of the N_p detected obstacles of the robot environment is the union of the O_i.

The obstacles are supposed to have a closed contour:

$$\forall S_{ij} \in O_i, \exists S_{ik}, k \in 1, ..., N_i, k \neq j$$

$$such\ that\ A_{ik} \equiv A_{ij}\ or\ B_{ik} \equiv A_{ij}$$

$$\exists S_{il}, l \in 1, ..., N_i, l \neq j\ such\ that\ A_{il} \equiv B_{ij}\ or\ B_{il} \equiv B_{ij}.$$

In practice, this modeling in the form of a polygon will result from an perception algorithm of the robot (using a video camera, or a lidar or a rangefinder). The path planning algorithm must take into account the uncertainties of the perception algorithm. So, we replace the previous relations by a new one defined by equation (4):

$$A_{ik} \equiv A_{ij} \Leftrightarrow d(A_{ik}, A_{ij}) < \epsilon \tag{4}$$

where $d(A_{ik}, A_{ij})$ is the euclidian distance between A_{ik} and A_{ij} :

$$d(A_{ik}, A_{ij}) = \sqrt{(x_{A_{ik}} - x_{A_{ij}})^2 + (y_{A_{ik}} - y_{A_{ij}})^2} \tag{5}$$

The robot position is defined by the coordinates of the point $R(x, y)$. The distance between the robot and the segment S_{ij} is defined by $d(R, S_{ij})$, (6):

$$d(R, S_{ij}) = \min_{M \in S_{ij}} d(R, M) \tag{6}$$

Let segment S_{ij} be defined by its end points A_{ij} et B_{ij}. Distance $d(R, S_{ij})$, (6) is calculated according to the robot position compared to segment S_{ij}. Three cases are to be distinguished (i.e. Figure 3) :

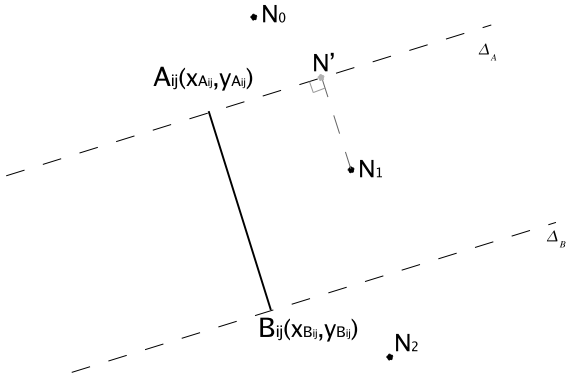

Fig. 3. The three cases for distance calculation

If $\overrightarrow{A_{ij}R} \cdot \overrightarrow{A_{ij}B_{ij}} \leq 0$, $\alpha > \frac{\pi}{2}$, then R and B_{ij} are on both sides of Δ_A, the line perpendicular to S_{ij} which passes by A_{ij} (case $R = N_0$ on Figure 3), so $d(R, S_{ij}) = \| A_{ij} - R \|$.

If not, if $\overrightarrow{B_{ij}R} \cdot \overrightarrow{B_{ij}A_{ij}} \leq 0$, $\beta > \frac{\pi}{2}$, then R and A_{ij} are on both sides of Δ_B, the line perpendicular to $(A_{ij}B_{ij})$ which passes by B_{ij} (case $R = N_2$ on Figure 3), so $d(R, S_{ij}) = \| B_{ij} - R \|$.

If not $0 < \alpha, \beta < \frac{\pi}{2}$, (case $R = N_1$ on Figure 3) $d(R, S_{ij}) = \| A_{ij} - N' \|$.
So, the definition of the distance between robot R and segment S_{ij} is given by equation (7):

$$d(R, S_{ij}) = \begin{cases} d(R, S_{ij}) = \| A_{ij} - R \|,\ if\ \alpha > \frac{\pi}{2} \\ d(R, S_{ij}) = \| B_{ij} - R \|,\ if\ \beta > \frac{\pi}{2} \\ d(R, S_{ij}) = \| A_{ij} - R \| \ sin(\alpha) = \\ \| B_{ij} - R \| \ sin(\beta),\ 0 < \alpha,\ \beta < \frac{\pi}{2} \end{cases} \tag{7}$$

As this distance belongs to the constraints to be introduced into the path planification algorithm, which calculates the optimal trajectory of the robot, it is necessary that it is of class C^1. It is thus necessary to check its continuity and its derivability when $\alpha = \frac{\pi}{2}$ or $\beta = \frac{\pi}{2}$. It is easy to prove that the distance defined by equation (7) belongs to class C^1. The proof is omitted due to the paper limitation.

Let $d(R, O_i)$ the distance between the robot and the visible part by the robot sensors of obstacle O_i. This distance is defined by equation (8):

$$d(R, O_i) = \min_j d(R, \mathbf{S}_{ij}),\ j \in 1, ..., N_i. \quad (8)$$

We can now add to the optimal control problem a new constraint which allows the avoidance of collision with polygonal obstacles :

$C4$ The distance between the robot and the N_p polygonal obstacles must obey to the following condition:

$$d(R, O_i) > \zeta > 0,\ \forall i \in 1, ..., N_p, \quad (9)$$

where the distance $d(R, O_i)$ is calculated by using equations (5), (6), (7) and (8).

Now, the path planning problem is completely defined: find an optimal control and an optimal trajectory which minimise the optimal cost (2), constraits C1 to C4 and the terminal constraint $q(t_0) = q_0$, the initial point. This problem is solved by using the flatness property of the system, the B-spline parametrization and constrained feasible sequential quadratic programming (for details see [7]).

C. Supervisor

The algorithm of trajectory planning enables us to find a trajectory for the robot between a starting point D and the point of arrival G in the space of the configuration which guarantees:

- the acceptable controls for the robots (which take into account the nonholonomic constraint and the maximum speed)
- the avoidance of circular and polygonal obstacles.

The representation of more complex obstacles by the segments involves the appearance of local minima for the function $d(R, G)$, (5). This function defines the distance between the current location of the robot and the objective to be reached. The problem of these local minima cannot be solved by the path planning algorithm. The idea that we have developed is to introduce a supervisor which according to the situation created by the obstacles, will give the robot some intermediate objectives. This supervisor must enable the robot to achieve the final goal by avoiding situations of local minima. The reach of the intermediate objectives is ensured by the path planning algorithm, which guarantees the respect of constraints. The supervisor acts according to the data received by the robot (its position and visible obstacles). It will then choose an intermediate objective for the robot. The trajectory between the current robot position and the intermediate objective is calculated by the path planning algorithm. Thus this trajectory will respect the various constraints imposed on the robot. The operation of the supervisor is inspired by the TangentBug algorithm. Just like this algorithm, the supervisor will switch between two operating processes: right mode towards the target and following an obstacle boundary mode. With each one of its interventions, the supervisor initially tests in which mode it must act, then it determines its intermediary objective following this mode.

1) Right towards the target mode: At each instant, the robot sees n_s segments $\mathbf{S}_j$, which define the graph $\mathbf{G}_0$, (10):

$$\mathbf{G}_0 = \{A_j, B_j\ \forall j \in \{1, ..., n_s\}\} \cup T \text{ if define} \quad (10)$$

This graph contains:

- ends points of segments which define the obstacles in the range of the robot sensors
- the intersection point T between the line (RG) and the range of robot sensors when no visible obstacle cuts $[RG]$.

The segments can no longer be identified as belonging to a particular obstacle since the knowledge of the environment is reduced to a local aspect. The algorithm defines the subgraph $\mathbf{G}_1$ in the following way:

$$\mathbf{G}_1 = \{V \in \mathbf{G}_0 \mid d(V, G) \leq \min\left(d(R, G), d_{Leave}\right)\} \quad (11)$$

In fonction of the content of subgraph $\mathbf{G}_1$, (11), two cases can appear:

1. $\mathbf{G}_1$ **is empty** The robot is then in the zone of attraction of a local minimum created by a single segment (i.e. Figure 4). The supervisor passes then in mode "following an obstacle boundary" after having saved the value of the local minimum:

$$d_{min} = d(I_{min}, G) \quad (12)$$

with I_{min} the intersection point between $[V_1\, V_2]$ and the line passing by point G and perpendicular with $(V_1\, V_2)$.

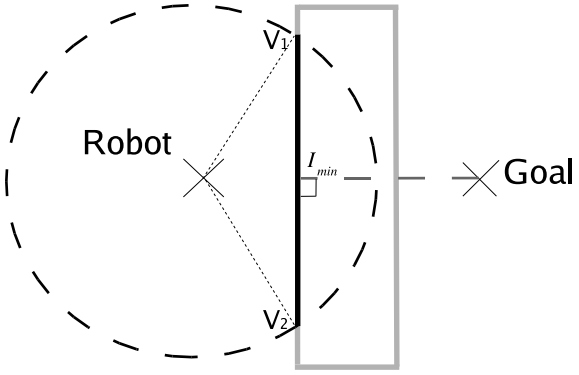

Fig. 4. Local minimum created by a single segment

2. $\mathcal{G}_1$ **is non empty**: There are four cases.
   a. If there is just one point in graph $\mathbf{G}_1$, the robot follows it, there is no local minimum.

   b. The robot is in the zone of attraction of a local minimum created by two segments (i.e. Figure 6). This situation appears if the vertex $V_p \in \mathbf{G}_1$ nearest to the objective belongs to two distinct segments $\mathbf{S}_1$ and $\mathbf{S}_2$. The supervisor passes then in mode "following an obstacle boundary" after having saved the value of the local minimum d_{min}:

$$d_{min} = \min\left(d(V_p, G), d(G, \mathbf{S}_1), d(G, \mathbf{S}_2)\right) \quad (13)$$

c. If there is a segment blocking the vision of the objective (Figure 5), and if this one creates a local minimum, then the robot passes in mode "following an obstacle boundary" after having saved the value of the local minimum in d_{min}.
d. Lastly, if none of the preceding cases is presented, the robot moves towards the locally optimal direction, defined by V_p, the nearest point to the objective belonging to $\mathbf{G}_1$.

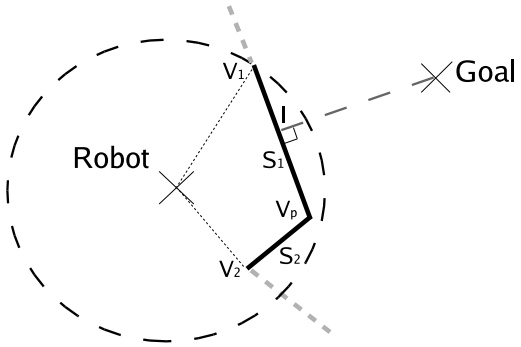

Fig. 5. The local minimum created by 2 segments, local minimum on the segment

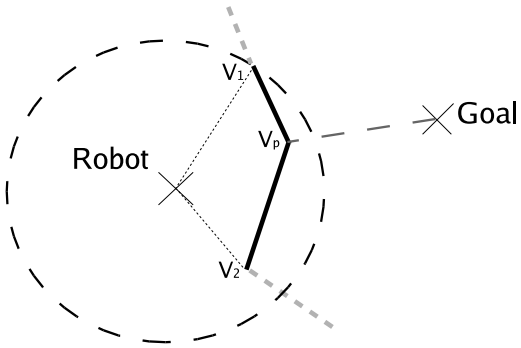

Fig. 6. The local minimum created by 2 segments, local minimum on the vertex

2) Following an obstacle boundary mode: When the robot is not in the right towards the target mode, it is in the following obstacle boundary mode. This behavioral change happens when the robot meets a local minimum, of which it saves the value d_{min}. In this mode, the robot will move along the contour of the obstacle while following an arbitrary direction (either definite according to local information, or fixed at the beginning). It still makes use of the local tangent graph to find shortcuts. It will switch in right towards the target mode once it finds a vertex V in its local graph of the tangents such that

$$d(V, G) < d_{min} \quad (14)$$

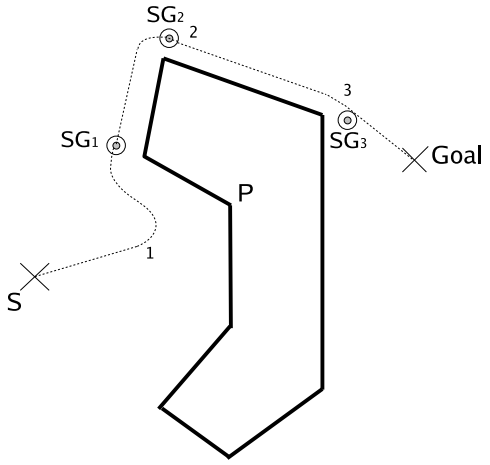

Fig. 7. Example of supervisor operating

On the example given in Figure 7, the robot leaves the point D, in right mode towards the target. In a, it meets the local minimum P and passes in skirting mode of the obstacle while running to the left. It then follows the under-objective S_{G1}. It continues skirting while passing by S_{G2}. In b, the robot finds a point closer to the objective than the local minimum P in its graph. Then it switches to right towards the target mode while following the under-objective S_{G3}. Once in c, it sees G, the objective and joins it.

The operating of the following an obstacle boundary mode is shown in the following organization chart (Figure 8):

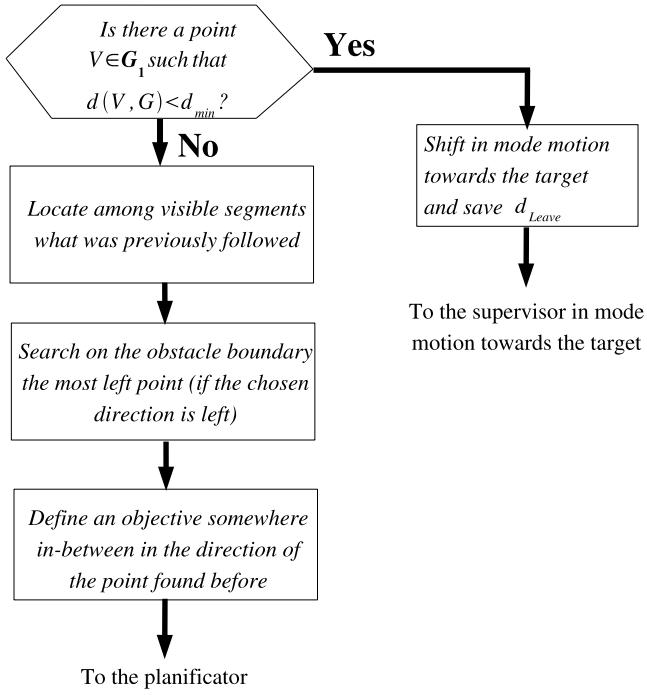

Fig. 8. Organization chart of the following an obstacle boundary mode

IV. Simulation Results

A. Comparison between the DVZ algorithm and the new path planning algorithm

The algorithm of planning trajectory based on the deformable virtual zones (DVZ) was tested in order to be able to compare it with the new proposed path planning algorithm using a supervisor. This one makes it possible to obtain good performances for simple scenes, containing convex obstacles. The following example shows a use of this algorithm for a more complex scene containing a concave obstacle, forming a broad zone of minimum local. One can see on Figure 9 trajectories generated by the DVZ algorithm and by the algorithm of the supervisor to go from the starting point $(3, 4)$ to the arrival point $(19, 19)$. The dotted line shows the trajectory generated by the DVZ method and the continuous line shows the trajectory generated by the new supervisor algorithm.

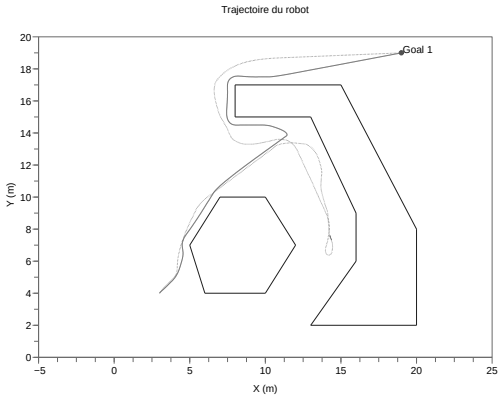

Fig. 9. Comparison between the DVZ algorithm and the supervisor algorithm : first case

The trajectory generated by the DVZ algorithm is far from optimality. However, the target is reached after a broad turning run time of 199 seconds. The algorithm of the supervisor will escape from the local minimum as soon as it has detected it, thus reducing waste of time in the zone of attraction. Run time is 107 seconds.

In the simulation presented on Figure 10, the conditions are identical, but, the objective to reach was moved into $(20, 15)$.

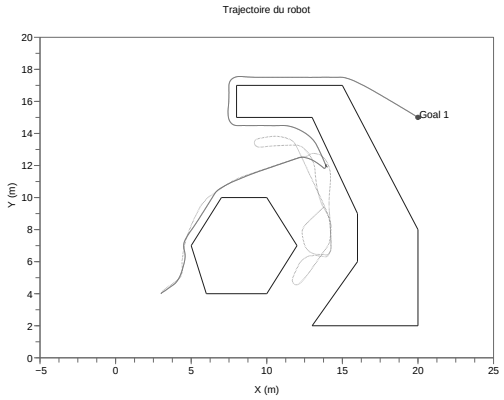

Fig. 10. Comparison between the DVZ algorithm and the supervisor algorithm : second case

In this example, we can see that the DVZ algorithm does not manage to achieve the goal, and that the trajectory remains blocked in the zone of attraction of the local minimum (dashed line in Figure 10). The algorithm of the supervisor manages to escape from this minimum, after having detected it. The difference between these two algorithms is that the supervisor takes into account in its reasoning the knowledge of the obstacles structure in a general way, by knowing that if there is a blocking on the contour of the obstacle, it is possible to find a solution by moving around the obstacle. The DVZ algorithm is obstinate to approach the target, without taking into account this more abstract information concerning the case studied. There may be a configuration of the parameters of the DVZ bringing the robot to the objective, but for this same configuration, there would be a scene for which the robot would be blocked.

B. Simulation results for the new supervisor algorithm in complex map

We will test the effectiveness of our new path planning algorithm in a complex scene where the robot will often be in zones of attraction of the local minima. For simulations we consider that the range of robot sensors is 2 meters. The maximum speed value is 0.3m/s. All the programs are written in C. The first map configuration and robot displacement are shown in Figure 11. The robot puts 73.5 second to move between its initial position and its final position on the unknown map. the robot is practically always at maximum Figure 12 illustrates the robot movement in another complex unknown map. It must go from the initial point $(4, 1)$ to the final point $(13, 14)$. It reaches that point in 95,9 seconds. During the skirting of the second obstacle, it seems obvious to us that the choice of skirting is bad. However, provided only with local information, the robot can only have one locally optimal decision.

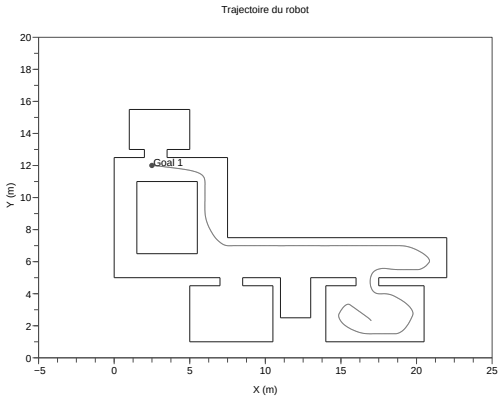

Fig. 11. Simulation result by using the supervisor path planning algorithm : first map configuration

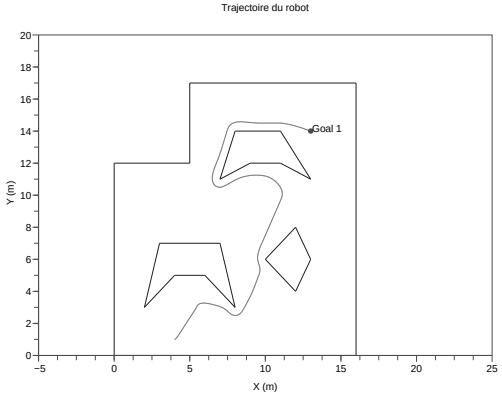

Fig. 12. Simulation result by using the supervisor path planning algorithm : second map configuration

V. Conclusion

This paper proposed a new path planning algorithm for a nonholonomic mobile robot moving in an unknown environment which contains obstacles. This algorithm is a generalization of the algorithm proposed in [7]. The new algorithm takes into account obstacles which can be included in a polygon. For example it allows navigation in a labyrinth, which was impossible to realize by using the algorithm presented in [7]. In order to solve problems of local minima, an architecture on two levels was developed. Indeed, a supervisor detects the problems of local minima and according to the nature of the problem met, it either goes right towards the target, or gets around the obstacle while following its contour. Effectiveness of the new algorithm was tested thereafter in simulation.

We compared it with algorithm DVZ because this algorithm is the only one which remains competitive compared to our starting criteria (take into account of the model of the robot and its physical constraints, avoidance of obstacles with unspecified shape and possibility of the implementation of the algorithm in real-time). These two algorithms were put in competition in the same environment. For a simple scene, method DVZ remains competitive, but in a little more complex environment, it does not manage any more to find a way to take the robot to its final point. Then, the algorithm of the supervisor was tested in more complex scenes in order to highlight its effectiveness. The algorithm proposed can be implemented in real-time, because the calculation of the optimal trajectory is done by using a receding horizon. It will be tested thereafter on a unicycle type robot.

Acknowledgment

This paper was partially supported by the ”Ensemble Innovons” Programme of Catholic University of Lille.

References

[1] Lumelsky, V., and Stepanov, A., ”Path-planning strategies for a point automaton moving amidst unknown obstacles of arbitrary shape”, *Algorithmica*, vol. 2, pp. 403-430, 1987.

[2] Lumelsky, V., and Skewis, T., ”Incorporating range sensing in the robot navigation function”, *IEEE Transactions on System, Man and Cybernetics*, vol. 20, no. 5, pp. 1058-1069, 1990.

[3] Kamon, I., Rivlin, E., and Rimon, E., ”A new range-sensor based globally convergent navigation algorithm for mobile robots”, *Proceedings of the IEEE ICRA*, Minneapolis, 1996, vol. 1, pp. 429-435.

[4] Magid, E., and Rivlin, E., ”Cautiousbug: a competitive algorithm for sensory-based robot navigation”, *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, vol. 3, pp. 2757-2762.

[5] Zapata, R., Lapierre, L. and Lepinay, P., ”Simultaneous path following and obstacle avoidance control of unicycle-type robot”, *Proceeding of IEEE ICRA*, Roma, Italy, 2007.

[6] Gil Pinto, A., ”Vers une architecture de commande des robots mobiles coopérants non holonomes”, *PhD thesis*, University of Montpellier 2, 2007.

[7] Defoort, M., Palos, J., Kokosy, A., Floquet, T., and Perruquetti, W., ”Performance-based reactive navigation for non-holonomic mobile robots”, *Robotica*, in press, 2008.