Reactive Obstacle Avoidance for Mobile Robots that Operate in Confined 3D Workspaces

***Abstract*— This paper addresses the obstacle avoidance problem for robots that operate in confined three-dimensional workspaces. The contribution of this research summarizes in: (i) the generalization of an existing technique to deal with $3D$ workspaces, (ii) an implementation of the method, and (iii) the experimental validation with an special emphasis on dense, cluttered and confined scenarios, which are the difficulty of the existing techniques. We show results that illustrate how this method successfully performs obstacle avoidance, and how it inherits the benefits of the original method being able to overcome the typical challenging problems of other obstacle avoidance methods but extrapolated to three dimensions.**

I. Introduction

This work addresses the autonomous motion aspect within a robotic rescue project. One example of application is a disaster area under a collapsed house where a robot searches for victims. From a robotic perspective, in these scenarios there is no a priori knowledge of a map since the robot might be the first entity to go in. These environments are usually unstructured since they are composed by randomly distributed pieces of material. Furthermore, the scenarios can evolve since one can find people working in the area, new collapses could arise, or some areas could be dynamically modified by fire for example. In other words, the scenarios are unknown, unstructured and dynamic. To autonomously move a vehicle under these circumstances, the techniques used have to use sensors in order to collect information of the scenario and to adapt the motion to any new contingency or event. In these scenarios, the natural choice for motion is the obstacle avoidance methods. Furthermore, from the technical point of view, these scenarios share two characteristics/difficulties. The first one is that motion cannot be restricted to a plane. Motion techniques have to deal with full $3D$ workspaces. The second one is that the scenarios are dense, cluttered and confined. Motion techniques have to be robust under these work conditions. This paper presents a technique to perform robust obstacle avoidance under both circumstances.

Motion using the onboard sensors in $3D$ environments has been addressed from two perspectives: dynamic planning and reactive obstacle avoidance. On one hand, dynamic planning techniques are based on a process that dynamically replan segments of paths based on sensor inputs [14], [6]. The advantage is that they are efficient and complete. However their performance heavily rely in the construction of an on-line 3D model that could be difficult to build under some circumstances (like the rescue context mentioned above).

On the other hand, the obstacle avoidance methods only use the current information gathered by the sensors (pure reactive methods). Their disadvantage is the local nature, thus global traps are difficult to avoid. However, they have demonstrated to efficiently adapt the motion to the sensory input. This is desirable in many motion contexts (one possibility is the unknown, unstructured and evolving disaster scenario).

Focusing in these last techniques, in $2D$ workspaces, many techniques have been proposed. Some of these methods solve the problem by means of a physical analogy [5], [1], [9]. Other techniques are based on calculating a set of motion commands to select one of them next [2], [13], [4]. Finally, other techniques calculate some type of device to compute next the motion [12], [3]. Although these techniques have been used with good results in many situations, in confined scenarios there still arise problems such as local traps, instabilities or oscillations for example (see [7], [11] for a discussion). This is why some researches have recently worked to close the gap of motion in troublesome scenarios [11], [10]. These last methods have demonstrated to generate robust motion in dense, complex and confined environments.

Very few methods [5], [12], [3] deal with $3D$ workspaces. If one assumes that the well-known limitations of these methods ([7], [11]) will be inherited from $2D$ to $3D$ workspaces, problems would arise when dealing with dense and complex scenarios. Furthermore, none of the techniques designed to deal with troublesome and confined scenarios [11], [10] has been extended to deal with the third dimension. This is the motivation and starting point of this research.

This paper describes the formulation of the Obstacle Restriction Method (ORM) [10] to work in three-dimensional workspaces. The contribution of this research summarizes in: (i) the generalization of the method to deal with $3D$ workspaces, (ii) an implementation of the method, and (iii) the experimental validation with an special emphasis on dense, complex and confined scenarios. We show results that illustrate how this method successfully performs obstacle avoidance overcoming typical problems of existing methods.

II. The Obstacle Restriction Method

In this Section we describe the formulation of the ORM to work in three-dimensional workspaces. We assume a spherical and omnidirectional robot (free flying object). The obstacle information is given in the form of points, which is the usual form in which sensor data is given (e.g. laser sensors).

Obstacle avoidance methods are based on an iterative perception – action process. Sensors collect information of the environment that is processed by the method to compute a motion command. The motion is executed by the robot and then the process restarts again. The ORM has two steps:

1) **Subgoal selector**: this procedure decides if the motion should be directed towards the goal or if it is better to direct the motion towards another location in the space (Subsection II-A).

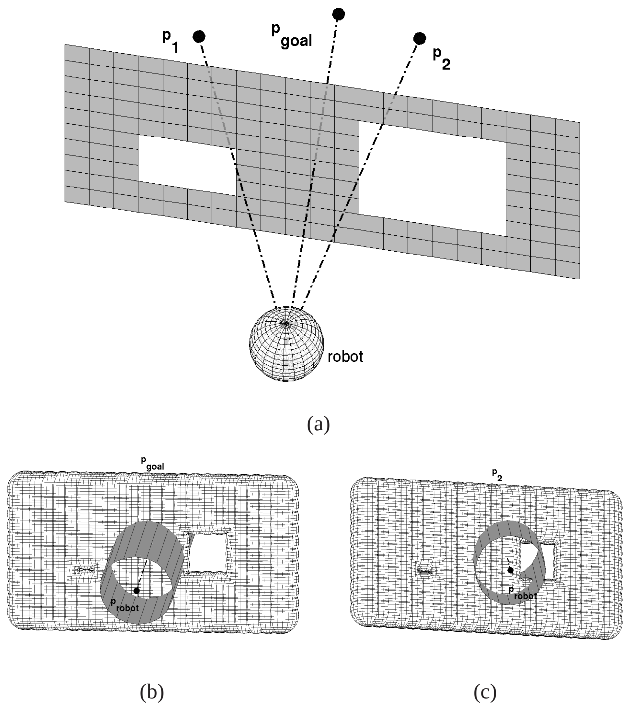

Fig. 1. This Figure depicts the subgoal selector step. (a) Goal location and robot facing a wall with two holes (the robot fits in the right-hand one but not in the left-hand one). (b) There is no path that joins the robot location with $\mathbf{p_{goal}}$ within the tunnel. (c) The tunnel is not blocked, thus it exists a path to $\mathbf{p_2}$. This subgoal is selected for the next step.

2) **Motion computation**: this procedure associates a motion constraint to each obstacle, which are managed next to compute the most promising motion direction (Subsection II-B).

We outline next both steps.

A. The Subgoal Selector

We present in this Subsection a procedure that decides whether the motion has to be directed towards the goal or towards an alternative subgoal. For example, in Figure 1a it is better to drive the vehicle towards location $\mathbf{p_2}$ (where the robot fits in the hole and easily reaches the goal $\mathbf{p_{goal}}$ turning left-hand latter on), rather than moving directly to the goal (where there is an obstacle that blocks the way).

The procedure has two steps. First we search for locations suitable to place a subgoal. They 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. among obstacles).
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 $\mathbf{p_1}$ and $\mathbf{p_2}$).

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 (the description of this algorithm with the mathematical demonstration is long and with more room would be included in an Appendix). In short, the algorithm searches the existence of a path that connects the two locations by checking if a local portion of the space (the *tunnel*) that joins them is blocked by obstacles in configuration space [8]. The algorithm returns:

- *NEGATIVE*: The tunnel is blocked. There is no local path joining both locations within the tunnel.
- *POSITIVE*: The tunnel is not blocked. There exists a set of homotopic paths joining both locations in the tunnel.

In order to select a subgoal we first use the algorithm with the goal. If the result is *NEGATIVE*, we choose the closest subgoal to the goal that has a path that reaches it. For instance, in Figure 1a we try first with $\mathbf{p_{goal}}$ with *NEGATIVE* result (Figure 1b). Then, we try with $\mathbf{p_1}$ with *NEGATIVE* result. Finally, we obtain *POSITIVE* with $\mathbf{p_2}$ (Figure 1c). This point is selected as subgoal.

In summary, the result of this process is a subgoal that can be reached from the current robot location.

B. Motion Computation

From now on we refer to $\mathbf{p_{target}}$ the subgoal computed in the previous Subsection. In this Subsection we first introduce the space division that will be used in the next subsections. Next, we discuss the motion computation that has two steps: first we compute a set of motion constraints for each obstacle and second we manage all the sets to compute the most promising direction of motion.

Let R be the radius of the robot and D_s a security distance around the robot bounds. Let the frame of reference be the robot frame with origin $\mathbf{p_0} = (0,0,0)$ and unitary vectors $(\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z})$, where $\mathbf{e_x}$ is aligned with current the main direction of motion of the vehicle. Depending on the relation between the robot configuration and the target direction we divide the space in four quadrants as follows. Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be the planes defined by $[\mathbf{p_0}, \mathbf{e_x}, \mathbf{e_z}]$, $[\mathbf{p_0}, \mathbf{e_z}, \mathbf{p_{target}}]$ and $[\mathbf{p_0}, \mathbf{e_y}, \mathbf{p_{target}}]$; and $\mathbf{n}_\mathcal{A} = \mathbf{e_y}$, $\mathbf{n}_\mathcal{B} = \mathbf{e_z} \otimes \mathbf{p_{targ}}$ and $\mathbf{n}_\mathcal{C}\mathbf{p_{targ}} \otimes \mathbf{e_y}$ be the normal to these planes respectively. Then, let $\mathbf{u}$ be a given vector, the quadrant is $\Omega(\mathbf{u}) =$

$$= \left(\begin{array}{lll} \text{TL} & \text{if} & (\mathbf{u}\cdot\mathbf{n}_\mathcal{A} \geq \mathbf{0})\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{B} \geq \mathbf{0})\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{C} \geq \mathbf{0}) \\ \text{TR} & \text{if} & ((\mathbf{u}\cdot\mathbf{n}_\mathcal{A} < \mathbf{0})\,|\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{B} < \mathbf{0}))\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{C} \geq \mathbf{0}) \\ \text{BL} & \text{if} & (\mathbf{u}\cdot\mathbf{n}_\mathcal{A} \geq \mathbf{0})\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{B} \geq \mathbf{0})\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{C} < \mathbf{0}) \\ \text{BR} & \text{if} & ((\mathbf{u}\cdot\mathbf{n}_\mathcal{A} < \mathbf{0})\,|\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{B} < \mathbf{0}))\,\&\,(\mathbf{u}\cdot\mathbf{n}_\mathcal{C} < \mathbf{0}) \end{array} \right. \tag{1}$$

where Figure 2 shows an example. We address next the two steps that compute the motion.

1) The motion constraints: In the ORM each obstacle creates a motion constraint. A motion constraint is a set of motion directions that are not desirable for motion S_{nD}. This set is computed as the union of two subsets S_1 and S_2. S_1 represents the side of the obstacle which is not suitable for avoidance, while S_2 is an exclusion area around the obstacle. We describe next the computation of the first set of subset of directions S_1 for a given obstacle point $\mathbf{p_{obst}}$.

Let $\mathcal{D}$ be the plane defined by $[\mathbf{p_0}, \mathbf{p_{obst}}, \mathbf{p_{target}} \otimes \mathbf{p_{obst}}]$, and $\mathbf{n}_\mathcal{D} = (\mathbf{p_{target}} \otimes \mathbf{p_{obst}}) \otimes \mathbf{p_{obst}}$ the normal to this plane.

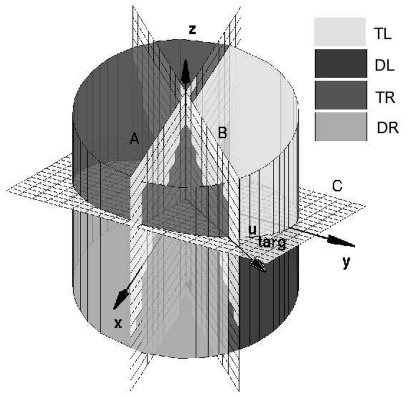

Fig. 2. This Figure depicts the division of the space in four quadrants.

The first set of constraints is:

$$S_1 = \left(\begin{array}{ll} (\mathcal{A}^+ \cap \mathcal{B}^+) \cap \mathcal{D}^+ & \text{if } \Omega(\mathbf{p_{obst}}) = [TL, BL] \\ (\mathcal{A}^+ \cup \mathcal{B}^+) \cap \mathcal{D}^+ & \text{if } \Omega(\mathbf{p_{obst}}) = [TR, BR] \end{array} \right. \quad (2)$$

where $\mathcal{A}^+$, $\mathcal{B}^+$ and $\mathcal{D}^+$ are sets of directions:

$$\mathcal{A}^+ = \{\mathbf{p} \mid \mathbf{p}_\mathcal{A} \cdot \mathbf{p} \geq 0 \text{ and } \Omega(\mathbf{p_{obst}}) = [TL, BL]$$
$$\text{or } \mathbf{p}_\mathcal{A} \cdot \mathbf{p} < 0 \text{ and } \Omega(\mathbf{p_{obst}}) = [TR, BR] \ \}$$
$$\mathcal{B}^+ = \{\mathbf{p} \mid \mathbf{n}_\mathcal{B} \cdot \mathbf{p} \geq 0 \text{ and } \Omega(\mathbf{p_{obst}}) = [TL, BL]$$
$$\text{or } \mathbf{p}_\mathcal{B} \cdot \mathbf{p} < 0 \text{ and } \Omega(\mathbf{p_{obst}}) = [TR, BR] \ \}$$
$$\mathcal{D}^+ = \{\mathbf{p} \mid \mathbf{p}_\mathcal{A} \cdot \mathbf{p} > 0 \ \}$$

The second set is:

$$S_2 = \{\mathbf{p} \mid \arccos \frac{\mathbf{p} \cdot \mathbf{p_{obst}}}{||\mathbf{p}|| \cdot ||\mathbf{p_{obst}}||} \leq \gamma\} \quad (3)$$

where $\gamma = \alpha + \beta$,

$$\alpha = |atan\left(\frac{R + D_s}{||\mathbf{p_{obst}}||}\right)|$$

$$\beta = \left\{ \begin{array}{ll} (\pi - \alpha)\left(1 - \frac{||\mathbf{p_{obst}}|| - R}{D_s}\right) & \text{if } ||\mathbf{p_{obst}}|| \leq D_s + R, \\ 0 & \text{otherwise} \end{array} \right.$$

Finally, the motion constraint for obstacle $\mathbf{p_{obst}}$ is the union of both subsets:

$$S_{nD} = S_1 \cup S_2$$

Another important feature is the boundary of the constraint, that we compute as:

$$S_{bound} = \{\mathbf{p} \mid \mathbf{p} \in \mathcal{B}(S_2) \text{ and } \mathbf{p} \notin S_1\}$$

where $\mathcal{B}$ is the boundary of a given set. Figure 3 depicts an example of all these features.

The next step is to compute the motion constraints for obstacle points in each quadrant. Let $\mathbf{p_i}$, $i = 1 \ldots m$ be the obstacle points that belong to a quadrant $\mathcal{G} = \Omega(\mathbf{p_i})$ (expression (1)). Then, the motion constraints for all the obstacles in quadrant $\mathcal{G}$ is

$$\mathcal{S}_{nD}^\mathcal{G} = \cup_{i=1}^m S_1^i \cup S_2^i \quad (4)$$

and the bound is

$$\mathcal{S}_{bound}^\mathcal{G} = \{\mathbf{p} \mid \mathbf{p} \in \mathcal{B}(\cup_{i=1}^m S_2^i) \text{ and } \mathbf{p} \notin \cup_{i=1}^m S_1^i\}$$

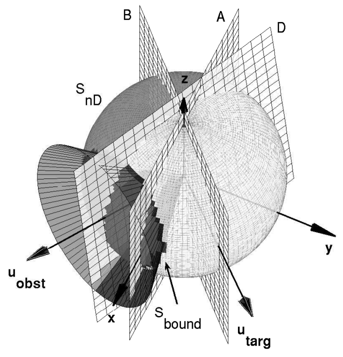

Fig. 3. This Figure depicts the computation of the motion constrain $S_{nD} = S_1 \cup S_2$ for a given point $\mathbf{u_{obst}}$ and target $\mathbf{u_{target}}$.

The set of desirable directions is the complementary of constrained directions $\mathcal{S}_D^\mathcal{G} = \{\mathbb{R}^3 \setminus \mathcal{S}_{nD}^\mathcal{G}\}$.

2) Motion computation: Once computed the motion constraints for each quadrant there are five cases depending on the relative location of the target $\mathbf{p_{target}}$ and the set of non desirable directions $\mathcal{S}_{nD}^\mathcal{G}$ in each quadrant.

1) **Case 1**: Target direction is not constrained $\mathbf{u_{targ}} \in \mathcal{S}_\mathbf{D}$:
$$\mathbf{u_{sol}} = \mathbf{u_{targ}}$$
2) **Case 2**: Target direction is constrained but only in the constrained directions of 1 quadrant (i.e. $\mathbf{u_{targ}} \in \mathcal{S}_{nD}^\mathcal{G}$ where $\mathcal{G} = [TL \mid DL \mid TR \mid DR]$). Then:
$$\mathbf{u_{sol}} = \mathbf{u_{dom}^\mathcal{G}}$$
3) **Case 3**: Target direction is constrained in the intersection of constrained directions of 2 quadrants (i.e. $\mathbf{u_{targ}} \in (\mathcal{S}_{nD}^{\mathcal{G}_1} \cap \mathcal{S}_{nD}^{\mathcal{G}_2})$ where $\mathcal{G}_1, \mathcal{G}_2 = [TL \mid DL \mid TR \mid DR]$):
$$\mathbf{u_{sol}} = \frac{\mathbf{u_{dom}^{\mathcal{G}_1}} + \mathbf{u_{dom}^{\mathcal{G}_2}}}{2}$$
4) **Case 4**: Target direction is constrained in the intersection of constrained directions of 3 quadrants (i.e. $\mathbf{u_{targ}} \in (\mathcal{S}_{nD}^{\mathcal{G}_1} \cap \mathcal{S}_{nD}^{\mathcal{G}_2} \cap \mathcal{S}_{nD}^{\mathcal{G}_3})$ where $\mathcal{G}_1, \mathcal{G}_2 = [TL, DR]$ and $\mathcal{G}_3 = [TR \mid DL]$ or $\mathcal{G}_1, \mathcal{G}_2 = [TR, DL]$ and $\mathcal{G}_3 = [TL \mid DR]$). Then:
$$\mathbf{u_{sol}} = \frac{\frac{\mathbf{u_{dom}^{\mathcal{G}_1}} + \mathbf{u_{dom}^{\mathcal{G}_2}}}{2} + \mathbf{u_{dom}^{\mathcal{G}_3}}}{2}$$
5) **Case 5**: Target direction is constrained in the intersection of constrained directions of the 4 quadrants (i.e. if $\mathbf{u_{targ}} \in (\mathcal{S}_{nD}^{TL} \cap \mathcal{S}_{nD}^{DL} \cap \mathcal{S}_{nD}^{TR} \cap \mathcal{S}_{nD}^{DR})$).
$$\mathbf{u}_{sol} = \mathbf{n}_\mathcal{E} \otimes \mathbf{n}_\mathcal{F}$$

where $\mathbf{u_{dom}^\mathcal{G}}$, $\mathbf{n}_\mathcal{E}$ and $\mathbf{n}_\mathcal{F}$ are computed as follows. Direction $\mathbf{u_{dom}^\mathcal{G}}$ is the best direction of motion in a given quadrant $\mathcal{G}$. Let be $\mathbf{p_i^\mathcal{G}}$ the points of the $\mathcal{S}_{bound}^\mathcal{G}$, then:

$$\mathbf{u_{dom}^\mathcal{G}} = \left(\begin{array}{ll} \arg\min_i(\frac{\mathbf{p_i^\mathcal{G}} \cdot \mathbf{e_x}}{||\mathbf{p_i^\mathcal{G}}|| \cdot ||\mathbf{e_x}||}) & \text{if } \mathcal{S}_D^\mathcal{G} = \emptyset \\ \arg\min_i(\frac{\mathbf{p_i^\mathcal{G}} \cdot \mathbf{p_{target}}}{||\mathbf{p_i^\mathcal{G}}|| \cdot ||\mathbf{p_{target}}||}) & \text{otherwise} \end{array} \right.$$

Directions $\mathbf{n}_\mathcal{E}$ and $\mathbf{n}_\mathcal{F}$ are:

$$\mathbf{n}_\mathcal{E} = (\mathbf{u_{dom}^{TL}} \otimes \mathbf{u_{dom}^{DR}}) \otimes \frac{\mathbf{u_{dom}^{TL}} + \mathbf{u_{dom}^{DR}}}{2}$$

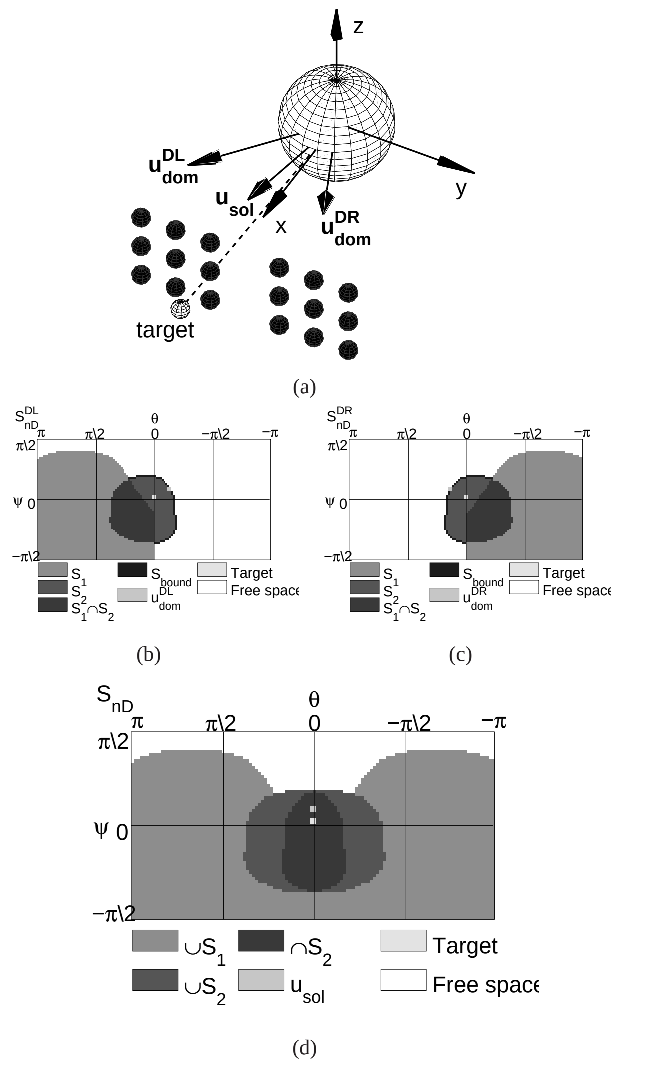
z
u^{DL}_{dom}
u_{sol}
x
u^{DR}_{dom}
y
target
(a)
S^{DL}_{nD}
θ
π $\pi\backslash 2$ 0 $-\pi\backslash 2$ $-\pi$
$\pi\backslash 2$
ψ 0
$-\pi\backslash 2$
S_1 S_2 $S_1 \cap S_2$ S_{bound} u^{DL}_{dom} Target Free space
(b)
S^{DR}_{nD}
S_1 S_2 $S_1 \cap S_2$ S_{bound} u^{DR}_{dom} Target Free space
(c)
S_{nD}
$\cup S_1$ $\cap S_2$ Target
$\cup S_2$ u_{sol} Free space
(d)

Fig. 4. Computation of the direction solution in Case 3. (a) Robot and obstacle configuration. (b,c) Representation of the motion constraints in latitude and longitude coordinates for quadrants DL and DR. (d) Representation in the same figure of both quadrants. Notice how the target direction is constrained in the intersection of constrained directions of both quadrants (Case 3). The motion solution in this case represented in (a).

$$\mathbf{n}_{\mathcal{F}} = (\mathbf{u^{TR}_{dom}} \otimes \mathbf{u^{DL}_{dom}}) \otimes \frac{\mathbf{u^{TR}_{dom}} + \mathbf{u^{DL}_{dom}}}{2}$$

As a result of this process we get a direction of motion solution $\mathbf{u_{sol}}$. Figure 4 shows an example of the motion computation in Case 3.

Finally the motion commands are translational $\mathbf{v}$ and rotational $\mathbf{w}$ velocities. The direction of $\mathbf{v}$ is the unitary vector $\mathbf{e_{sol}}$ in direction $\mathbf{u_{sol}}$. The magnitude is $||\mathbf{v}|| =$

$$= \begin{cases} v_{max} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}}) & \text{if } ||\mathbf{p_{obst}}|| > R + D_s, \\ v_{max} * \frac{||\mathbf{p_{obst}}|| - R}{D_s} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}}) & \text{if } ||\mathbf{p_{obst}}|| \le R + D_s. \end{cases}$$

where $\theta = \arccos(\frac{\mathbf{u_{sol}} \cdot \mathbf{e_x}}{||\mathbf{u_{sol}}|| \cdot ||\mathbf{e_x}||})$, D_s is a security margin around the robot bounds, v_{max} is the maximum velocity, and $||\mathbf{p_{obst}}||$ is the distance to the closest obstacle.

The rotational velocity is a rotation over $\mathbf{e_{sol}}$ with module:

$$||\mathbf{w}|| = \omega_{max} * \frac{\theta}{\frac{\pi}{2}}$$

where w_{max} is the maximum rotational velocity.

In summary in this Section we have presented the formulation of the ORM to work in three-dimensional workspaces. The next Section shows the experimental results.

III. Implementation and Experimental Validation

We tested the method in simulation with a spherical and omnidirectional robot equipped with a range laser sensor. The simulator emulates the motion of the vehicle and the sensor measurement process. We fixed the maximum velocities $v_{max} = 0.3\frac{m}{sec}$ and $w_{max} = 0.7\frac{rad}{sec}$ respectively. We assumed a frequency of 5Hz (perception–action cycle). All the scenarios where unknown and only the goal location was given in advance to the vehicle. The method used the information provided by the sensors to compute motion. No structure of the scenario is assumed and the scenario could be dynamic.

We describe next four experiments with three different objectives. The first one to show how the method correctly performs obstacle avoidance; the second is to confirm motion in dense, complex and confined scenarios; and the third is show how this method avoids classical shortcomings [7], [11] of other approaches like:

- To fall in trap situations due to the perceived environment structure (e.g U-shaped obstacles) or due to motion among very close obstacles.
- To generate unstable or oscillatory motion in constrained spaces.
- To have high goal insensitivity, i.e to be unable to choose directions far away from the goal direction.
- The impossibility to select motion directions towards obstacles.

These shortcomings usually lead to collisions or the impossibility to solve the navigation and thus to reach the goal. We describe next the four experiments:

a) Motion confined spaces: This experiment illustrates the method driving the vehicle in a dense and confined scenario (Figure 5a-c). The vehicle was introduced and moved along the pipe in order to reach the goal. Within the pipe there was little space to maneuver (the pipe and robot diameters are 0.6m and 1m respectively, i.e. 0.2m at both sides when centered). However, the method reactively managed to center the vehicle (Figure 5b). There were no trap situations due to motion among very closed obstacles, and the motion was smooth and oscillation free (see the robot's path and the velocity profiles in Figures 5a, c). The experiment was complete in 20sec, with an average translational velocity of $0.159\frac{m}{sec}$.

b) Motion in dense and complex scenarios: In this experiment the method drove the vehicle between two consecutive non aligned gaps. Firstly, the vehicle was driven to enter in the room by the first hole, and then was maneuvered to reach the second hole, pass it and reach the goal location (Figure 5d-f). In some parts of the experiment the robot navigated among closed obstacles where no traps were found. Stable and oscillation free motion was generated in all the experiment (see the robot path and the velocity profiles in Figures 5d, f). During almost all the experiment the method directed the motion towards obstacles (Figure 5e). Furthermore, the

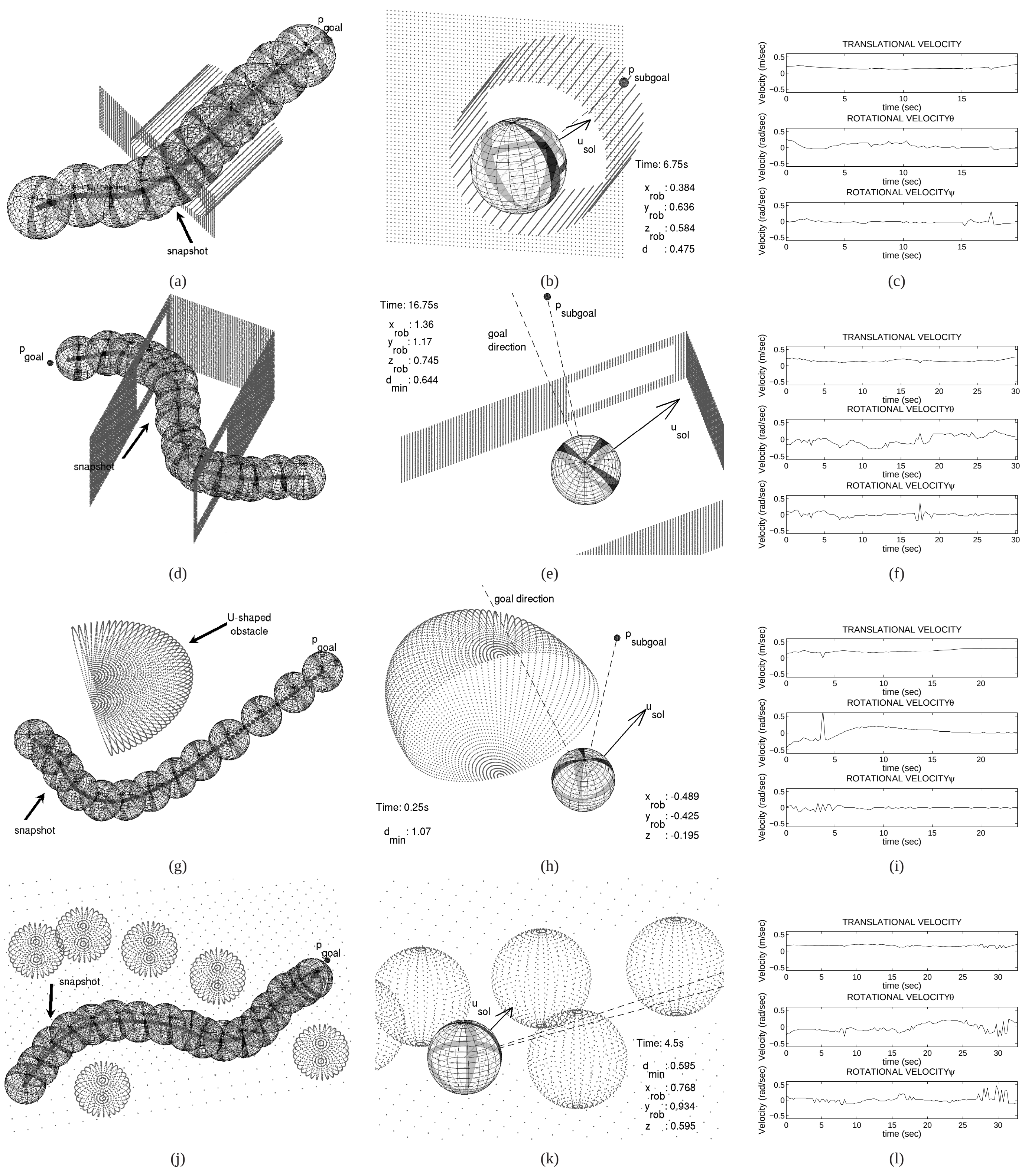

Fig. 5. These Figures show the experimental results. (Left column) Robot, Goal and path executed by the vehicle. (Center column) Representative snapshot of the experiment with the robot and the current perception. (Right column) Velocity profiles of the experiment.

method computed directions of motion far away from the goal direction (Figure 5e). The time of the experiment was 30sec and the average translational velocity was $0.153\frac{m}{sec}$.

c) Motion avoiding a U-shape obstacle: Here, the method avoided a large U-shape obstacle located between the vehicle and the goal (Figure 5g-i). The method avoided to enter and getting trapped inside the obstacle by directing the motion towards alternative subgoals located on the outside of the U-shaped obstacle (Figure 5h). In this case, motions directed far away from the goal direction had to be selected in order to avoid the U-shaped obstacle. The time of the experiment was 24sec and the average translational velocity was $0.223\frac{m}{sec}$.

d) Motion in a dense and cluttered environment: In this experiment, the robot navigated in a cluttered non structured scenario to reach the goal location. A *ceiling* and a *floor*, that are not displayed in the images for clarity, were added to force the robot to navigate among the obstacles (Figure 5j-l). No problems of traps or oscillations were found. In order to reach the goal the robot had to select motions directed toward the obstacles and as well, motions directed far away from the goal direction (Figure 5k). The experiment was carried out in 33sec, and the average translational velocity was $0.162\frac{m}{sec}$.

The experiments confirm that the method is able to perform robust obstacle avoidance in confined $3D$ scenarios. We discuss next advantages of this method regarding the classical problems or limitations of existing obstacle avoidance methods described earlier in this section.

The **local trap situations** due to U-shape obstacles or due to the motion among close obstacles are overcome with this method. The vehicle avoids entering and getting trapped within U-shape obstacles because the subgoal selector naturally place intermediate locations to avoid these regions (Figure 5h shows one of these situations). Furthermore, there is no difficulty to move among very close obstacles because: (i) the possibility of whether the vehicle fits in a passage among close obstacles is checked with the subgoal selector (in Figure 5e the $\mathbf{p_{subgoal}}$ is placed after checking that the vehicle fits in the hole), and (ii) the solution in Case 5 has been designed to center the vehicle in these circumstances. When moving among very close obstacles, we observed **oscillation free motion** (see the smooth paths generated and the velocity profiles in Figure 5). With this method, **directions far away from the goal direction** can be obtained when required. Given the current perception, the subgoal selector place a subgoal that is used in the sequent step as target location. This target can be located in any location in the space irrespective of the goal location. This property was determinant to successfully accomplish the experiments (Figures 5e,h). In addition, in the action formulation of the method nothing prohibits the **selection of directions of motion towards the obstacles**. Thus they were selected during all the experiments almost every time (Figures 5e,k). Another difficulty of many existing techniques is the **tuning of the internal parameters**. In the ORM formulation, the only parameter is a security margin D_s around the robot bounds (in our implementation is twice the robot radius).

IV. Conclusions

Many of the current applications in robotics deal with $3D$ confined scenarios like exploration in disaster arenas or cave inspection. Additionally, a great deal of work is been done for robots that operate in full $3D$ workspaces like outdoor, flying or underwater robots. This paper deals with one of the fundamental skills for robot autonomy that is the reactive obstacle avoidance.

The motivation to develop this work was the existing research gap related with obstacle avoidance methods that deal with $3D$ confined scenarios. The contribution is the formulation of an existing technique to work in $3D$ scenarios, its implementation and experimental validation. The difficulty of this work is twofold. First, the third dimension has to be constructed in such a way that the elimination of one dimension leads to the formulation proposed in the original $2D$ method. Second, the method has to have the same motion properties that the original method. On one hand, the first statement has been fully accomplished and one can see that the mathematical formulation proposed leads to the original method without one dimension. In other words, the formulation proposed is a valid generalization of the method.

On the other hand, we have demonstrated the second issue by developing and testing an implementation of the method. The experiments demonstrate that the method inherits the advantages of its predecessor, being able to avoid classical limitations of many existing obstacle avoidance methods such as local trap situations, instabilities or oscillations for example. The experiments confirm that the method correctly performs obstacle avoidance in dense, complex and confined $3D$ scenarios. This was the objective of this research.

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. 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.

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

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

[6] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *International Conference on Robotics and Automation*, Washington, USA, 2002.

[7] Y. Koren and J. Borenstein. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1398–1404, Sacramento, CA, 1991.

[8] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[9] 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.

[10] J. Minguez. The Obstacle Restriction Method (ORM): Obstacle Avoidance in Difficult Scenarios. In *IEEE Int. Conf. on Intelligent Robot and Systems*, Edmonton, Canada, 2005.

[11] 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.

[12] 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.

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

[14] A. Stenz. The focussed d^* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, Montreal, CA, 1995.