The Obstacle-Restriction Method (ORM) for Robot Obstacle Avoidance in Difficult Environments

Javier Minguez
Instituto de Investigación en Ingeniería de Aragón
Dept. de Informática e Ingeniería de Sistemas. Universidad de Zaragoza, Spain
jminguez@unizar.es

Abstract— **This paper addresses the obstacle avoidance problem in difficult scenarios, that usually are dense, complex and cluttered. The proposal is a method called the Obstacle-Restriction. At each iteration of the control cycle, this method addresses the obstacle avoidance in two steps. First there is procedure to compute instantaneous subgoals in the obstacle structure (obtained by the sensors). The second step associates a motion restriction to each obstacle, which are managed next to compute the most promising motion direction. The advantage of this technique is that it avoids common limitations of previous obstacle avoidance methods, improving their navigation performance in difficult scenarios. Furthermore, we obtain similar results to the recent methods that achieve navigation in troublesome scenarios. However, the new method improves their behavior in open spaces. The performance of this method is illustrated with experimental results obtained with a robotic wheelchair vehicle.**

I. Introduction

A large section of Robotics is currently focused on the development of applications where a vehicle performs in unknown and dynamic scenarios. The development of autonomous motion in these environments requires the usage of sensors to collect information on line, which is processed next to compute motion. Under these circumstances the usual selection to generate motion are the obstacle avoidance methods. These methods compute the goal-oriented collision-free motion based on the sensory input. Their advantage is that they adapt quickly the motion to the environmental changes. Currently, there is a demand of obstacle avoidance methods able to compute robust motion in realistic scenarios, which usually are difficult for obstacle avoidance since they are dense, complex and cluttered. Our work addresses this issue.

Many techniques have been proposed during the last years to perform obstacle avoidance. Some of these methods solve the problem by means of a physical analogy, like those of potentials [7], the gaseous substance [1], the fluids methods [8], the circulatory fields [15]. Other techniques are based on calculating a set of motion commands to select one of them next (e.g. steering angle field approach [5], vector field histogram [2], curvature velocity method [14], dynamic window approach [6]). Finally other techniques calculate some type of device to compute next the motion (e.g. elastic bands [13], elastic strips [4]). Although these techniques have been used with good results in many situations, in awkward scenarios there still arise problems as trap situations, instabilities or oscillations for example.

Recently, some researches have developed obstacle avoidance methods to close the gap of motion in troublesome scenarios [10], [12]. These methods employ a "divide and conquer" strategy based on situations to address difficult navigation cases. In this design, the motion laws associated to the situations only use partial information of the obstacle information available. This is enough to move the vehicle in dense scenarios, since the obstacle information used in these cases is the risky for the motion. However, in open spaces many information is not used, and the resulting trajectories are longer and less natural than expected. The idea behind the Obstacle-Restriction method (ORM in short) is to use all the obstacle information available in all the parts of the method. We achieve this goal in the two steps of the design of the method: first we use a local procedure that computes subgoals based on the obstacle distribution, and second we associate a motion restriction to each obstacle, which are managed later to compute the safest motion.

The advantage of this method is to avoid technical problems of classic methods like the local trap situations due to the obstacle structure (e.g. U-shape obstacles) or to the motion among very close obstacles, the oscillations or unstable motions. The method is able to compute directions far away from the goal direction and directions that point towards the obstacles, and it does not have internal parameters to tune. Furthermore, with this method we obtain similar results to the Nearness Diagram family of methods [10], [12] in very difficult (dense and complex) scenarios, but we improve their behavior in open spaces.

In the paper, we present first the ORM for obstacle avoidance (Section II). Next we outline the experimental results on a wheelchair vehicle (Section III), and we discuss and draw our conclusions next (Sections V).

II. The Obstacle-Restriction Method

We describe the method for a circular and omnidirectional robot, and we assume that the obstacle information is available in the form of points (usually the sensory information is given in this form, e.g. laser sensors).

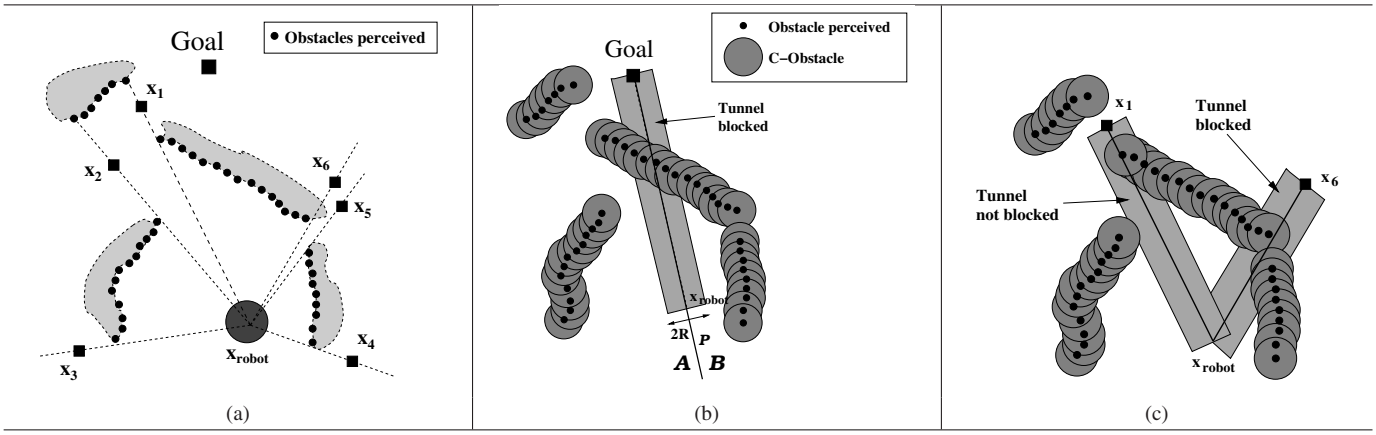


Fig. 1. This Figure illustrates the subgoal selector step of the ORM. (a) Robot, obstacle information perceived and the six candidate subgoals $x_1 \dots x_6$. (b) The tunnel to the goal is blocked, thus there is no path within the tunnel. The C-Obstacles are the obstacle points enlarged with the robot radius. (c) The tunnel to x_6 is also blocked, but the one to x_1 is not. Thus, there is a path that joins the current robot location and x_1. In this situation, x_1 is selected as the subgoal.

As all the obstacle avoidance methods, the ORM is based on a perception - action process that works at a high frequency (sampling period). Sensors collect the information of the environment, processed by the method to compute the control command. The vehicle executes the motion and the process restarts. For the ORM this process has two differentiated parts: the first one is a local selector of subgoals used to decide when, based on the structure of the surroundings, instead of directing the motion towards the goal it is better to direct it towards another location of the space (Subsection II-A). Secondly a motion constraint is associated to each obstacle, which are managed next to compute the motion command with a goal oriented strategy (Subsection II-B).

A. The Subgoal Selector

There are situations where it is more suitable to direct the motion towards a given zone of the space (that ameliorates the situation to reach the goal latter), rather than directly towards the goal itself. For example, in Figure 1a it is better to drive the vehicle towards location x_1 (where the robot fits in the passage and easily reaches the goal turning right-hand latter on), rather than moving directly to the goal (where there is an obstacle that blocks the way).

We present in this Subsection a procedure that decides whether the motion has to be directed towards the goal or towards an alternative subgoal. The procedure has two steps. First we search for locations suitable to place a subgoal, and next we select one of them (computing if it can be reached from the current location). The subgoals are located in between obstacles or in the edge of an obstacle:

1) In the middle point between two angular contiguous obstacle points whose distance is greater than the robot diameter (e.g. locations x_1 and x_2 in Figure 1a).
2) In the direction of the edge of an obstacle (obstacle point without contiguous) at a distance farther than the robot diameter (e.g. locations x_3, x_4, x_5 and x_6).

The result of this process is a list of candidate subgoals that capture the structure of the scenario.

The second step is to decide whether to use the goal for motion or to select a subgoal of the list. We do it by checking with a local algorithm whether the goal or a subgoal can be reached from the current robot location. To do this, we have developed a simple algorithm that checks the existence of a path that connects two locations.

Let $\mathbf{x_a}$ and $\mathbf{x_b}$ be two locations of the space, R the robot radius, and L a list of obstacle points, where $\mathbf{x_p^L}$ is an obstacle. The algorithm is:

1) Let be L' the list of points of L that are in the rectangle with height the segment $\overline{\mathbf{x_a x_b}}$ and width $2R$.
2) Let be $\mathcal{A}$ and $\mathcal{B}$ the two semiplanes divided by the line that joins $\mathbf{x_a}$ and $\mathbf{x_b}$. If for all the points of L', $d(\mathbf{x_j^L}, \mathbf{x_k^L}) > 2R$ (with $\mathbf{x_j^L} \in \mathcal{A}$ and $\mathbf{x_k^L} \in \mathcal{B}$) then return: *POSITIVE* else return: *NEGATIVE*.

The algorithm returns:

- *POSITIVE*: it exists a path joining $\mathbf{x_a}$ and $\mathbf{x_b}$, i.e. the final location can be reached.
- *NEGATIVE*: the final location cannot be reached within a local portion of the space (a rectangle that we call the *tunnel* of width the robot diameter $2R$, and height the segment that joins the two locations). This means that in this local area there is no path (although it could exist a global one).

The tunnel is blocked when there are two $\mathcal{C}$-Obstacles that belong to different semiplanes ($\mathcal{A}$ and $\mathcal{B}$) and intersect (the distance among the points is lower than $2R$). Thus there is no path within the tunnel. This is because an obstacle blocks the way (goal in Figure 1b) or because the robot does not fit in the passage (x_6 in Figure 1c). However, the interesting result is when the tunnel is not blocked, because then it exists a local path in the tunnel and thus a global path that joins both locations (e.g. x_1).

In order to select a subgoal we first use the algorithm with the goal. If the result is *NEGATIVE*, we chose the closest subgoal to the goal that has a path that reaches it. For example, in Figure 1 we try first with the $\mathbf{x_{goal}}$ with *NEGATIVE* result. Then we try with x_1 and the result is *POSITIVE*. Thus we select x_1 as location to direct the vehicle (instead of $\mathbf{x_{goal}}$). Notice that this was the result desired outlined in the beginning of this Subsection.

We depict next how with this simple algorithm we avoid two common situations that causes local minima in many of the obstacle avoidance methods:

1) The U-shape obstacle (Figure 2a). In this case the algorithm returns *NEGATIVE* when is used with the goal location. However, the solution is *POSITIVE* for both subgoals x_1 or x_2, and thus one of them would be selected to direct the vehicle. Notice that moving towards these subgoals avoids entering in the U-shape obstacle and thus the trap situation.
2) Motion between very close obstacles (Figure 2b). Here there are two potential passages to traverse in order to reach the goal. However the algorithm returns *NEGATIVE* for subgoal x_1 because the tunnel is blocked (the robot does not fit in the *Passage* 1). On the other hand the algorithm selects subgoal x_2 (because the robot fits in *Passage* 2). Notice that moving towards x_2 avoids the trap situation or a collision due to a wrong selection of the passage.

The complexity of the algorithm is $\mathcal{O}(N^2)$, with N the number of obstacle points. However, in practice the algorithm is very efficient since there are always a few number of points in the tunnel, and thus it is very suitable for real-time implementations. Furthermore, notice that the usage of the algorithm is combined with a procedure that locates the subgoals in the places to capture the structure of the scenario.

B. Motion Computation

In the previous step we have seen how to compute an instantaneous subgoal (if required) based on the structure of the obstacle information and the goal location. We call target the goal or subgoal selected in the previous step. We present next a procedure to compute the motion towards the target direction while avoiding collisions with the obstacles. In a first step we calculate a set of constraints of motion for each obstacle, and next we manage all of them in order to compute the final direction to move.

Let the frame of reference be the robot frame. Let R be the radius of the robot, D_s a security distance around the robot bounds and θ_{target} the direction of the target location.

Step 1: The motion constraints

For each obstacle we compute a set of directions $\mathcal{S}_{nD} \in [-\pi, \pi]$ that are not desirable for motion (motion constraint). This set is computed as the union of two subsets $\mathcal{S}_1$ and $\mathcal{S}_2$. $\mathcal{S}_1$ represents the side of the obstacle not suitable to do the avoidance, and $\mathcal{S}_2$ an exclusion area around the obstacle. Let be θ_{obst} and d_{obst} the direction and distance to the obstacle:

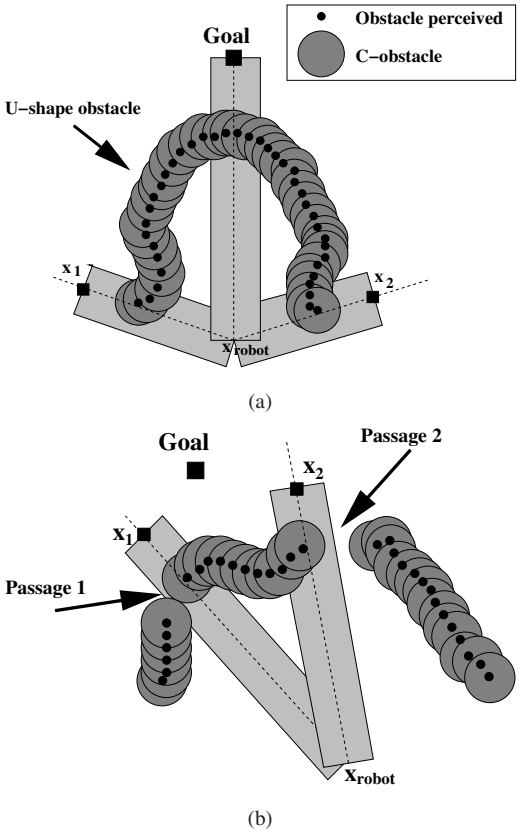


Fig. 2. Two problematic cases for obstacle avoidance: (a) U-shape obstacle and (b) an open passage where the robot does not fit. In both cases the algorithm selects a subgoal to reach the goal that avoids the trap situation.

$$\mathcal{S}_1 = \begin{cases} (\theta_{obst}, \pi), & if \quad \theta_{target} < \theta_{obst} \\ (-\pi, \theta_{obst}), & i.o.c. \end{cases} \tag{1}$$

From the robot, each obstacle has two sides. This set contains all the directions on the opposite side of the target. For instance in Figure 3a the target is in the left-hand side of the obstacle, and then $\mathcal{S}_1$ contains all the directions on the right-hand side, and vice versa in Figure 3d.

The second subset of directions $\mathcal{S}_2 = [\gamma_L, \gamma_R]$, where:

$$\gamma_L = max[\theta_{obst} - (\alpha + \beta), -\pi] \tag{2}$$

$$\gamma_R = min[\theta_{obst} + (\alpha + \beta), \pi] \tag{3}$$

where α and β are given by,

$$\alpha = |atan(\frac{R + D_s}{d_{obs}})| \tag{4}$$

$$\beta = \begin{cases} (\pi - \alpha).(1 - \frac{d_{obst} - R}{D_s}) & if \quad d_{obst} \leq D_s + R \\ 0 & i.o.c \end{cases} \tag{5}$$

This set of directions $\mathcal{S}_2$ creates an exclusion area around the obstacle. The directions $\theta_{obst} \pm \alpha$ are the directions that remove the obstacle of the security zone at the height of the obstacle (Figure 3b). Furthermore, we add a term β when the obstacle is in the security zone. Angle β ranges from 0 to $\pi - \alpha$ depending of how close the obstacle is to the robot bounds. For instance in Figure 3e the obstacle is closer than

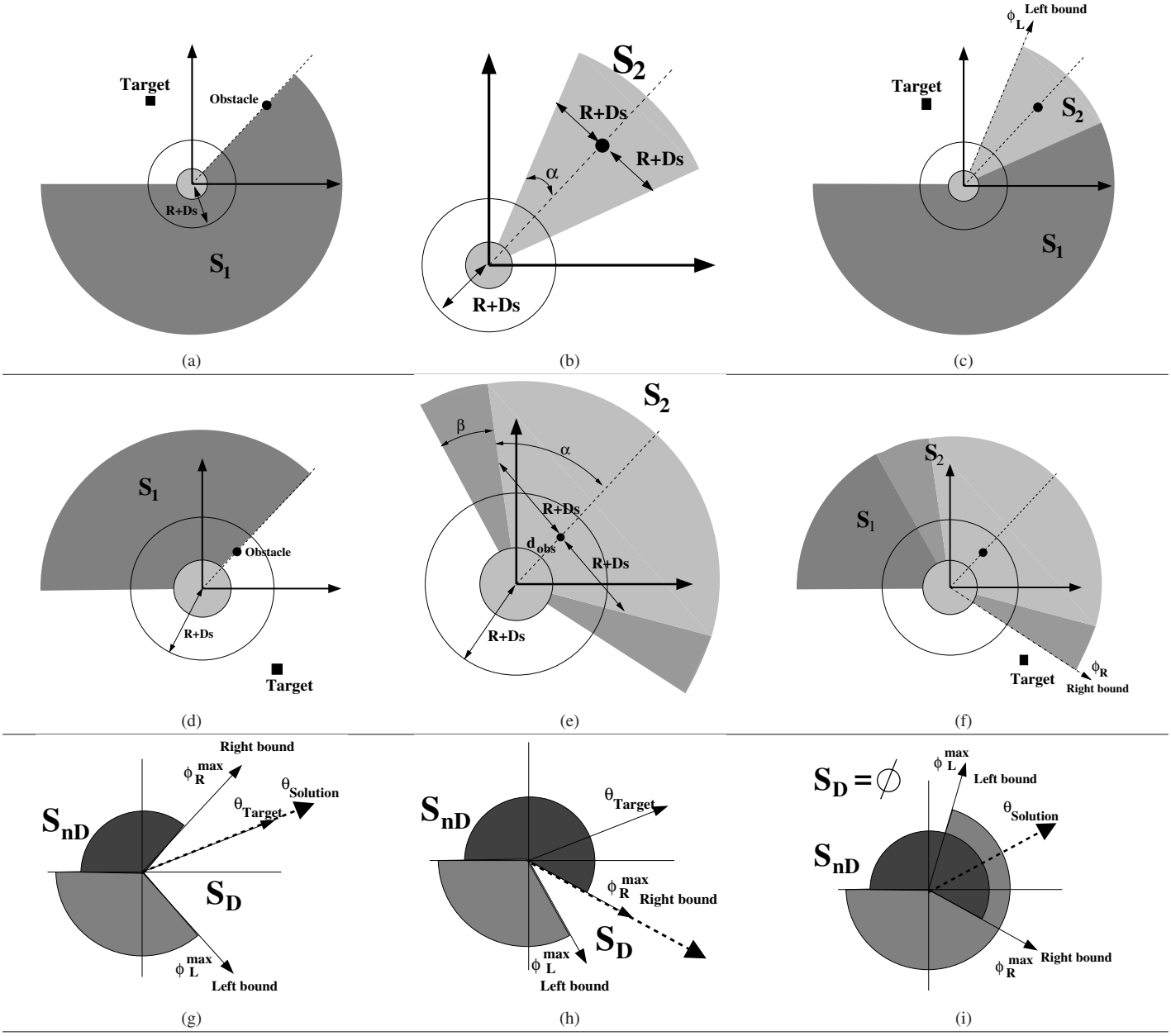


Fig. 3. (a,b,c) Set of motion constraints for an obstacle out of the security distance and with the goal located in the left-hand side. (d,e,f) Set of motion constraints for an obstacle in the security distance and with the goal located in the right-hand side. (g,h,i) Computation of the direction solution in the three possible cases.

the security distance and thus there are additional directions in $\mathcal{S}_2$ due to the proximity of the obstacle. Notice that $\beta = 0$ when the obstacle is at a distance of $D_s + R$ in order to have continuity in the limit of the security distance.

The motion constrain for the obstacle is the union of both sets $S_{nD} = S_1 \cup S_2$ (Figures 3c,f).

A feature that will be used is the concept of left and right bounds of S_{nD}. When $\theta_{target} > \theta_{obst}$ (the set $\mathcal{S}_1$ is on the right-hand side of the obstacle) we call the $\phi_L = max(S_{nD})$ the *left bound* (Figure 3c), and in the opposite case $\phi_R = min(S_{nD})$ is the *right bound* (Figure 3f).

Step 2: Selecting one direction of motion

Up to now we have discussed how an obstacle point has associated a set of motion directions that are not suitable. By joining all the sets of all the obstacles we get the full set of motion constraints. That is, if for each obstacle i we compute $\mathcal{S}^i_{nD} = S^i_1 \cup S^i_2$, the final set is $\mathcal{S}_{nD} = \cup_i \mathcal{S}^i_{nD}$. The set of desired directions of motion is the complementary $S_D = \{[-\pi, \pi] \setminus \mathcal{S}_{nD}\}$. Let be $\phi_L^{max} = max(\phi_L^i)$ and $\phi_R^{max} = min(\phi_R^i)$. There are three cases to select the motion direction, θ_{sol}:

1) If the set $S_D \neq \emptyset$ and $\theta_{target} \in S_D$ then $\theta_{sol} = \theta_{target}$ (Figure 3g).
2) If the set $S_D \neq \emptyset$ and $\theta_{target} \notin S_D$ then:

$$\theta_{sol} = \begin{cases} \phi_R^{max} & if\ |\theta_{target} - \phi_R^{max}| < |\theta_{target} - \phi_L^{max}| \\ \phi_L^{max} & i.o.c \end{cases} \quad (6)$$

The closest left bound ϕ_L^{max} or right bound ϕ_R^{max} to

the target direction is selected (e.g. in Figure 3h we select the right bound for motion).

3) If final set $S_D = \emptyset$ then $\theta_{sol} = \frac{\phi_R^{max} + \phi_L^{max}}{2}$. We compute the medium value between the left and right bounds (Figure 3i).

As a result we obtain the most promising direction of motion θ_{sol}.

In summary, we have described a method that, given the current perception (obstacle information) and a goal location, computes the most promising motion direction to avoid collisions while converging the robot location towards the target.

III. Experimental results

For experimentation, we used a commercial wheelchair that we have equipped with two on-board computers, and with a SICK laser. The vehicle is rectangular $(1.2 \times 0.7 meters)$ with two tractor wheels that work in differential-driven mode. We set the maximum operational velocities to $(v_{max}, w_{max}) = (0.3\frac{m}{sec}, 0.7\frac{rd}{sec})$ due to the application context (human transportation). The ORM runs at a medium of $1000Hz$ on the on-board $PentiumIII850Mhz$. However, we reduced this frequency to $5Hz$ (frequency of the laser) sleeping the process in order to have a perception-action scheme.

Notice that we have presented the obstacle avoidance method assuming that the robot can move in any direction and is circular. In order to adapt this technique to the rectangular and non-holonomic robot, we followed the solution proposed in [9]: in a first step, the direction solution computed by the obstacle avoidance method is converted into commands that comply with the kinematics and dynamics using a kinodynamic model of the robot. Next, these commands are modified if collisions appear due to the robot shape (rectangular here). As a result, the commands computed tend to align the vehicle with the instantaneous direction of motion whilst avoiding collisions.

We outline next two experiments in unknown, unstructured and dynamic office-type environment, where the goal location was the only information provided in advance.

In the first experiment, the robot moved avoiding two large U-shape structures (formed by furniture) to reach the goal location (see a snapshot in Figure 4a and the complete trajectory and the laser points in Figure 4d). The first difficulty avoided was a narrow passage where the robot did not fit in (*Passage* 1 in Figures 4a,b that seems to be closed because it is blurred due to the robot drift). On-line, it was detected that that the vehicle did not fit in and moved to the right-hand avoiding the large U-shape structure formed by the furniture and this narrow passage. To overcome this situation, motion directions far away from the goal direction (some of them differ in more than $90°$) were computed. Next, the vehicle moved to the bottom part to cross the narrow corridor, whilst avoided moving within the U-shape structure to the right-hand side. The next difficulty was found in the central corridor, since it was very narrow (*Passage* 2 in Figures 4a,b). Again on-line, the vehicle detected that it was possible to cross this narrow place and proceeded to do it. Notice that while the vehicle was moving within the corridor there was very little room to maneuver ($< 10cm$ on both sides). However, there were not oscillations or unstable behaviors. Finally the robot reached the goal location without collisions. During the experiment, directions of motion towards obstacles were selected in almost all the experiment, since the laser has a large field of view ($8m$), about the office size. The computational load of the algorithm in each cycle was almost constant ($\simeq 0.001sec$), the experiment was accomplished in $102sec$ and the velocity profiles are shown in Figure 4c.

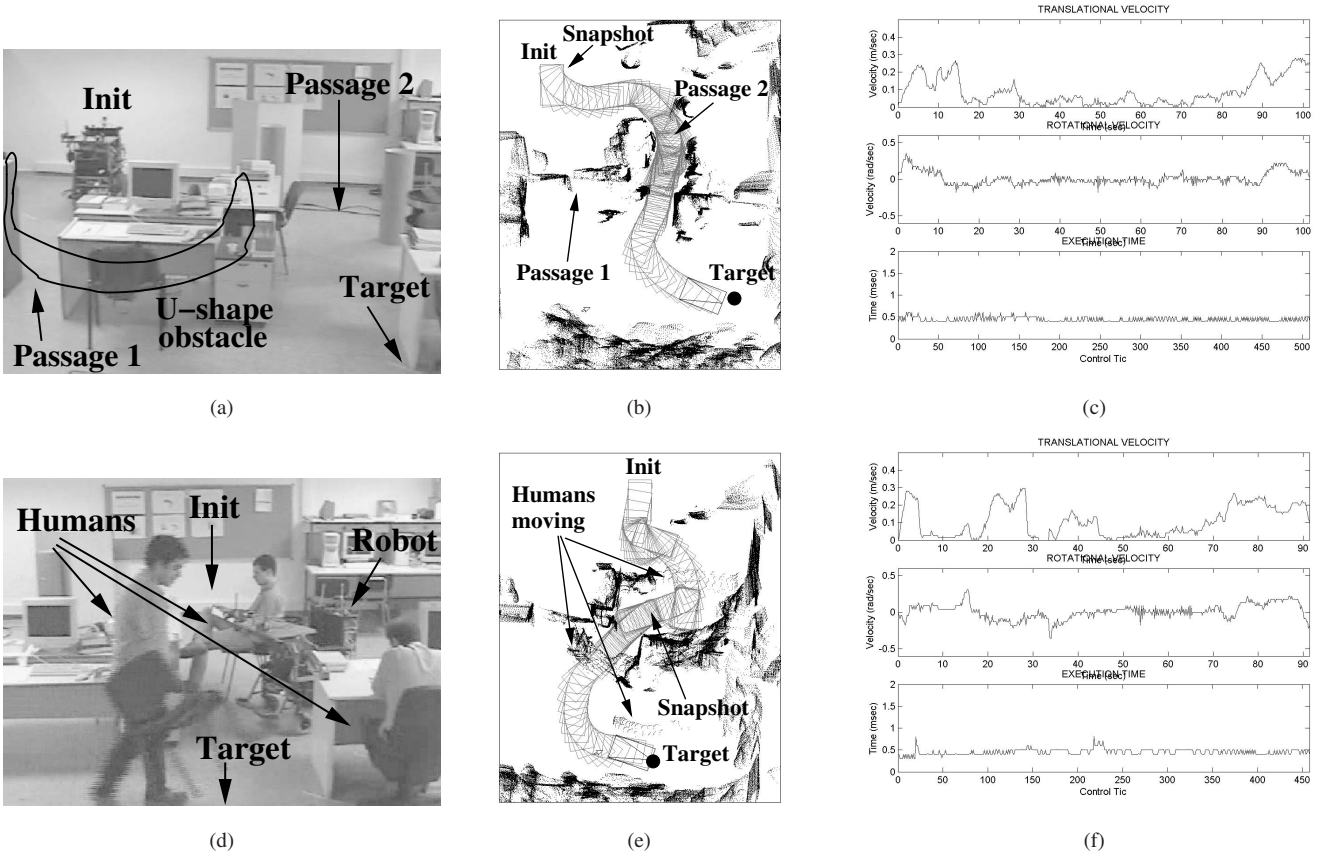
In the next experiment we tested the method in the office but with humans moving around (what converted the scenario in a dynamic place), Figures 4d,e. First, the vehicle maneuvered to the right-hand to avoid the human that moved towards the table on the left-hand and took a sit. Then, it was checked that the vehicle fitted in the very narrow passage (the same size of the previous experiment) and thus proceed to cross it. At the exit of the passage, other humans hindered the vehicle motion. However, collisions were avoided with smooth motions (see the velocities profiles, Figure 4f). Finally, the vehicle reached the goal location. The experiment was accomplished in $92sec$ and velocity profiles are shown in Figure 4f.

IV. Discussion

We discuss next the advantages of this method with respect to existing techniques on the basis of the difficulties shown in the experimental results.

The **local trap situations** due to U-shape obstacles or due to the motion among close obstacles are overcome with this method. The ORM does not direct the vehicle within U-shape obstacles since the first step of the method places subgoals out of these obstacles (Figure 2a). Thus, these situations are avoided in a natural way. For example, in the first experiment the vehicle avoided the large U-shape structure. Secondly, the method drives the vehicle among very close obstacles because: (i) the possibility of whether the vehicle fits is checked with the subgoal selector (Figure 2b). For example in the first experiment, *Passage* 1 was not selected for motion (since the vehicle did not fit in) and the vehicle proceeded through *Passage* 2. And (ii) the motion centers the vehicle among obstacles. This is because the motion is obtained as the bisector of the extreme constrains of motion (which are generated by the most risky obstacles). Notice that *Passage* 2 is very narrow (Figures 4a,b). Furthermore, when moving among very close obstacles we observed **motion free of oscillations or instabilities** (see the smooth paths generated in Figure 4b,e and the velocity profiles of the experiments Figures 4c,f), which in addition rely on a comfortable motion with the wheelchair.

The ORM **selects motion directions far away from the goal direction** when required. This is because the subgoals can be placed in any location in the space. In other words, any deviation from the goal direction can be obtained with this method. This property was determinant in the two



(a) (b) (c)

(d) (e) (f)

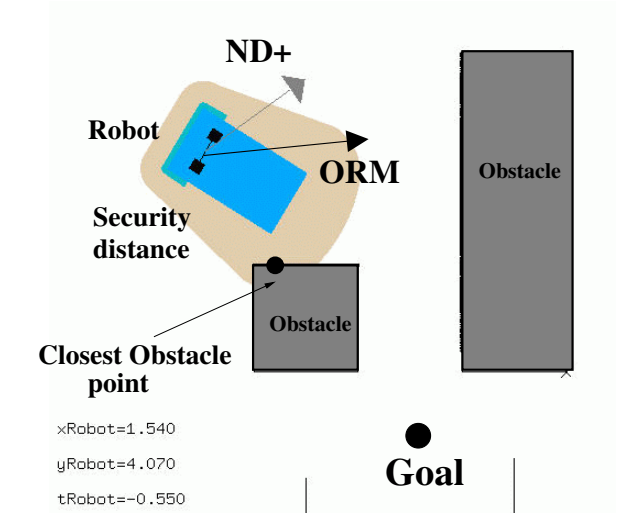
Fig. 4. This Figure depicts two experiments with the ORM. (a,d) Snapshots of the experiment. (b,e) Trajectory of the vehicle and laser points perceived. (c,f) Velocity profiles of the experiment and computational load of the method.

experiments. For example, in the beginning of both, the only way to reach the target was to move right-hand, that implies directions far from the goal direction. In addition, in the ORM formulation, the **selection of directions of motion towards the obstacles** is possible (which is usually the case in $S_D = \emptyset$). This issue was critical when moving in narrow corridors, because sometimes only this type of directions allows to move safely.

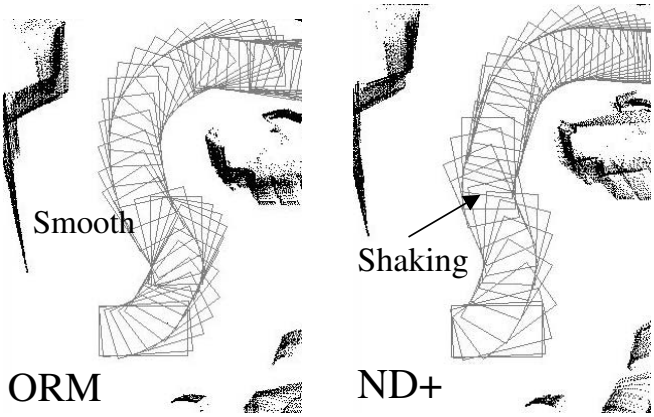
The ORM has no **internal parameters**. Only the security distance has to be set with a coherent value (we chose $D_s = 0.75m$, the shorter side of the vehicle). Another point is the computational load because it sets the reactivity of the system. The execution time of the ORM is in medium $0.001sec$ (Figure 4c,f) on a $PentiumIII850Mhz$. In other words, once a new sensor reading is available, in $0.001sec$ there is a motion command ready to be executed (thus **the reaction is immediate**). Another advantage is that the processor is free the majority of the time (the obstacle avoidance method consumes less than 2.5% of the CPU), which could be employed by other modules that require higher computational loads (e.g. path planning, simultaneous location and map building, supervisors, etc).

We have seen how this method overcomes many of the limitations and problems of previous obstacle avoidance methods, which leads to the outstanding results obtained in difficult scenarios. However, we still have to compare the ORM with the ND methods [10], [12], since they also have demonstrated to successfully navigate in these scenarios. Notice that it is difficult to compare in rigorous terms (there is no metric to measure the quality of these methods). Thus, we direct the comparison in qualitative terms.

The ND method is based on a "divide and conquer" strategy based on situations. At each time, one navigation situation is selected and the associated action (law of motion) computes the motion. In the method design, the motion laws only use partial information of the obstacle information available. This is enough to move in dense scenarios, however in open spaces much information is not used. As a result, the trajectories are longer and less natural than expected. For example, Figure 5a depicts the direction solution of the ND+ method in a given scenario. The action in this situation computes the solution only taking into account the closest point of the frontal obstacle and the borders of the passage. However, in this case the ORM computes a better solution. This is because, the ORM also considers the obstacle on the right-hand, and thus it corrects the solution towards the passage. Figures 5b,c depict the partial trajectory of an experiment carried out with both methods driving the wheelchair. Notice how there are some shaking behaviors (oscillations) with the ND+ that are not obtained with the ORM. This is also because all the obstacle information is considered in the ORM method. Then, since this information does not change significantly between sampling periods, the



(a)



(b) (c)

Fig. 5. (a) ORM and the ND+ method solutions in a given situation. (b) and (c) Real trajectories of the wheelchair using both methods.

resulting changes in direction are smooth. Our impression is that with the ORM we achieve better results in open spaces than the ND+ method and similar results in places with little room to maneuver.

V. Conclusions

This paper presents the Obstacle-Restriction method for robot obstacle avoidance in difficult scenarios. The method proposed has two steps. The first one is a procedure to compute instantaneous subgoals based on the obstacle structure. The second step associates a motion restriction to each obstacle, which are managed next to compute the most promising motion direction.

The advantage of this technique is that it avoids common limitations of previous obstacle avoidance methods, improving their navigation performance in difficult scenarios. Furthermore, we obtain similar results to the ND family of methods in troublesome scenarios. However, the new method improves the behavior in open spaces.

We want to conclude remarking that the usage of this method does not avoid the problems inherent to the local nature of obstacle avoidance methods: the global trap situations persist. This issue is far beyond the scope of this work, however, this method could be used together with techniques that aim to increase the locality of obstacle avoidance methods, such as those described in [17], [3], [11], [16]. Then, these undesirable situations would be mitigated.

VI. Acknowledgments

We want to thank Javier Osuna, José Luis Villarroel and Luis Montano for participating in this project with technical discussions and contributing with many ideas. This research was supported by Spanish project MCYT DPI2003-07986.

References

[1] K. Azarm and G. Schmidt. Integrated mobile robot motion planning and execution in changing indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 298–305, Munchen, Germany, 1994.

[2] 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.

[3] O. Brock and O. Khatib. High-Speed Navigation Using the Global Dynamic Window Approach. In *IEEE Int. Conf. on Robotics and Automation*, pages 341–346, Detroit, MI, 1999.

[4] O. Brock and O. Khatib. Real-Time Replanning in High-Dimensional Configuration Spaces using Sets of Homotopic Paths. In *IEEE Int. Conf. on Robotics and Automation*, pages 550–555, San Francisco, USA, 2000.

[5] W. Feiten, R. Bauer, and G. Lawitzky. Robust Obstacle Avoidance in Unknown and Cramped Environments. In *IEEE Int. Conf. on Robotics and Automation*, pages 2412–2417, San Diego, USA, 1994.

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

[7] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. Journal of Robotics Research*, 5:90–98, 1986.

[8] A. Masoud, S. Masoud, and M. Bayoumi. Robot navigation using a pressure generated mechanical stress field, the biharmonical potential approach. In *IEEE International Conference on Robotics and Automation*, pages 124–129, San Diego, USA, 1994.

[9] J. Minguez and L. Montano. Robot Navigation in Very Complex Dense and Cluttered Indoor/Outdoor Environments. In *15th IFAC World Congress*, Barcelona, Spain, 2002.

[10] J. Minguez and L. Montano. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.

[11] J. Minguez, L. Montano, N. Simeon, and R. Alami. Global Nearness Diagram Navigation (GND). In *IEEE International Conf. on Robotics and Automation*, pages 33–39, Seoul, Korea, 2001.

[12] J. Minguez, J. Osuna, and L. Montano. A Divide and Conquer Strategy to Achieve Reactive Collision Avoidance in Troublesome Scenarios. In *IEEE International Conference on Robotics and Automation*, Minessota, USA, 2004.

[13] S. Quinlan and O. Khatib. Elastic Bands: Connecting Path Planning and Control. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 802–807, Atlanta, USA, 1993.

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

[15] L. Singh, H. Stephanou, and J. Wen. Real-time robot motion control with circulatory fields. In *IEEE International Conference on Robotics and Automation*, pages 2737–2742, Mineapolis, USA, 1996.

[16] C. Stachniss and W. Burgard. An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 508–513, Switzerland, 2002.

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