The Obstacle-Restriction Method (ORM) for Reactive Obstacle Avoidance in Difficult Scenarios in Three-Dimensional Workspaces

DANIEL VIKENMARK

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2006
The Obstacle-Restiction Method (ORM) for Reactive Obstacle Avoidance in Difficult Scenarios in Three-Dimensional Workspaces

DANIEL VIKENMARK

Master’s Thesis in Computer Science (20 credits)
at the School of Electrical Engineering
Royal Institute of Technology year 2006
Supervisor at CSC was Henrik Christensen
Examiner was Henrik Christensen

TRITA-CSC-E 2006:172
ISRN-KTH/CSC/E--06/172--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication
KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.csc.kth.se
Abstract

This master’s project addresses the obstacle avoidance problem in difficult scenarios in three-dimensional workspaces.

The main contribution of this project is the theoretical extension of the Obstacle-Restriction Method (ORM) in two dimensions to work in 3D workspaces.

This master’s thesis describes a reactive obstacle avoidance technique to drive a robot in dense, cluttered and complex environments in three-dimensional workspaces. The method has two steps: First a procedure computes instantaneous sub goals in the obstacle structure, second a motion restriction is associated with each obstacle which next are managed to compute the most promising direction of motion.

The advantage with the ORM, regarding the obstacle avoidance problem, is that it avoids the problems and limitations common in other obstacle avoidance methods, leading to improved navigation results in difficult scenarios. This is confirmed by simulations made in difficult scenarios in three-dimensional workspaces.

Sammanfattning

Hinderrestriktionsmetoden ORM för reaktiv hinderdetektion i komplicerade scenarion i tredimensionella arbetsmiljöer

Det här examensarbete behandlar hinderdetektionsproblemet i komplicerade scenarion i tredimensionella arbetsmiljöer.

Det största bidraget med det här examensarbetet är den teoretiska utökningen av metoden ORM i aspekten utav restrikterad hinderdetektion i tredimensionella arbetsmiljöer.


Fördeken med hinderrestriktionsmetoden, med avsikt på hinderdetektionsproblemet, är att denna metod löser de problem och begränsningar som andra hinderdetektionsmetoder har. Detta leder till förbättrade navigeringsresultat i svåra arbetsmiljöer. Dessutom löser den föreslagna metoden hinderdetektionsproblemet i tredimensionella arbetsmiljöer. Detta bekräfts av simuleringar utförda i komplicerade scenarion i tredimensionella arbetsmiljöer.
Acknowledgments

The Department of Signals Sensors and System (S3) and the Centre for Autonomous Systems (CAS) at the Royal Institute of Technology (KTH) in Stockholm have a STINT sponsored collaboration with the Robotics, Perception and Real Time Group (RoPeRT) at the University of Zaragoza, Spain. This master’s project was done within this collaboration at the University of Zaragoza, Spain, during the academic year 2004/2005.

First of all, I would like to thank my supervisor at RoPeRT, Ph.D Javier Minguez, with whom I have worked very close. Thank you for all the help with this project.

Also, I would like to thank my examiner at KTH Prof. Henrik I. Christensen and Prof. Bo Wahlberg for giving me the opportunity to do my master’s project in Zaragoza; Richard W Carlson and his daughter Hadley for proofreading the thesis; and finally, the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) for the scholarship I received for this project.
# Contents

1 Introduction ........................................... 1

2 Related Work ........................................... 3
   2.1 Potential Field Methods (PFM) ..................... 4
   2.2 Vector Field Histogram (VFH) ..................... 5
   2.3 Nearness Diagram Navigation (ND) .................... 6
   2.4 Obstacle-Restriction Method (ORM) .................. 8
   2.5 Why the ORM? ...................................... 9

3 The Obstacle-Restriction Method (ORM) in Three Dimensional Workspaces 11
   3.1 Problem Representation ............................. 11
   3.2 Sub Goal Selector .................................. 13
   3.3 Motion Computation ................................ 18
      3.3.1 Portions of the Space (subspaces) ............. 18
      3.3.2 The motion constraints ........................ 20
      3.3.3 Selecting the direction of motion .............. 25
   3.4 Computing the Robot Motion ........................ 33

4 Simulation Results ..................................... 35
   4.1 Simulation Settings ................................. 35
   4.2 Simulations ....................................... 35

5 Discussion ............................................. 41

6 Conclusions ............................................. 43

Bibliography ............................................. 45

Appendix A .............................................. 47
Chapter 1

Introduction

A large section of robotics is currently focused on the development of applications where a vehicle performs in unknown, unstructured and dynamic scenarios. An example is in an office where a robot moves among chairs, tables, shelves and doorways (which locations are unknown for the robot) and humans (who make the scenario highly dynamic). Another example is a submarine robot exploring an Underwater canyon (which is an unknown, unstructured and dynamic scenario). To drive a vehicle in a complex and evolving scenario, like the two examples, the vehicle needs a reactive system that can adapt the motion to any new contingency or event. The natural choice seems to be the obstacle avoidance methods which react on the sensory information within the control cycle. However, most of these methods run into difficulties in various scenarios. A classic drawback for the reactive approaches is the local minima situations due to motion between very close obstacles or due to U-shaped obstacles. This means that most of these methods cannot carry out robust or trustworthy navigation in complex evolving scenarios.

Recently two methods have been developed, the Nearness Diagram (ND) Navigation [16] and the Obstacle Restriction Method (ORM) [15], that overcome the difficult situations in obstacle avoidance. These methods are both able to successfully drive a vehicle in an unknown, unstructured and dynamic scenario.

Although there are some techniques that have been used in three dimensional workspaces, they inherit the limitations of the two dimensional case. Thus, if one wants to perform 3D obstacle avoidance in complex scenarios, an extension of either the Nearness Diagram or the Obstacle Restriction method would be a good choice. Unfortunately, there is no extension of these methods for 3D workspaces. As a result the motion problem in the second example (the underwater canyon) cannot be solved with today’s obstacle avoidance methods. This master’s project addresses this problem: Reactive obstacle avoidance in very dense, complex and cluttered three dimensional workspaces.

The design is an extension of an already existing method, the ORM, that exhibits good results when performing in two dimensions. When extending this method from two dimensions to three dimensions the existing advantages and disadvantages are
assumed to be inherited. Therefore the goal of the present work is to develop a method for 3D navigation with the same results as achieved in 2D. In particular the method should be able to overcome classical limitations of existing methods such as:

- avoid trap situations due to the perceived environmental structure, e.g. U-shaped obstacles and very close obstacles
- compute stable and oscillation free motion,
- exhibit a high goal insensitivity, i.e. to be able to choose directions of motion far away from the goal direction,
- be able to select motion directions towards obstacles.

This project strictly deals with the theoretical aspects of extending the method to three-dimensional workspaces. Thus, related issues such as shape, kinematics and dynamics of the vehicle and integration of the method are not within the frame of this work.

The outline of the thesis is as follows: Related work is presented in Chapter 2, the design of the extension of the ORM is presented in Chapter 3, the simulation results are presented in Chapter 4 which is followed by a discussion in Chapter 5, and in Chapter 6 I draw my conclusions.
Chapter 2

Related Work

This chapter presents the obstacle avoidance problem, a classification [13] of obstacle avoidance techniques, some representative methods and a discussion regarding the choice of the method.

In [13] the obstacle avoidance problem is formulated as:

The *obstacle avoidance problem* consists of moving a vehicle towards a target location free of collisions with the obstacles detected by the sensors during the motion execution. This involves the sensor information within the control cycle adapting to the motion of any contingency incompatible with initial plans.

Many different methods of addressing this problem have been proposed in the literature. Table 2.1 gives a brief overview of the classification and the representative methods.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Groups</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Heuristic</td>
<td>First Algorithms</td>
</tr>
<tr>
<td></td>
<td>Physical Analogies</td>
<td>PFM</td>
</tr>
<tr>
<td>2</td>
<td>Subsets of Commands</td>
<td>VHF</td>
</tr>
<tr>
<td></td>
<td>High Level Information</td>
<td>ND, ORM</td>
</tr>
</tbody>
</table>

*Table 2.1.* An overview of the two groups and the representative techniques discussed in this chapter.

Many of the existing obstacle avoidance techniques can be roughly grouped as follows:

- The *heuristic* methods were the first techniques to compute motion based on sensor information. The majority of these techniques derived later in classic planning methods and will not be discussed here.
CHAPTER 2. RELATED WORK

- The methods of physical analogies assimilate the obstacle avoidance to a known physics problem. One example, the Potential Field Method (PFM) [10] is discussed here.

- The methods of subsets of commands first compute an intermediate set of motion commands, and second, one of them is selected as solution. The Vector Field Histogram (VFH) [3] is discussed here.

- The methods that as intermediate information computes some kind of high level information, which in a next step is translated to a motion command. Here the Nearness Diagram Navigation (ND) [16] and the Obstacle Restriction Method (ORM) [15] is discussed.

2.1 Potential Field Methods (PFM)

The potential field method uses an analogy in which the robot is a particle that moves in the configuration space\(^1\) under the influence of a force field. The target location exerts a force that attracts the particle \(\mathbf{F}_{\text{att}}\) while the obstacles exerts a repulsive force \(\mathbf{F}_{\text{rep}}\). At each time \(t_i\) the motion is computed to follow the direction of the force induced by the sum of both potentials (figure 2.1):

\[
\mathbf{F}_{\text{tot}}(\mathbf{q}_i) = \mathbf{F}_{\text{att}}(\mathbf{q}_i) + \mathbf{F}_{\text{rep}}(\mathbf{q}_i)
\]

\(^1\)The configuration space is a transformation from the physical space in which the robot is of finite size into another space in which the robot is treated as a point. That is, the configuration space is obtained by shrinking the robot to a point, while growing the obstacles by the size of the robot.

Figure 2.1. (a) Shows the computation of the motion direction with a potential field method. The target attracts the particle while the obstacle exerts a repulsive force. The resulting force \(\mathbf{F}_{\text{tot}}\) is the most promising direction of motion. (b) Shows the motion direction computed in each point of the space with this method. (Figures taken from [13] by courtesy of J. Minguez).
2.2. VECTOR FIELD HISTOGRAM (VFH)

Figure 2.2. The computation of the direction of motion $\theta_{sol}$ with the VHF. (a) Robot and obstacle occupancy distribution. (b) The selected valley is the set of adjacent components with lower value than the threshold. The extreme sectors of the valley are $k_i$ and $k_j$. Here the solution sector is $k_{sol} = \frac{k_i + k_j}{2}$, whose bisector is $\theta_{sol}$. (Figures taken from [13] by courtesy of J. Minguez).

where $q_i$ is the vehicle’s current configuration. From $F_{tot}(q_i)$, which is the most promising direction of motion, the motion command is obtained.

Advantages / Disadvantages

+ The mathematical formalism behind the method and the simplicity.

– Produces potential minima that traps the robot due to motion among close obstacles or U-shaped obstacles [11].

– Produces oscillatory motion between two close obstacles or in narrow corridors.

2.2 Vector Field Histogram (VFH)

The Vector Field Histogram address the motion problem in two steps by computing an intermediate set of motion directions. First the space is divided in sectors from the robot’s location. A polar histogram $H$ is constructed around the robot where each component in the histogram represents the obstacle polar density in the corresponding sector.

Typically the resulting histogram has peaks (directions with high density of obstacles) and valleys (directions with low density). The set of candidate directions is the set of adjacent components with lower density than a specified threshold
CHAPTER 2. RELATED WORK

Figure 2.3. Shows the computation of the most promising direction of motion with the ND. To identify the situation some criteria has to be checked: First, there are no obstacle closer than the security distance $D_s$; second, the target is not within the motion area; third, the motion area is wide. With these criteria the current situation is identified. Here the motion command is $v_i = (v_{sol}, \theta_{sol})$ where $v_{sol}$ is the maximum velocity and $\theta_{sol}$ is computed as a deviation $\alpha$ from the limit direction $\theta_{disc}$ of the motion area. (Figure taken from [13] by courtesy of J. Minguez).

Advantages / Disadvantages

+ Has been formulated to work with probability obstacle distributions, thus it is well adapted to work with sensor readings corrupted by noise.

– Has difficulties navigating among close obstacles due to the tuning of an empirical threshold.

2.3 Nearness Diagram Navigation (ND)

The ND is a method that computes intermediate high level information that in a second step is converted to a motion command. This method simplifies the obstacle
2.3. NEARNESS DIAGRAM NAVIGATION (ND)

Figure 2.4. Illustrates the sub goal selector. (a) Robot, obstacle information perceived and six candidate sub goals $x_1...x_6$ (b) The tunnel to the goal location is blocked, thus there is no path within the tunnel. The C-obstacles are the obstacle points enlarged with the radius of the robot. (c) The tunnel to $x_6$ is also blocked, but the one to $x_1$ is not. Thus, there exists a path joining the current robot location and $x_1$ within the tunnel. In this situation $x_1$ is selected as sub goal. (Figures taken from [13] by courtesy of J. Minguez).

avoidance problem by employing a divide and conquer strategy based on situations. First, there is a set of situations that represent all cases among robot locations, target locations and obstacles. For each situation there is a motion law associated. During the execution phase at time $t_i$, one situation is identified, and the corresponding law is used to compute the motion.

The situations are represented in a binary decision tree where the criteria used is based on high level entities like a security distance around the robot and a motion area that identifies suitable regions of motion. For example one criteria is if there is an obstacle within the security zone, another is if the motion area is wide or narrow. The result is a unique situation since the sets of situations are complete and exclusive.

For every situation the motion is computed to adapt the behavior to the case represented by the situation. The motion is computed geometrically using the intermediate information of the dangerous obstacles and the motion area (Figure 2.3).

Advantages / Disadvantages

+ High-level description of the method.
+ It employs a divide and conquer strategy to simplify the difficulty of the navigation problem, thus it is able to address navigation in dense, complex and difficult scenarios.

− It does not use all the obstacle information in all the steps of the method, thus in some circumstances, the motion is less ”optimal” than expected.
2.4 Obstacle-Restriction Method (ORM)

The ORM is another method that computes intermediate high-level information that in a second step is converted to a motion command. This method uses all the obstacle information available in all the parts of the method. This is done in two steps, first a local procedure that computes sub goals is used to decide when (based on the structure of the surroundings) it is better to direct the motion toward an alternative computed sub goal rather then the goal location itself. If there is no local path to the goal, the closest sub goal to which there exists a collision-free path is selected to direct the motion (figure 2.4).

In the second step, a set of motion constraints is calculated for every obstacle. Strictly speaking, the set $S_{nD}$ of not desired directions of motion is computed as the union of two subsets, $S_1$ and $S_2$. $S_1$ represents the side of the obstacle that is not suitable to do the avoidance. From the robot each obstacle has two sides, $S_1$ contains all the directions of the opposite side of the target (figure 2.5(a)). $S_2$ is an exclusion area around the obstacle, which size is based on the distance to the obstacle $d_{obs}$ and the security distance $D_s$. The purpose of $S_2$ is to remove the obstacle from the robot’s security zone at the height of the obstacle (figure 2.5(b)).
2.5. WHY THE ORM?

The set $S_{nD} = S_1 \cup S_2$ (figure 2.5(c)). Joining all the sets by all the obstacles gives the full set of motion constraints $S_{nD}$.

Finally, depending on the topology of $S_{nD}$ and the target direction there are three cases (figure 2.5(d)–(f)) and in each of them one action law is executed computing the motion $\theta_{solution}$.

**Advantages / Disadvantages**

+ A strictly geometrical method.
+ Takes into account all the obstacle information in every part of the method.
+ Achieves safe navigation in dense, complex and difficult scenarios.

− ORM is a very young method that needs to be more mature to detect failures or misbehaviors.

2.5 Why the ORM?

The goal of this project is to design an obstacle avoidance method for motion in difficult scenarios in three-dimensional workspaces. The question that follows is: Why extend the Obstacle Restriction Method (ORM)?

There are two important things to consider:

1. The performance in two dimensions. If the method cannot drive a vehicle in difficult environments in two-dimensional workspaces it would not be able to drive it successfully in three-dimensional workspaces. The only two methods capable of driving a vehicle in difficult environments are the ORM and the ND. The performances of ORM and ND are very similar. They achieve the same result when navigating in dense, complex and difficult scenarios, however, preliminarily it seems that ORM achieves better results in open spaces [15].

2. The formulation of the ORM. It is a big advantage if the method is geometric since it seems possible to address an extension from two to three dimensions. On the other hand, the ND simplifies the obstacle avoidance problem by employing a divide and conquer strategy based on situations. When extending the ND, it seems more intricate to find all the situations formulated in 3D workspaces and the formulation of all the associated actions to each situation.
Chapter 3

The Obstacle-Restiction Method (ORM) in Three Dimensional Workspaces

This chapter presents the extension of the ORM to operate in 3D workspaces. The robot is assumed to be spherical and holonomic. The obstacle information is given in the form of mathematical points, which usually is the form in which sensor data is given (e.g. laser sensors).

This method like all other obstacle avoidance methods, is based on an iterative perception – action process. Sensors collect information from the environment, the information is processed by the method to compute a motion command, the motion is executed by the robot and the process restarts. For the extension of the ORM, this process has three steps. First a local selector of sub goals decide (based on the structure of the surroundings) if the motion should be directed toward the goal or if it is better to direct the motion toward another location in the space (subsection 3.2). Second, a motion constraint is associated with each obstacle. These constraints are managed next to compute the most promising direction of motion (subsection 3.3). Third, the motion command given to the robot is computed (subsection 3.4).

3.1 Problem Representation

In this section we describe mathematical representations that will be used in the remainder of the project.

We use spherical coordinates to represent the space. The unit sphere is divided in $m \times n$ sectors, $S_{m,n}$, centered at the origin of the robot frame (Figure 3.1(a)). We can see in Figure 3.1(b) how each sector is derived by spherical coordinates where $\theta$ is the azimuth angle with range $\theta \in (-\pi, \pi]$, $\psi$ is the elevation angle with range $\psi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $r$ is the distance to the obstacle point. The sector size is given by the angles $\Delta \theta$ and $\Delta \psi$.

In order to represent $S_{m,n}$ in a more compact way, we use a matrix $M_{m,n}$ (Figure
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.1. Space representation used by the ORM. (a) The space is divided in \( m \times n \) sectors, where the sector size is \([\Delta \theta, \Delta \psi]\) (in the implementation \( \Delta \theta = \Delta \psi = 3^\circ \) which gives \( m = 120 \) and \( n = 61 \) with a total of 7320 sectors). (b) Spherical coordinates \([\theta, \psi, r]\) are used. \( \theta \) range from \(-\pi\) to \(\pi\), \(\psi\) range from \(-\frac{\pi}{2}\) to \(\frac{\pi}{2}\) and \(r\) is the distance to the obstacle.

Figure 3.2. \(M_{m,n}\) is a more compact representation of \(S_{m,n}\). (a) \(M_{m,n}\) can be seen as \(S_{m,n}\) being cut open in the back and folded out. (b) The matrix \(M_{m,n}\) and its representation of the coordinate axis and the planes.

3.2. Each element \(m_{i,j}\) in \(M\) corresponds to the sector \(s_{i,j}\) in \(S\). The matrix element \(m_{i,j}\) contains the following information: The distance \(r\) to the obstacle \((r = m_{i,j})\), where \(m_{i,j} = \infty\) if the matrix element is empty, that is, if there is no obstacle in the direction of sector \(s_{i,j}\), the angle \(\theta\) \((i = 1, 2, ..., m\) corresponds to \(\theta = \pi, \pi - \Delta \theta, ..., -\pi\)\) and the angle \(\psi\) \((j = 1, 2, ..., n\) corresponds to \(\psi = \frac{\pi}{2}, \frac{\pi}{2} - \Delta \psi, ..., \frac{-\pi}{2}\)\).

A feature that will be used in the sub goal selecting procedure is contiguous sectors. If the matrix element \(m_{i,j} < \infty\) and if (at least) one of the neighbor elements \(m_{i\pm1,j\pm1} < \infty\) (see figure 3.3), then the obstacle points corresponding to the neighboring elements are called contiguous sectors.

Figure 3.4(c) shows an example of the representation matrix \(M\).
3.2. **Sub Goal Selector**

There are many situations where it is more suitable to direct the motion to a given area in the space rather than direct the motion toward the goal itself. For example, we can see in Figure (3.4) that it is better to direct the robot’s motion toward location $p_2$ (where the robot fits in the passage) and then (after the robot passed the passage) turn and direct the motion toward the goal location $p_{\text{goal}}$, rather than move directly toward the goal (where there is a wall blocking the way). In this section we present a procedure to decide if the motion should be directed toward the goal or if it is better to direct the motion toward an alternative computed sub goal. This procedure has two steps: First suitable locations in the space to place a sub goal have to be found, and second a sub goal is selected by computing if it can be reached from the current location.

In the first step locations to place the sub goals have to be found. The sub goals will be located:

1. In the middle point between two obstacle points, whose distance is greater than the robot’s diameter, when the corresponding sectors are contiguous, (e.g. location $p_1$ and $p_2$ in figure 3.5).

2. In the direction of the edge of the obstacle, at a distance greater than the robot’s diameter, when the corresponding sector does not have any contiguous sectors (e.g. location $p_3$, $p_4$ and $p_5$ in figure 3.5).

The result of the first step is a list of possible sub goals that capture the structure of the scenario.

The second step of the procedure is to decide whether the goal should be used to direct the motion, or if it is more suitable, to direct the motion toward one of the possible sub goals in the list. This is done by checking with an algorithm (described in the Appendix) whether the goal or a sub goal can be locally reached from the current location.

---

**Figure 3.3.** If an obstacle point is located in the direction of $m_{i,j}$ and there are obstacle points located in the direction of the sectors $m_{i\pm1,j\pm1}$, then these sectors are called *contiguous sectors* to $m_{i,j}$.
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.4. (a)–(b) The robot is facing a wall with two doorways, behind the first wall there is a second wall. The robot can easily pass through the right doorway while the left doorway is too narrow. In the figure the goal location and five of the candidate sub goals are shown. (c) The angular representation \( M_{m,n} \) of this scenario. Here we clearly see the wall with the two doorways. Through the doorways the second wall is seen. The black and the white contours are the possible locations to place sub goals. The distance to the obstacles is seen as a change of color.

The current location. This algorithm checks if there is a local path that connects the two locations or not. The algorithm returns:

**NEGATIVE:** The tunnel is blocked! There are two possibilities to block the tunnel, either there is an obstacle blocking the way (e.g. \( \mathbf{p}_{\text{goal}} \) in figure 3.6(c)) or the robot does not fit in a narrow passage (e.g. \( \mathbf{p}_1 \) in figure 3.6(d)).

**POSITIVE:** The tunnel is open! A local path in the tunnel joining the robot location and the sub goal exists (e.g. \( \mathbf{p}_2 \) in figure 3.6(e)).

In short, the algorithm checks if a *tunnel* (the cylinder in figure 3.6 and 3.7) in the configuration space is blocked by obstacles or not. If the tunnel is not blocked, one can demonstrate that there exists a non empty set of paths that join both
3.2. SUB GOAL SELECTOR

Figure 3.5. (a) Shows the same scenario as in figure 3.4, (b) shows the angular representation of this scenario. (c) The subfigure (b) is zoomed in at the right doorway to get a closer look at $p_2$. (d) The possible sub goal $p_2$ is placed in the direction of $m_{i,j}$ in between the obstacle points corresponding to the contiguous sectors $m_{i,j}$ and $m_{i,j-1}$. (e) The subfigure (b) is zoomed in at the top of the wall to get a closer look at $p_3$. (f) There are no contiguous sectors with a distance greater than the robot diameter at the edge of the obstacle. Therefore the possible sub goal $p_3$ is placed in the direction of $m_{i,j}$. 

15
Figure 3.6. Challenging case 1: Motion between very close obstacles. (a) The robot is facing a wall with two passages, the left passage is too narrow for the robot. (b) The wall with its passages in configuration space, the C-obstacles are the obstacle points enlarged with the radius of the robot. (c) The tunnel to the goal location is blocked by the C-obstacles. (d) The tunnel to \( p_1 \) is blocked, the C-obstacles are blocking the narrow passage to the left. (e) The tunnel is open and there is a path through the right passage leading to \( p_2 \).
3.2. SUB GOAL SELECTOR

Figure 3.7. Challenging case 2: A U-shaped obstacle. (a) The robot is facing a U-shaped obstacle where the goal is located behind the obstacle and an alternative computed sub goal is located to the right. (b) The U-shaped obstacle in configuration space. The C-obstacles are the obstacle points enlarged with the radius of the robot. (c) The tunnel to the goal location is blocked by the C-obstacles, there is no path within the tunnel. (d) The tunnel to \( p_1 \) is open, there is a path connecting \( p_1 \) with the robot location \( p_{\text{robot}} \). In this case the algorithm selects \( p_1 \) as sub goal, and avoids the possible trap situation created by the U-shaped obstacle.

locations within the tunnel. I.e. there is one collision free path between both locations. However, if the tunnel is blocked there is the guarantee that there is no path within the tunnel that joins both locations. On one hand, up to the locality of the algorithm, there is no collision free path between them. On the other hand there might exist a global path.

The interesting result is when the solution is positive, since it means that there is the guarantee that the sub goal can be reached. If the solution is negative, one may always choose another sub goal.

The sub goal is selected by first running the algorithm on the goal location. If the result is \textit{NEGATIVE} then we iteratively test with other sub goals (ordered by shorter distance to the goal) until the result is \textit{POSITIVE} for one of them. This
sub goal is selected for motion. For example, in figure (3.6) the algorithm returns \textit{NEGATIVE} when tried with \textbf{p}_{\text{goal}} and \textbf{p}_1, while the result is \textit{POSITIVE} when tried with \textbf{p}_2.

Many of the obstacle avoidance methods have situations that cause local minima (the trap situation mentioned in the introduction). Next we describe how the sub goal algorithm selects sub goals that direct the motion in such a way that these situations are avoided:

1. Motion between very close obstacles (figure 3.6). In the wall, there are two passages and thus, two different ways to reach the goal. The algorithm returns \textit{NEGATIVE} for \textbf{p}_1 because the robot does not fit in the narrow passage and thus the tunnel is blocked (figure 3.6(d)). But the algorithm returns \textit{POSITIVE} for \textbf{p}_2, which means that the robot fits in this passage (figure 3.6(e)). Notice that directing the motion towards \textbf{p}_2 avoids the trap situation or collision in the narrow passage.

2. The U-shaped obstacle (figure 3.7). The algorithm returns \textit{NEGATIVE} when it is used on the goal location \textbf{p}_{\text{goal}} (figure 3.7(c)). When it is used on \textbf{p}_1 (figure 3.7(d)) it returns \textit{POSITIVE} and \textbf{p}_1 is selected as alternative sub goal where to direct the motion of the robot. Notice that when choosing \textbf{p}_1 as sub goal, the robot avoids entering the U-shaped obstacle, and thus the trap situation is avoided.

Notice that this algorithm is used in combination with a procedure that locates the sub goals in the space to capture the structure of the scenario.

### 3.3 Motion Computation

The sub goal algorithm in the previous section selects a location in the space where to direct the motion of the robot. This location, the goal or a computed sub goal, is from now on referred to as the target. This section introduces the space division that will be used in the next subsections. Then, we discuss the motion computation that has two steps: First a set of motion constraints for each obstacle is computed, second, all sets are managed in order to compute the most promising direction of motion.

Let the frame of reference from now on be the robot frame, where the origin is \textbf{p}_0 = (0,0,0), the unitary vectors of the axes \((\textbf{e}_x, \textbf{e}_y, \textbf{e}_z)\) where \textbf{e}_x is aligned with the main direction of motion of the vehicle.

#### 3.3.1 Portions of the Space (subspaces)

Depending on the relation between the robot configuration and the target direction we divide the space in four quadrants as follows. Let \(\mathcal{A}\) be the plane defined by \([\textbf{p}_0, \textbf{e}_x, \textbf{e}_z]\), \(\mathcal{B}\) the plane \([\textbf{p}_0, \textbf{e}_z, \textbf{u}_{\text{target}}]\), and \(\mathcal{C}\) the plane \([\textbf{p}_0, \textbf{e}_y, \textbf{u}_{\text{target}}]\). Let \(\mathbf{n}_\mathcal{A}, \mathbf{n}_\mathcal{B}\)
3.3. MOTION COMPUTATION

Figure 3.8. Shows step by step how the space is divided into four subspaces. (a) Plane $A$ with normal $n_A$ divides the space into the left hand side and the right hand side of the robot. (b) Plane $B$ with normal $n_B$ divide the space into the left side and the right side of the target. (c) Plane $C$ with normal $n_C$ divide the space into the top side and the down side of the target. (d) Plane $A$ and plane $B$ creates together left side and right side. (e) Plane $C$ divide left side and right side into the four subspaces: top-left, down-left, top-right and down-right.
and \( \mathbf{n}_C \) be the normals to these planes respectively, computed by:

\[
\mathbf{n}_A = \mathbf{e}_y \\
\mathbf{n}_B = \mathbf{e}_z \otimes \mathbf{u}_{\text{targ}} \\
\mathbf{n}_C = \mathbf{u}_{\text{targ}} \otimes \mathbf{e}_y
\]  \(3.1\) \(3.2\) \(3.3\)

Then, let \( \mathbf{u} \) be a given vector, the quadrants are computed by:

\[
\begin{align*}
\text{TL} : & \quad (\mathbf{u} \cdot \mathbf{n}_A \geq 0) \land (\mathbf{u} \cdot \mathbf{n}_B \geq 0) \land (\mathbf{u} \cdot \mathbf{n}_C \geq 0) \\
\text{TR} : & \quad ((\mathbf{u} \cdot \mathbf{n}_A < 0) \lor (\mathbf{u} \cdot \mathbf{n}_B < 0)) \land (\mathbf{u} \cdot \mathbf{n}_C \geq 0) \\
\text{DL} : & \quad (\mathbf{u} \cdot \mathbf{n}_A \geq 0) \land (\mathbf{u} \cdot \mathbf{n}_B \geq 0) \land (\mathbf{u} \cdot \mathbf{n}_C < 0) \\
\text{DR} : & \quad ((\mathbf{u} \cdot \mathbf{n}_A < 0) \lor (\mathbf{u} \cdot \mathbf{n}_B < 0)) \land (\mathbf{u} \cdot \mathbf{n}_C < 0)
\end{align*}
\]  \(3.4\)

where TL, TR, DL and DR are Top-Left, Top-Right, Down-Left, Down-Right respectively. Figure 3.8 shows an example.

### 3.3.2 The motion constraints

This section describes the computation of a motion constraint for a given obstacle. A motion constraint includes a set of directions \( S_{nD} \in \mathbb{R}^3 \) that are not desired for motion. 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. The first subset of directions \( S_1 \) is created by the three planes described next.

Let \( \mathbf{u}_{\text{obst}} \) be the unitary vector in the direction of the obstacle point \( \mathbf{p}_{\text{obst}} \). Let \( \mathcal{D} \) be the plane defined by \( [\mathbf{p}_0, \mathbf{u}_{\text{obst}}, \mathbf{u}_{\text{targ}} \otimes \mathbf{u}_{\text{obst}}] \), where the normal \( \mathbf{n}_D \) is given by:

\[
\mathbf{n}_D = (\mathbf{u}_{\text{targ}} \otimes \mathbf{u}_{\text{obst}}) \otimes \mathbf{u}_{\text{obst}}
\]  \(3.5\)

We now define three sets of the space \( \mathcal{A}^+, \mathcal{B}^+ \) and \( \mathcal{D}^+ \). Given a vector \( \mathbf{v} \), we check if it belongs to any of these sets as follows:

\[
\begin{align*}
\mathbf{v} \in \mathcal{A}^+ : & \quad \mathbf{n}_A \cdot \mathbf{v} \geq 0 \land (\mathbf{p}_{\text{obst}} \in \text{TL} \lor \mathbf{p}_{\text{obst}} \in \text{DL}) \\
\mathbf{v} \in \mathcal{B}^+ : & \quad \mathbf{n}_A \cdot \mathbf{v} < 0 \land (\mathbf{p}_{\text{obst}} \in \text{TR} \lor \mathbf{p}_{\text{obst}} \in \text{DR}) \\
\mathbf{v} \in \mathcal{B}^+ : & \quad \mathbf{n}_B \cdot \mathbf{v} \geq 0 \land (\mathbf{p}_{\text{obst}} \in \text{TL} \lor \mathbf{p}_{\text{obst}} \in \text{DL}) \\
\mathbf{v} \in \mathcal{B}^+ : & \quad \mathbf{n}_B \cdot \mathbf{v} < 0 \land (\mathbf{p}_{\text{obst}} \in \text{TR} \lor \mathbf{p}_{\text{obst}} \in \text{DR}) \\
\mathbf{v} \in \mathcal{D}^+ : & \quad \mathbf{n}_D \cdot \mathbf{v} > 0
\end{align*}
\]  \(3.6\)

where the expression to compute if a direction belongs to a given subspace (TL, DL, TR, DR) is given by expression (3.4). Figure 3.9 and 3.10 shows examples.

Then the motion constraint \( S_1 \) is computed by:

\[
S_1 = \begin{cases} 
(\mathcal{A}^+ \cap \mathcal{B}^+) \cap \mathcal{D}^+ & \text{if } \mathbf{p}_{\text{obst}} \in \text{TL or } \mathbf{p}_{\text{obst}} \in \text{DL} \\
(\mathcal{A}^+ \cup \mathcal{B}^+) \cap \mathcal{D}^+ & \text{if } \mathbf{p}_{\text{obst}} \in \text{TR or } \mathbf{p}_{\text{obst}} \in \text{DR} 
\end{cases}
\]  \(3.7\)

where the expression (3.4) computes whether a point belongs to a given subspace. For example the first case is depicted in figure 3.9 and the second one in figure 3.10.
3.3. MOTION COMPUTATION

Figure 3.9. Shows step by step how $S_1$ is created. (a) The two planes $A$ and $B$ with normals $n_A$ and $n_B$ gives the two sets of directions, $A^+$ and $B^+$. *left side* is the intersection between these two sets. (b) Angular representation of *left side*, when computing the motion constraints the angular presentation shows which directions that are prohibited and which directions that are not. (c) The orientation of plane $D$ depends on the location of the target and the obstacle. The plane is orientated in such a way that set $D^+$ belongs to the side of the obstacle that is not desirable for avoidance. (d) The angular representation of set $D^+$. (e) *Left side* creates together with set $D^+$ the subset $S_1$ which contains the directions that is not desirable for motion. (f) Here we see $S_1$ which is the intersection between *left side* in (b) and set $D^+$ in (d).
Chapter 3. The Obstacle-Restriction Method (ORM) in Three Dimensional Workspaces

Figure 3.10. Shows step by step how $S_1$ is created. (a) The two planes $A$ and $B$ with normals $n_A$ and $n_B$ give the two sets of directions, $A^+$ and $B^+$, right side is the union of these two sets. (b) Angular representation of right side. (c) The orientation of plane $D$ depends on the location of the target and the obstacle. The plane is orientated in such a way that set $D^+$ belongs to the side of the obstacle that is not desirable for avoidance. (d) The angular representation of set $D^+$. (e) Right side creates together with set $D^+$ the subset $S_1$ which contains the directions that is not desirable for motion. (f) Here we see $S_1$ which is the intersection between right side in (b) and set $D^+$ in (d).
3.3. MOTION COMPUTATION

The second subset of directions \( S_2 \) is an exclusion area around the obstacle. Let \( R \) be the radius of the robot, \( D_s \) a security distance around the robot’s bounds and \( d_{\text{obst}} = \|p_{\text{obst}}\| \) the distance to the obstacle. We define \( S_2 \) as follows:

\[
S_2 = \{ u_i \mid \gamma_i \leq \gamma \}
\]  

(3.8)

where

\[
\gamma_i = \arccos \left( \frac{u_i \cdot u_{\text{obst}}}{\|u_i\| \cdot \|u_{\text{obst}}\|} \right)
\]  

(3.9)

and

\[
\gamma = \alpha + \beta \quad , \quad 0 < \gamma < \pi
\]  

(3.10)

where \( \alpha \) and \( \beta \) is given by,

\[
\alpha = |\arctan \left( \frac{R + D_s}{d_{\text{obst}}} \right)|
\]  

(3.11)

\[
\beta = \begin{cases} 
(\pi - \alpha) \left(1 - \frac{d_{\text{obst}} - R}{D_s}\right) & \text{if } d_{\text{obst}} \leq D_s + R, \\
0 & \text{otherwise}
\end{cases}
\]  

(3.12)

Figure 3.11 shows an example. The subset \( S_2 \) is a cone with the radius \( R + D_s \) at the height of the obstacle. The surface of the cone is the boundary to the subset \( S_2 \). Moving in any direction of this boundary will remove the obstacle from the robot’s security zone at the height of the obstacle.

The angle is \( \gamma = \alpha \) when the distance to the obstacle is greater than the security distance plus the robot’s radius (Figures 3.11(b)–(c)). When the obstacle is within the robot’s security zone a term \( \beta \) is added and \( \gamma = \alpha + \beta \) (Figure 3.11(d)–(e)). The angle \( \gamma \) ranges from 0 to \( \pi \) depending on the distance to the obstacle (\( \gamma = 0 \) where the \( d_{\text{obst}} \to \infty \) and thus there is no deviation, and \( \gamma = \pi \) when \( d_{\text{obst}} \to R \) and thus the exclusion area is maximized). Notice that in order to have continuity \( \beta = 0 \) when the distance to the obstacle is \( D_s + R \) (transition limit in expression (3.12)).

Finally, the motion constraint (\( S_{nD} \), set of not desirable directions of motion) for the obstacle is the union of the two subsets: \( S_{nD} = S_1 \cup S_2 \), Figure 3.12 shows an example.

A feature that will be used latter is the concept of the boundary \( S_{\text{bound}} \) to \( S_{nD} \). Let us denote \( S_{2-\text{bound}} \) the boundary to \( S_2 \) (i.e. all directions \( u_i \) such as \( \gamma_i = \gamma \)). We define \( S_{\text{bound}} \) as:

\[
S_{\text{bound}} = \{ u_i \in S_{nD} \mid u_i \in S_{2-\text{bound}} \text{ and } u_i \notin S_1 \}
\]  

(3.13)

In other words, \( S_{\text{bound}} \) is the part of the boundary to \( S_2 \) that does not belong to the subset \( S_1 \) (see Figure 3.12). When moving in a direction of this boundary, the robot avoids the obstacle on the desired side while, at the same time, selecting one of the shorter directions to reach the target.
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.11. Subset $S_2$. (a) $S_2$ is an exclusion area around the obstacle in form of a cone. The radius at the height of the obstacle is $R + D_s$, $\gamma$ is the angle between the center of the cone and the surface. (b) When the distance to the obstacle is $d_{\text{obst}} > R + D_s$ then $\gamma = \alpha$. All directions $u_i$ inside the cone satisfies $\gamma_i \leq \gamma$ and belongs to $S_2$. (c) Angular representation of the cone in figure b, the circular shape contains the directions that are not desirable for motion. (d) When the distance to the obstacle is $d_{\text{obst}} < R + D_s$ then $\gamma = \alpha + \beta$. Here $\gamma > \frac{\pi}{2}$ and the subset $S_2$ is an anti cone, all directions outside the cone in the figure belongs to $S_2$. (e) Angular representation of the anti cone in figure d. The circular shape contains the directions desirable for motion.
3.3. MOTION COMPUTATION

Figure 3.12. The motion constraints for one obstacle. (a) The union \( S_{n,D} \) of the two subsets \( S_1 \) and \( S_2 \) from figure 3.10(e) and 3.11(b). Here we see both the three planes that creates \( S_1 \) and the cone creating \( S_2 \). The surface of the cone that does not belong to the set \( S_1 \) makes the boundary \( S_{bound} \) to the set \( S_{n,D} \). (b) The angular representation gives a good image of the union of the two subsets. Here we see the intersection between the two subsets \( S_1 \) (from figure 3.10(f)) and \( S_2 \) (from figure and 3.11(c)). We also see the boundary \( S_{bound} \) as the border between the subset \( S_2 \) and the free space. (c) The union \( S_{n,D} \) of the two subsets \( S_1 \) and \( S_2 \) from figure 3.9(e) and 3.11(d). Here we see both the three planes that creates \( S_1 \) and the anti cone creating \( S_2 \). The surface of the anti cone makes the boundary \( S_{bound} \) to the set \( S_{n,D} \). (d) The angular representation shows the intersection between the two subsets and how the anti cone prohibits almost every direction of motion in the space.

3.3.3 Selecting the direction of motion

Previous we showed how to compute a set of directions that are not suitable for motion given an obstacle point. Next, we separate the motion constraints for each of the subspaces (TL, DL, TR, DR). The most promising direction of motion is computed given the motion constraints for each subspace and in a last step these directions are managed in order to select the final direction of motion. In this subsection we first describe how to manage the obstacles motion constraints, second
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

we show how to compute the most promising direction of motion for one subspace, and third how to compute the final direction of motion.

In order to compute the full set of motion constraints we first have to compute the motion constraints for each subspace, and then join them together. For each obstacle $p_i$ in subspace $G$, we compute the union as:

$$S_{G,i}^{nD} = S_i^1 \cup S_i^2$$  \hspace{1cm} (3.14)

then the set of motion constraints for subspace $G$ (e.g. the top-left subspace in figure 3.13) can be written:

$$S_{G}^{nD} = \bigcup_i S_{G,i}^{nD}$$  \hspace{1cm} (3.15)

Doing this for each obstacle group gives the four sets $S_{TL}^{nD}, S_{DL}^{nD}, S_{TR}^{nD}$ and $S_{DR}^{nD}$. By joining these four sets together (see example in figure 3.18(b)) we get the full set of motion constraints:

$$S_{nD} = S_{TL}^{nD} \cup S_{DL}^{nD} \cup S_{TR}^{nD} \cup S_{DR}^{nD}$$  \hspace{1cm} (3.16)

$S_{nD}$ contains the directions that are not desired for motion, thus the set of desired directions is:

$$S_D = \{ \mathbb{R}^3 \setminus S_{nD} \}$$  \hspace{1cm} (3.17)

We call $v$ a free direction when $v \in S_D$. Then we will say that there are free directions where $S_D \neq \emptyset$.

We next describe how to compute the most promising motion direction for each subspace. To compute the most promising direction of motion $u_{dom}^G$ for subspace with $S_{nD}^G$, we first have to compute the boundary (figure 3.13(b)–(e)). Let $S_{2-bound}^G$ be the boundary to $\bigcup_i S_i^1$ for all obstacles $p_i$ in obstacle group $G$, then the boundary $S_{bound}^G$ can be written:

$$S_{bound}^G = \{ u_i \in S_{nD} \mid u_i \in S_{2-bound}^G \text{ and } u_i \notin S_i^1 \}$$  \hspace{1cm} (3.18)

Moving in any direction along this boundary makes sure that all obstacles in $G$ will be kept at a distance equal to or greater then the robot’s security distance, at the same time, as one of the shorter directions to reach the target is selected. Therefore when we select $u_{dom}^G$ we choose between the directions in the boundary $S_{bound}^G$. There are two cases depending on the set of desired directions:

1. There are no free directions, $S_D = \emptyset$. $u_{dom}^G$ is the direction in the boundary which has the minimum angle to the robot’s current direction of motion, i.e. the x-axis. Let $u_i^G$ be all directions in $S_{bound}^G$ and let $\gamma_i$ be the angle between $u_i^G$ and $e_x$. Let $\gamma_{min} = \min(\gamma_i)$ and let $u_{min}^G$ be the direction vector with angle $\gamma_{min}$, then:

$$u_{dom}^G = u_{min}^G$$  \hspace{1cm} (3.19)

In this case the robot navigates among close obstacles. Therefore, as small changes as possible in the direction of motion is wanted.
3.3. MOTION COMPUTATION

Figure 3.13. Joining the sets. (a) A scenario where the robot is facing two obstacles that both belong to the top-left subspace and creates the top-left motion constraint. (b) This figure shows how the top-left subspace is computed in relation with the constraints for each obstacle. (c) The angular representation of the set of motion constraints created by the obstacle located to the left. (d) Angular representation of the set created by the obstacle that is located in above the robot. (e) This figure shows the top-left motion constraint $S_{\text{TL}}$ which is the union of the two sets in (c) and (d). The boundary $S_{\text{bound}}$ and the most promising direction of motion $u_{\text{dom}}$ to the set is seen in this figure. Notice that the target direction is a free direction, thus the direction of motion and the target direction is the same.
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.14. Case 2: (a) shows a scenario with obstacles in the top-right subspace. In (b) we see the top-right set of motion constraints which also is the full set since there are no other obstacles. The target direction belongs to this set and therefore the most promising direction of motion $u_{sol}$ is selected from the directions in the boundary $S_{bound}$. Free directions exist and thus the direction in the boundary closest to the target direction is selected.

2. There are free directions, $S_D \notin \emptyset$. $u_{dom}^G$ is the direction in the boundary which has the minimum angle to the direction of the target. Let $u_{G}^i$ be all directions in $S_{bound}^G$ and let $\gamma_i$ be the angle between $u_{G}^i$ and $u_{targ}$. Let $\gamma_{min} = min(\gamma_i)$ and let $u_{min}$ be the direction vector with angle $\gamma_{min}$, then:

$$u_{G \, dom}^G = u_{min}^G$$

(3.20)

In this case large changes in the direction of motion is allowed, since moving in any of the free directions increase the distance to the risky obstacles (the obstacles within the security distance).

As a result, we have a direction $u_{dom}^G$ for each of the four subspaces $G$.

In a last step the directions of motion $u_{dom}^G$ are managed in order to compute the final direction of motion $u_{sol}$. Figure 3.15 and Table 3.1 show the five cases used to compute this direction.

| Case 1 | $u_{sol} = u_{targ}$ |
| Case 2 | $u_{sol} = u_{dom}^G$ |
| Case 3 | $u_{sol} = \frac{u_{dom}^{G1} + u_{dom}^{G2}}{2}$ |
| Case 4 | $u_{sol} = \frac{u_{dom}^{G1} + u_{dom}^{G2} + u_{dom}^{G3}}{2}$ |
| Case 5 | $u_{sol} = n_{F} \otimes n_{F}$ |

Table 3.1. The five different cases, obtained from Figure 3.15, to compute the direction of motion.
3.3. MOTION COMPUTATION

The following describes each of the five cases outlined in Figure 3.15.

1. There exist a set of free directions, $\mathcal{S}_D \neq \emptyset$, and the target direction belongs to this set, $u_{\text{targ}} \in \mathcal{S}_D$. The target direction is selected as the final direction of motion:

$$ u_{\text{sol}} = u_{\text{targ}} \quad (3.21) $$

Figure 3.13 shows an example.

2. The target direction belongs to only one ($\mathcal{S}_{G_1}^D$) of the four sets of motion constraints. The most promising direction of motion to this set ($u_{\text{dom}}^{G_1}$) is selected as the final direction of motion:

$$ u_{\text{sol}} = u_{\text{dom}}^{G_1} \quad (3.22) $$

Figure 3.14 shows an example.

3. The target direction belongs simultaneously to the sets of non desirable directions of two subspaces ($\mathcal{S}_{G_1}^nD$ and $\mathcal{S}_{G_2}^nD$). The medium value of the most promising direction from each set is computed and selected as the final direction of motion:

$$ u_{\text{sol}} = u_{\text{dom}}^{G_1} + u_{\text{dom}}^{G_2} \quad (3.23) $$

Figure 3.16 shows an example.
Figure 3.16. Case 3: In (a) we have a scenario with obstacles in the the down-left and the down-right subspaces. There is a space between the two groups of obstacles where the robot does not fit, the target is located above and beyond this space. We also see the direction of motion from the two sets of motion constraints and the medium value between them \( u_{sol} \). The angular representation of the down-left set of motion constraints is seen in (c), here we see the boundary to the set and the direction of motion \( u_{dom}^{DL} \). (d) shows the same as in (c) but for the down-right motion constraints. In (b) we see the full set of motion constraints \( S_{nD} \) and the direction of motion \( u_{sol} \). Notice that the motion is directed higher then the target direction in order for the robot to avoid the obstacles.

4. The target direction belongs simultaneously to the sets of non desirable motion directions of three subspaces \( (S_{nD}^{G1}, S_{nD}^{G2} \) and \( S_{nD}^{G3}) \). First the medium value of the most promising direction from the two anti symmetric sets (the two anti symmetric sets are either TL–DR or TR–DL) is computed:

\[
\vec{u}_{dom}^{G1-G2} = \frac{\vec{u}_{dom}^{G1} + \vec{u}_{dom}^{G2}}{2}
\]  

Then, the medium value of this direction and the most promising direction from the remaining set is computed and selected as the final direction of
3.3. MOTION COMPUTATION

Figure 3.17. Case 4: In (a) we see a scenario were the robot is facing a L-shaped obstacle. Obstacle points are detected in the three subspaces: top-right, down-left and down-right. These three subspaces give rise to the three sets of motion constraints and the three direction of motion seen in (b)–(d). In (e) we see the union of the three motion constraints and the final direction of motion $u_{sol}$. In (a) we can see the two steps to compute $u_{sol}$: first the medium value between $u_{TR}^{dom}$ and $u_{DL}^{dom}$ is computed and gives $u_{TR-DL}^{dom}$, second the medium value between $u_{TR-DL}^{dom}$ and $u_{DR}^{dom}$ finally gives $u_{sol}$. Notice that $u_{sol}$ gives the robot a motion directed away from the L-shaped obstacle in order to avoid it.
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.18. Case 5: This is the most complex case to compute $u_{sol}$. We can see in (a) how the robot is facing a rectangular doorway. Obstacle points are detected in all four subspaces which gives rise to four sets of motion constraints and four directions of motion. In (b) we see the full set of motion constraints $S_{nD}$. Fig (c) shows how to create plane $E$: The medium value $u_{dom}^{TL-DR}$ is computed, also the plane spanned by $u_{TL}^{dom}$ and $u_{DR}^{dom}$ is created. Then the normal to this plane and the medium value $u_{sol}^{dom}$ spans the plane $E$ (plane $F$ is created in the same way). The two planes $E$ and $F$ are seen in (d), $u_{sol}$ is the intersection between these two planes. Notice how the motion direction is centered between the obstacle points.

motion:

$$u_{sol} = \frac{u_{TL}^{dom} - u_{DR}^{dom} + u_{sol}^{dom}}{2}$$  \hspace{1cm} (3.25)

Figure 3.17 shows an example.

5. The target direction belongs simultaneously to the sets of non desirable motion directions of four subspaces (example in figure 3.18). First the medium value between the diagonal sets are computed:

$$u_{dom}^{TL-DR} = \frac{u_{dom}^{TL} + u_{dom}^{DR}}{2}$$  \hspace{1cm} (3.26)
3.4. Computing the Robot Motion

\[ u_{DL}^{TR-DL} = \frac{u_{dom}^{TR} + u_{dom}^{DL}}{2} \]  

(3.27)

Let plane \( \mathcal{E} \) and \( \mathcal{F} \) be the planes defined by \([p_0, u_{dom}^{TR-DL}, u_{dom}^{TL} \otimes u_{dom}^{DR}] \) and \([p_0, u_{dom}^{TR-DL}, u_{dom}^{TR} \otimes u_{dom}^{DL}] \) respectively. The normals \( n_{\mathcal{E}} \) and \( n_{\mathcal{F}} \)
are:

\[ n_{\mathcal{E}} = (u_{dom}^{TL} \otimes u_{dom}^{DR}) \otimes u_{dom}^{TL-DR} \]  

(3.28)

\[ n_{\mathcal{F}} = (u_{dom}^{TR} \otimes u_{dom}^{DL}) \otimes u_{dom}^{TR-DL} \]  

(3.29)

Moving along the plane \( \mathcal{E} \) will make the robot centered between the two sets of motion constraints \( TL \) and \( DR \). In the same way the robot will be centered between \( TR \) and \( DL \) when moving along plane \( \mathcal{F} \). Therefore, moving along the intersection between these two planes will give the robot a direction of motion which keep the robot centered between the four sets of motion constraints. The direction along the intersection between the two planes is selected as the final motion of direction:

\[ u_{sol} = n_{\mathcal{E}} \otimes n_{\mathcal{F}} \]  

(3.30)

As a result we are able to obtain a promising direction of motion \( u_{sol} \) in every situation.

3.4 Computing the Robot Motion

In the previous section the most promising direction of motion \( u_{sol} \) was determined. This section will show how to compute the motion command that will be given to the robot. The motion command contains two velocities, the translational velocity \( v \) and the rotational velocity \( \omega \). Let us recall that the robot is assumed to be holonomic.

**Translational velocity \((v)\):** Let \( v_{max} \) be the maximum translational velocity, \( d_{obs} \) be the distance to the closest obstacle and let \( D_s \) be the robot’s security distance, then the module of translational velocity:

\[ v = \begin{cases} 
  v_{max} \times \left( \frac{\pi}{2} - |\theta| \right) & \text{if} \quad d_{obs} > D_s, \\
  v_{max} \times \frac{d_{obs}}{D_s} \times \left( \frac{\pi}{2} - |\theta| \right) & \text{if} \quad d_{obs} \leq D_s.
\end{cases} \]  

(3.31)

where \( \theta \) is the angle between the new direction of motion \( u_{sol} \) and the current robot direction of motion, i.e. the x-axis \( e_x \) (see figure 3.19):

\[ \theta = \arccos \left( \frac{u_{sol} \cdot e_x}{\|u_{sol}\| \cdot \|e_x\|} \right) \]  

(3.32)

From equation (3.31) we see that the translational velocity will be reduced if an obstacle shows up within the security distance. This velocity will also be reduced if there are large changes in the direction of motion, that is, when \( \theta \) is large.
CHAPTER 3. THE OBSTACLE-RESTRICTION METHOD (ORM) IN THREE DIMENSIONAL WORKSPACES

Figure 3.19. $\theta$ is the angle between the new direction of motion $u_{sol}$ and the current robot heading $e_x$.

Rotational velocity ($\omega$): Let $\omega_{max}$ be the maximum rotational velocity, then:

$$\omega = \omega_{max} \frac{\theta}{\pi}$$  \hspace{1cm} (3.33)

We see in equation (3.33) that the rotational velocity is reduced when there are small changes in the direction of motion, that is, when the angle $\theta$ is small.

The translational and rotational velocities are designed in such a way that the front of the robot always tries to be aligned with the direction of motion. When $\theta$ is big and the robot wants to make a large turn, the translational velocity is heavily reduced so the robot can turn and face the direction of motion as soon as possible. When $\theta$ is small the rotational velocity is reduced in order to make smooth turns. Notice that the angle $\theta$ will be forced, $\theta \in [0, \pi/2]$, in order to prohibit instantaneous backward motion. This makes the robot stop the translational motion when $\theta > \pi/2$ (backward motion), turn until $\theta < \pi/2$ (forward motion) and then start to increase the translational velocity again.

To summarize the chapter, we have presented a strictly geometry based method that, given information about the obstacle configurations and the goal location, computes the motion commands needed to avoid collision with obstacles while converging the robot location toward the target.
Chapter 4

Simulation Results

4.1 Simulation Settings

The extension of ORM was tested in Matlab where the simulation environments were created with points. This because usually the sensor information is given in this form (i.e. laser rangefinders) or other information can be expressed in this form. The robot was assumed to be spherical and holonomic. We added the constraint that instant backwards motion was prohibited. This is because usually robots and sensors have a main direction of motion. The radius of the robot was set to $0.3\,\text{m}$ and the security distance to $0.6\,\text{m}$. The maximum translational velocity was set to $0.3\,\text{m}/\text{sec}$ and the maximum rotational velocity to $0.7\,\text{rad}/\text{sec}$. The velocities, the security distance and the size of the robot were set to match the parameters in the Obstacle Restriction Method [15], where the application is human transportation. The sampling time was set to $250\,\text{ms}$ to match the frequency of the 3D laser scanners available today.

4.2 Simulations

The five simulations presented here were carried out in unknown and unstructured scenarios. Only the goal location was given in advance. The simulations were designed to verify that the extension of the ORM complies with the goal of this project: \textit{To safely drive a robot in dense, cluttered and complex scenarios}. Furthermore, these simulations will allow a discussion in Chapter 5 of the advantages of this method. The advantages are briefly summarized next:

- Avoiding trap situations due to the perceived environment structure, e.g. U-shaped obstacles and very close obstacles,
- computing stable and oscillation free motion,
- exhibiting a high goal insensitivity, i.e. to be able to choose a direction far away from the goal direction,
- selecting motion direction towards obstacles.
CHAPTER 4. SIMULATION RESULTS

Figure 4.1. Simulation 1. (a) Trajectory of vehicle and scenery. (b),(c) Snapshots of simulation. (d) Velocity profiles of the simulation.

Simulation 1: Motion in narrow spaces

In this simulation, the robot reached the goal location in a dense scenario with narrow places, where the space to maneuver was highly reduced (see the scenario and the robot’s trajectory in figure 4.1(a)). Here the robot managed to enter the narrow passage (the snapshot in figure 4.1(b)) and travel along it (the snapshot in figure 4.1(c)) with oscillation free motion, which is illustrated in the robot’s trajectory and the velocity profiles (the figure 4.1(d)). In the passage the robot moved among obstacles, distributed all around the robot, to which the distance to the robot bounds was about $0.2 \text{ m}$. The narrow passage had a radius of $0.5 \text{ m}$.

The simulation was carried out in 20 sec, and the average translational velocity was $0.159 \text{ m/sec}$.

Simulation 2: Motion in complex scenario

In this simulation the robot reached the goal location after navigating in a complex scenario (see the robot’s trajectory and the scenery in figure 4.2(a)). In some
parts of the simulation (when passing the doorways) the robot navigated among close obstacles (the snapshots in figure 4.2(b) and 4.2(d)). During almost all the simulation the method had to direct the motion toward obstacles (the snapshots in figure 4.2(b) and 4.2(c)) in order to converge to the goal location. The method
also selected directions of motions far away from the goal direction (the snapshot in figure 4.2(c)). The velocity profiles is presented in figure (4.2(e)).

The time of the simulation was 30 sec and the average translational velocity was 0.153 m/sec.

**Simulation 3: Motion avoiding a U-shape obstacle**

In this simulation, the robot had to avoid a U-shaped obstacle before it could reach the goal location (see the trajectory and the scenery in figure 4.3(a)). The method avoided entering and getting trapped inside the obstacle by directing the motion toward alternative sub goals that were located on the outside of the U-shaped obstacle (the snapshot in figure 4.3(b)). Motions directed far away from the goal direction had to be selected (the snapshot in figure 4.3(c)) in order to avoid the U-shaped obstacle. The velocity profiles are presented in figure (4.3(d)).

The time of the simulation was 24 sec, and the average translational velocity was 0.223 m/sec.
4.2. SIMULATIONS

In this simulation the robot had to navigate in a cluttered scenario to reach the goal location. A ceiling and a floor were added to force the robot to navigate in...
between the obstacles (see the scenery and the robot trajectory in figure 4.4(a) and 4.4(b)). In order to reach the goal the robot had to select motions directed toward the obstacles (the snapshots in figure 4.4(c) and 4.4(d)) as well as motions directed far away from the goal direction (the snapshot in figure 4.4(d)). The velocity profiles is presented in figure (4.4(e)).

The simulation was carried out in 33 sec, and the average translational velocity was $0.162 \frac{m}{sec}$. 
Chapter 5

Discussion

We discuss in this section the advantages of this method with regard to existing techniques, on the basis of the difficult scenarios shown in the simulation results. Since there is little evidence of obstacle avoidance methods performing in three dimensional workspaces, we assume here that the 3D methods would inherit the properties of their formulation in 2D.

The local trap situations due to U-shaped obstacles or due to motion among close obstacles are overcome with this method. First, the extension of ORM does not direct the motion within U-shaped obstacles since the sub goal selector places the sub goals outside of these obstacles. However, sometimes while the obstacle is not fully perceived with the sensors, the motion could be directed within an U-shaped obstacle (this is an intrinsic limitation of obstacle avoidance methods). Second, the method drives the robot among close obstacles because: (i) the sub goal selector first checks whether the robot fits in the narrow passage; (ii) the motion computed centers the robot among obstacles. This is because, among close obstacles, the motion is obtained as the intersection of the two planes (see Case 5 in section 3.3.3) which centers the motion between the four subspaces. In both cases, the Potential Field Methods [10] produce local minima that trap the robot [11]. Also the methods based on polar histogram [3], [20], [21] have difficulties navigating among close obstacles due to the tuning of an empirical threshold. Traps due to U-shaped obstacle are not avoided by the methods that use constrained optimizations [1], [7], [18], [6]. This occurs because the optimization loses the information of the environment structure that is necessary to solve this situation. The methods based on a given path deformed in real time [9], [17], [5], [4] are trapped when the path lies within dynamically created U-shaped obstacles.

The extension of ORM has an oscillation free and stable motion when moving among close obstacles. This is because information from all obstacles are used to compute the motion, and thus, the difference of the sensor information from two following time cycles is very small. Here the potential field methods can produce oscillatory motion when moving among close obstacles or in narrow corridors [11].

Motion directions far away from the goal direction is obtained with the
extension of ORM. This is because the sub goals can be placed in any location of the
space, and any direction of the space can be obtained as a solution. The obstacle
avoidance methods that makes a physical analogy (e.g. [8], [2], [10], [12], [14] and
[19] ) uses the goal location directly in the motion heuristic. The methods that
use constrained optimizations [6], [18], [7], [1] use the goal direction as one of the
balance terms. Therefore, these methods have high goal sensitivity.

The selection of motions towards the obstacles is obtained by the extension
of ORM. Some methods explicitly prohibit the selection of motion towards obstacles
(e.g. [20]).

One difficulty found in almost every obstacle avoidance method is the tuning
of internal parameters. The extension of ORM has no internal parameters, only
the security distance has to be set to a coherent value (in our implementation we
set this distance to twice the robot radius).

The presented method overcomes many of the problems and limitations found
in obstacle avoidance methods performing in two dimensional workspaces, which
lead to very good results when driving a vehicle in difficult scenarios. This was the
objective of this experimental validation.
Chapter 6

Conclusions

This master’s project addresses obstacle navigation in three dimensional workspaces. We have presented the design of an obstacle avoidance method that successfully drives a robot in dense, cluttered and complex environments in three dimensional workspaces. The design is an extension of an existing method [15] performing in two dimensions with outstanding results. The proposed method has two steps: First a procedure computes instantaneous sub goals in the obstacle structure; second a motion restriction is associated with each obstacle which next is managed to compute the most promising direction of motion.

The advantage with this method is inherited from its precursor: It avoids the problems and limitations that are common in other obstacle avoidance methods, which leads to very good navigation performance in difficult scenarios.

The disadvantage with this and all the obstacle avoidance methods is that they are local techniques used to address the motion problem, and thus, the global trap situations persist (in these cases the robot might never converge to the goal location). This problem is solved by integrating the obstacle avoidance method with a global motion planning method, but this issue is far beyond the scope of this project.

Obstacle avoidance in three dimensions is a very new research area. The main contribution of this project is the theoretical aspects of obstacle avoidance in difficult three dimensional workspaces.
Bibliography


Appendix A

Here we describe the algorithm that checks whether there is a path connecting two points in the space (the robot location and an alternative sub goal).

Let \( p_a \) and \( p_b \) be two locations in space, \( R \) the radius of the robot and \( L \) a list of points where \( p^L_p \) is an obstacle. The algorithm is as follows:

1. Let \( L' \) be the list of points of \( L \) that belongs to the cylinder \( S \) with the height \( p_a p_b \) and the diameter \( 2R \) (figure 6.1(a) and (b)).

2. Let \( C \) be a list of clusters. The cluster \( C_i \) is a list of points of \( L' \) where for all points in \( C_i \) the distance \( d(p^L_j, p^L_k) < 2R \) (figure 6.1(b)).

3. Let \( A \) be the plane perpendicular to \( p_a p_b \) (figure 6.1(b)).

4. Let \( x_c \) be the intersection between \( p_a p_b \) and \( A \) (figure 6.2).

5. Let \( T \) be the intersection between the cylinder \( S \) and the plane \( A \). \( T \) is a circle with \( x_c \) in the center and the diameter \( 2R \) (figure 6.2).

6. **Repeat step 6–9 for each cluster.** Let \( x^L_i \) be the points \( p^L_i \) projected to the plane \( A \) and let the points be angular contiguous (figure 6.2).

7. Create a polygon by connecting the points, \( \text{line}(x^L_i, x^L_{i+1}) \) where \( i = 1, ..., N \). If \( x_c \) belongs to the polygon then: \( \text{GOTO 8} \) (figure 6.2(b) and (d)), else cluster \( C_i \) returns: \( \text{POSITIVE} \) (figure 6.2(a)).

8. Let \( T^L_i \) be the circles created with the points \( x^L_i \) in the center and with the diameter \( 2R \) (figure 6.2(c) and (e)).

9. Let \( (u^L_i, v^L_i) \) be the intersections points of the circles \( (T^L_i, T^L_{i+1}) \). If for any intersection pair \( (u^L_i \& v^L_i) \) in \( T \) then cluster \( C_i \) returns: \( \text{POSITIVE} \) (figure 6.2(e)) else cluster \( C_i \) returns: \( \text{NEGATIVE} \) (figure 6.2(e)).

10. If any cluster returns \( \text{NEGATIVE} \) then the algorithm returns \( \text{NEGATIVE} \), else the algorithm returns \( \text{POSITIVE} \)

\(^1\)The \( \text{mod}(\ast, N) \) function is used to give continuity, if \( i = N \) then \( i + 1 = 1 \). This function is also used to compute the intersections of the circles in step 9.
If the algorithm returns:

*POSITIVE*: there is a local path that connects $p_a$ and $p_b$, i.e. the final location can be reached from the current location.

*NEGATIVE*: the final location can not be reached within a local area of the space (the cylinder with radius $2R$ and height the segment that joins the two locations). This means that it does not exist a path in this local area of the space, although there could exist a global one.
Figure 6.1. (a) The obstacles belongs to the list $L'$, that is, they belong to the cylinder in the next figure. (b) shows the configuration space of (a). Here we see the cylinder $S$, the clusters $C = [C_1; C_2; C_3]$ and the plane $A$ perpendicular to $P_a P_b$. 
Figure 6.2. In this figure we see the circle $T$ with its center point $x_c$, we also see the projection $x^L_1$ of the obstacle points. In (a) $x_c$ is not within the polygon created by the points, thus this cluster does not block the cylinder. In (b) and in (d) the center of the circle $x_c$ is located within the polygon, these two clusters may block the cylinder. In (c) we see the intersections of the circles $T_i$. Both intersections (the arrows) between circle $T_1$ and $T_3$ is located within the circle $T$, thus this cluster does not block the cylinder (the cylinder is open where we can see the right arrowhead). In (d) we see the intersections of the circles $T_i$. One intersection between circle $T_i$ and $T_{i+1}$ is located outside of the circle $T$ (the arrows) for all intersections, thus this cluster blocks the cylinder.