Pedro Miguel de Barros Gomes

LADAR Based Mapping and Obstacle Detection System for Service Robots

Lisboa

2010

UNIVERSIDADE NOVA DE LISBOA

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Sistema de Mapeamento e Detecção de Obstáculos Baseado em LADAR para Robôs de Serviço

Pedro Miguel de Barros Gomes

Dissertação apresentada na Faculdade de Ciências e Tecnologia
da Universidade Nova de Lisboa para a obtenção do grau de
Mestre em Engenharia Electrotécnica e de Computadores

Orientador: Prof. Doutor Pedro Alexandre da Costa Sousa

Lisboa

2010

NEW UNIVERSITY OF LISBON

Faculty of Sciences and Technology

Electrical Engineering Department

LADAR Based Mapping and Obstacle Detection System for Service Robots

Pedro Miguel de Barros Gomes

Dissertation presented at Faculty of Sciences and Technology of the New University of Lisbon to attain the Master degree in Electrical and Computer Science Engineering.

Supervisor: Prof. Doutor Pedro Alexandre da Costa Sousa

Lisbon

2010

Acknowledgements

First of all I would like to express my gratitude to my dissertation supervisor, Prof. Pedro Sousa, for giving me this opportunity and also for all the motivation, availability and advises. I would also like to thank the Portuguese SME company Holos, S.A.[1] for supporting the development of this dissertation and for providing the necessary resources.

A very special thanks to my colleagues Rúben Lino and Tiago Ferreira for all the support, precious help, valuable comments and for all the interesting discussions about this dissertation and other issues. I would also like to thank João Lisboa for all the relevant comments and support in some stages of this dissertation. An additional acknowledge to all the employees of Holos for the good atmosphere at work.

I would also like to express some words of thanks to all friends, colleagues and teachers who contributed to my professional and personal growth.

Last but not least, I would like to express my deep and sincere gratitude to my parents João and Lídia and my girlfriend Catarina. They have always been there in the bad moments giving me strength to never give up. They are the ones to whom I owe more and nothing I can do will ever thank them for their love, care, friendship and unconditional support.

[1] www.holos.pt

To my parents and my girlfriend

Resumo

Ao percorrer ambientes desconhecidos, um robô de serviço móvel precisa de adquirir informação sobre o ambiente que o rodeia, para poder detectar e evitar obstáculos e chegar com segurança ao seu destino.

Esta dissertação apresenta uma solução para o problema de mapeamento e detecção de obstáculos em ambientes estruturados[2] interiores ou exteriores, com particular aplicação em robôs de serviço equipados com um LADAR. Esta solução foi desenhada apenas para ambientes estruturados e, como tal, ambientes todo-o-terreno estão fora do âmbito deste trabalho. A utilização de qualquer conhecimento, obtido *a priori*, sobre o que rodeia o LADAR também está descartada, ou seja, o sistema de mapeamento e detecção de obstáculos desenvolvido trabalha em ambientes desconhecidos.

Nesta solução, assume-se que o robô, que transporta o LADAR e o sistema de mapeamento e detecção de obstáculos, está posicionado sobre uma superfície plana, que é considerada como sendo o plano do chão. O LADAR é posicionado de uma forma apropriada para um mundo tridimensional e é utilizado um sensor AHRS para aumentar a robustez do sistema em relação a variações na orientação do robô que, por sua vez, podem originar falsos positivos na detecção de obstáculos.

Os resultados dos testes efectuados em ambientes reais, através da incorporação deste sistema num robô físico, sugerem que o sistema desenvolvido pode ser uma boa opção para robôs de serviço que operem em ambientes estruturados interiores ou exteriores.

Palavras-Chave: detecção de obstáculos, robôs de serviço, robôs móveis, LADAR, mapeamento, interior, exterior, AHRS.

[2] Nesta dissertação, a expressão “ambientes estruturados” refere-se a ambientes em que a superfície do chão deverá ser plana.

Abstract

When travelling in unfamiliar environments, a mobile service robot needs to acquire information about his surroundings in order to detect and avoid obstacles and arrive safely at his destination.

This dissertation presents a solution for the problem of mapping and obstacle detection in indoor/outdoor structured[3] environments, with particular application on service robots equipped with a LADAR. Since this system was designed for structured environments, off-road terrains are outside the scope of this work. Also, the use of any a priori knowledge about LADAR's surroundings is discarded, i.e. the developed mapping and obstacle detection system works in unknown environments.

In this solution, it is assumed that the robot, which carries the LADAR and the mapping and obstacle detection system, is based on a planar surface which is considered to be the ground plane. The LADAR is positioned in a way suitable for a three dimensional world and an AHRS sensor is used to increase the robustness of the system to variations on robot's attitude, which, in turn, can cause false positives on obstacle detection.

The results from the experimental tests conducted in real environments through the incorporation on a physical robot suggest that the developed solution can be a good option for service robots driving in indoor/outdoor structured environments.

Keywords: obstacle detection, service robots, mobile robots, LADAR, mapping, indoor, outdoor, AHRS.

[3] In this dissertation, the term "structured environments" refers to environments in which the ground surface is expected to be planar.

Symbols and Notations

Symbol	Description
AHRS	Attitude and Heading Reference System
ahrsDriv	A *Player driver* that interacts with an AHRS sensor
ContDriv	A *Player driver* that communicates with a controller
IFR	International Federation of Robotics
IMU	Inertial Measurement Unit
IP	Internet Protocol
LADAR	LAser Detection And Ranging or Laser Radar
ladarDriv	A *Player driver* that communicates with a LADAR and acquires its measures
LIDAR	LIght Detection And Ranging
LTP	Local Tangent Plane
MapOD	A *Player driver* responsible for the mapping and obstacle detection system
NED	North-East-Down
RPY	Roll-Pitch-Yaw
SLAM	Simultaneous Localization And Mapping
SME	Small Medium Enterprise
TCP	Transmission Control Protocol
x_{ref_σ}	X coordinate of a *reference plane* point at angle σ
x_{int}	X coordinate of a point that intersects the XY plane
x_t	X coordinate of a transformed point
y_{ref_σ}	Y coordinate of a *reference plane* point at angle σ
y_{int}	Y coordinate of a point that intersects the XY plane
y_t	Y coordinate of a transformed point
z_t	Z coordinate of a transformed point
Avg_h	The average of the heights of all computed points that correspond to each cell
b	Y-intercept of a line segment
c_h	Height of a cell of the *elevation map*, measured in meters
Ch_{obs}	Height of a cell of the *obstacle map*, measured in meters
c_v	The value of an *obstacle map*'s cell
c_w	Width of a cell of the *elevation map*, measured in meters

Cw_{obs}	Width of a cell of the *obstacle map*, measured in meters
d	The theoretical range provided by the LADAR's central beam
d_1	First measured distance
d_n	Nth measured distance
d_{xy}	Projection of distance d onto the XY plane
h	LADAR's height
H_{neg}	Minimum height that an object that stands below the ground plane must have to be considered as an obstacle
H_{pos}	Minimum height that an object that stands above the ground plane must have to be considered as an obstacle
H_σ	Height of a hypothetical obstacle at each angle σ
L	LADAR's position on the map
m	Number of width cells of the *elevation map*
m_{obs}	Number of width cells of the *obstacle map*
m_{zx}	Slope of a line segment from ZX plane
m_{zy}	Slope of a line segment from ZY plane
n	Number of height cells of the *elevation map*
n_{obs}	Number of height cells of the *obstacle map*
P_σ	A 3D point at angle σ
$R_{real(\sigma)}$	Real range provided by the LADAR at each angle σ
$R_{xy\sigma}$	Projection of distance R_σ onto the XY plane
R_σ	LADAR's theoretical range at each angle σ
α	Angular step of the LADAR
ε	A threshold for *pitch* angle, in degrees
η	A threshold for *pitch* angle, in degrees
θ	Field of view of the LADAR
ρ	A generic angle, in degrees
σ	Represents each angle where the LADAR emits a beam
ϕ	An angle obtained from LADAR's tilt ($\phi = 90^\circ$ - tilt)
ω	A generic angle, in degrees
$new_{dist(\sigma)}$	A new distance for the *reference plane* at angle σ

Contents

List of Figures

1. Introduction

Nowadays, robots perform important roles in our society. They are present on many different areas such as automotive industry, entertainment, military operations, among others. In particular, mobile service robots employment has been widespread in the last years and today these kind of robots can perform several tasks like, for example, guiding tourists in museums [Chella et al., 2007], house cleaning, elderly care [Graf et al., 2004], interplanetary exploration [Cheng et al., 2005], or humanitarian demining [Santana et al., 2007].

The International Federation of Robotics (IFR) defines a service robot as *"(…) a robot which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations*" [IFR, 2009]. Through this definition one can notice that autonomy is a key feature for a service robot. Therefore, and mostly on mobile service robots area, mapping is generally considered as one of the most important topics [Thrun, 2003]. In fact, once exploring an unknown area, the mobile service robot needs a process to construct a representation of the environment (map) and this can be used for navigation tasks such as path planning or obstacle detection and avoidance. Despite the fact that obstacle detection is often performed without using maps, it is also common to find approaches where mapping and obstacle detection are interconnected, which is the case in this dissertation. In this thesis, mapping is performed with the purpose to detect obstacles, i.e. obstacle detection will be accomplished with basis on the construction of terrain maps.

In order to build a map, a robot must be equipped with sensors that enable it to perceive its surroundings such as LADARs, Stereoscopic Cameras or Ultrasonic Range Sensors. Also, there are additional sensors that are often used in mapping including Inertial Measurement Units (IMU), Attitude and Heading Reference Systems (AHRS) or Global Positioning Systems (GPS).

Ultrasonic Range Sensors are very popular and they are present in many mobile robots nowadays [Lee et al., 2009]. They have good characteristics including low power consumption and low cost. However, comparing to other range sensors they exhibit low angular resolution.

LADARs and Stereoscopic Cameras are complementary and are often used simultaneously [Moghadam et al., 2008], [Matthies et al., 2002] and they both have advantages and disadvantages. A LADAR generally provides fast and accurate range measurements and works with big fields of view and at long ranges. However, it only provides information along the plane of the scanning laser beam and its performance is affected by weather conditions (such as fog or rain) and by objects reflectivity. On the other hand, Stereoscopic Cameras can provide 3D data and also color and texture information. Nevertheless, it generates large amounts of data (including noisy data) comparing to a LADAR, which can be computationally expensive to process and its measurements are affected by lightning conditions and by non-textured environments.

As mentioned before, the fusion of these two types of sensors is a common attempt to overcome their disadvantages and to take part of their main capabilities in order to build precise and reliable maps. However, in the context of this work a LADAR will be used as the only sensor for environment perception, mainly due to its accuracy and lower amount of generated data comparing to stereo vision. Also, the fusion of two or more range sensors will not be focused in the context of this thesis and, therefore, integration of stereo vision cameras and LADARs is discarded.

In the 1980s and early 1990s, two approaches for mapping were used: metric and topological [Thrun, 2003]. Metric maps can represent geometric features of the environment and two examples of this kind of maps are occupancy grid maps [Elfes, 1987] and feature maps [Chatila and Laumond, 1985]. In the former case, environments are represented by an occupancy grid where each cell can indicate the presence of an obstacle and in the latter case maps contain parametric features such as lines or arcs that intend to describe the environment. Topological maps can represent environments through connectivity between different places and an early example of this approach is the work of Kluipers and Byun [Kuipers and Byun, 1991]. In this case, the map contains a set of significant places that can be connected by arcs. These arcs are usually labeled with information about navigation from one place to another.

Since the 1990s, mapping has generally been named as SLAM (simultaneous localization and mapping) because many researchers have been trying to solve mapping and localization (determining a robot's pose) problems in conjunction [Thrun, 2003]. However this thesis does not try to contribute to this field of research because we will deal with the mapping problem assuming that robot's pose is known at any instant in time.

Over the last decades, several approaches have been proposed on the research field of obstacle detection systems for service robots. Some of the most successful ones are based on assumptions about terrain's geometry [Konolige et al., 2008], [Batavia and Singh, 2002], on performing traversability analysis of the robot's surroundings [Hamner et al., 2008], on creating representations of the environment and using them to detect obstacles or safe paths for the robot [Lacaze et al., 2002], on using statistical analysis of a 3D point cloud in order to characterize obstacles [Lalonde et al., 2006], or on describing obstacles in terms of geometrical relationships between 3D points [Manduchi et al., 2005]. These approaches are reviewed with more detail on chapter two.

The main goal of this dissertation is to present a LADAR-based solution for mapping and obstacle detection in structured environments, either indoor or outdoor. Off-road terrains are often unstructured environments and, consequently, they are beyond the scope of this work. In the context of this work it is assumed that the mapping and obstacle detection system travels in an unknown environment, i.e. the use of any a priori knowledge about its surroundings is discarded.

1.1 Problem Statement

As previously stated, this dissertation intends to present a solution for the problem of mapping and obstacle detection in indoor/outdoor structured environments that is targeted to a service robot equipped with a LADAR. In the development of such a solution, some main problems must be taken into consideration:

1. The proposed model must be suitable for structured environments (indoor or outdoor), where structured surfaces must be correctly mapped and classified either as obstacles or freespace. As previously referred, off-road terrains are not considered in the context of this dissertation. Thus, unstructured surfaces may not be correctly mapped or classified.

2. The proposed model must be robust to variations in service robot's attitude, i.e. changes in pitch, roll and yaw angles. These changes on robot's attitude could lead to false positives on obstacle detection which, in turn, should be avoided.

3. The proposed model must be computationally efficient and cope with real-time constraints, so that service robot's safety can be assured.

1.2 Solution Prospect

In order to overcome the problems mentioned on the previous section, this dissertation proposes the following solutions:

1. In this model, obstacles are defined as surfaces that stand above or below the plane where the service robot is based and that can prevent a wheeled service robot from passing through. The LADAR is positioned in order to enable the system to detect obstacles that stand below LADAR's height and also negative obstacles (obstacles that stand below the plane where the service robot is based), which are common in these environments.

2. The robustness in terms of variations on robot's attitude is achieved with the help of an Attitude and Heading Reference System (AHRS) that provides *pitch*, *roll* and *yaw* angles. This information is added to the mapping and obstacle detection algorithm so that it can be adapted to the current attitude values.

3. Although this model is based on the construction of terrain maps, which can require considerable storage capabilities, efforts were made to maintain low complexity on the algorithm and low computational cost on the performance of the system, mainly in the choice of an appropriate size and resolution for the maps.

This system is implemented on a framework for mobile robotics applications named *Player/Stage Project* that improves the communications between different modules of the system and increases computational efficiency by executing several navigation tasks simultaneously.

1.3 Dissertation Outline

This dissertation is organized as follows:

Chapter 1: introduces the reader to the subject of mapping and obstacle detection using a LADAR and lists some problems related to this subject. A solution prospect for these problems is also presented;

Chapter 2: exposes a brief overview of the state of the art about obstacle detection for service robots;

Chapter 3: gives an introduction to some supporting concepts used in this work;

Chapter 4: describes the mapping and obstacle detection system proposed;

Chapter 5: presents the experimental results;

Chapter 6: encompasses the conclusions about the developed work and presents some future work possibilities.

2. State of the Art

In the last decades, there have been a large number of contributions on the research field of obstacle detection systems for service robots. This chapter presents an overview of some approaches that were proposed on this area. In the sections of this chapter, the author's intention is to present different approaches to the problem of obstacle detection for service robots, and not a historical evolution of the research in this area.

On section 2.1 an approach that uses the flat terrain assumption is presented. This method takes advantage of simplifications in order to simplify and speed up the process of detecting obstacles. Section 2.2 presents a method that was designed for terrains that show smooth slope changes and that uses gradient techniques to detect obstacles. A different approach is exposed on section 2.3. It uses traversability analysis of the robot's surroundings instead of taking considerations about terrain's geometry. Section 2.4 exhibits a method that creates a representation of the environment and uses it to detect safe paths for the robot. Section 2.5 shows an obstacle detection algorithm that performs statistical analysis of a 3D point cloud in order to characterize obstacles. Finally, a technique that describes obstacles in terms of geometrical relationships between 3D points is exhibited on section 2.6.

2.1 Flat terrain assumption

When travelling in outdoor conditions, an autonomous mobile robot may be confronted with structured environments such as urban terrains or man-made facilities, or with rougher terrains with great slope variations like the off-road case. Despite that, it is often possible to find a dominant ground plane. The presence of a ground plane simplifies processing and reduces the complexity of obstacles characterization. In fact, if a relatively flat ground plane is assumed, it is possible to simply define obstacles as salient surfaces standing above or below the ground.

Konolige et al. [Konolige et al., 2008] proposed an obstacle detection algorithm based on the assumption that the robot travels on a locally flat ground. The authors use a stereo vision camera to obtain the disparity and color images. The disparity image is used to compute a 3D

point cloud of the environment and then the ground plane is determined by a RANSAC technique [Fischler and Bolles, 1981]. In this model, obstacles are defined as points that lie too high above the ground plane, but lower than the robot's height. Sight lines are used to infer freespace to more distant points and their computation is achieved by finding columns of ground plane pixels that lead up to a distant obstacle. The color image is applied in path analysis. This work is illustrated in Figure 2.1.

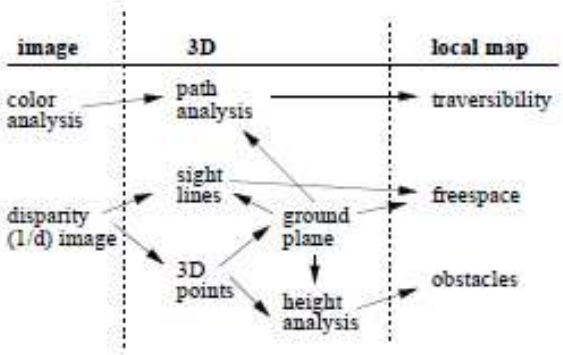


Figure 2.1 - Visual processing diagram proposed by [Konolige et al., 2008].

This approach presents good results on flat terrains but, at the same time, is very susceptible to fail on rough ones where the determination of the dominant ground plane, which is the key feature of this kind of methods, is not very reliable. Besides that, the authors only consider obstacles as points that stand above the ground plane not taking into consideration "negative obstacles", i.e. obstacles that stand below the ground plane, and this is a great drawback in outdoor navigation.

2.2 Obstacle detection in terrains with slightly variable slope

In the previous section, obstacle detection was performed on the assumption of a flat terrain. In this section a more generic method that also deals with non-flat terrains is presented.

Batavia and Singh [Batavia and Singh, 2002] proposed a process for obstacle detection that uses a two-axis laser scanner to obtain the input data. This approach is suitable for cases

where the terrain changes his slope smoothly enough to define obstacles as discrete discontinuities.

The two-axis laser scanner used by the authors consists of a single line laser range finder that operates as a two-axis scanner by being rotated so that the laser scans vertically instead of horizontally, and then mechanically swept from side to side to provide horizontal coverage. In their work, the authors consider a “scan” as one line of laser data, scanned vertically, and a “sweep” as a set of scans, collected by mechanically sweeping the laser from side to side.

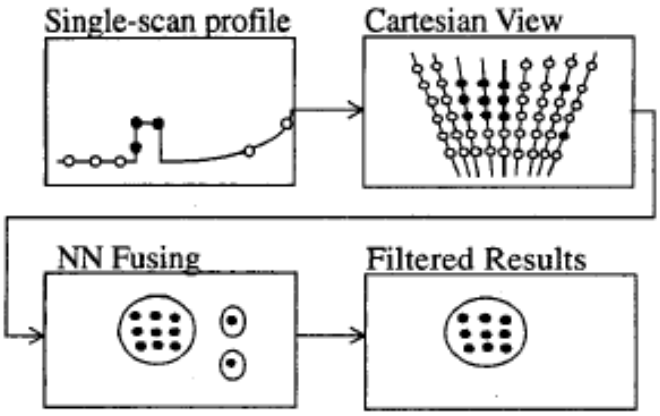


Figure 2.2 - Overview of the obstacle detection algorithm proposed by [Batavia and Singh, 2002]

The obstacle detection algorithm is summarized in Figure 2.2 and consists of two phases: classification and fusion. In the classification phase, each scan line is converted to Cartesian coordinates and terrain’s gradient is computed along the scan line. A threshold is then applied to the gradient to classify each pixel as ‘obstacle’ (dark dots on Figure 2.2) or ‘freespace’ (white dots on Figure 2.2). Each classified scan is then saved in a buffer containing a time-history of scans. This buffer is represented on the block named “Cartesian View” on Figure 2.2. The duration of this ‘time window’ establishes the amount of data that will be fused. In the fusion phase, obstacle pixels are clustered using a nearest neighbor (NN) criterion and candidate obstacles are then filtered based on their mass and size.

The two-axis scanning procedure used by the authors is obtained by sweeping a common (single-axis) laser scanner. In some cases, depending on the purpose of the robot that incorporates the obstacle detection system, this technique may consume too much time for real-time operation.

2.3 Traversability

Rather than taking geometric considerations about the terrain, there are some approaches that employ traversability analysis. Instead of determining if a certain region of the environment is an obstacle or freespace, this kind of obstacle detection systems try to avoid such a binary decision assigning to each region of the robot surroundings a cost value that represents the degree of difficulty for the robot to move across that region.

Hamner et al. [Hamner et al., 2008] present an obstacle detection system that performs traversability analysis. The authors use two laser range finders (one that is fixed and another that sweeps about an orthogonal axis) and a sliding window of point cloud data (obtained from both lasers) that is registered over time. Vehicle-sized planar patches are fit to the point cloud data and this process allows the settlement of three parameters: plane orientation (roll, pitch), terrain's roughness (obtained by the residual of the fitting process) and the height of data points above the plane. This method produces a grid-based traversability map and the plane fitting process is applied to each cell of the map, in order to acquire the parameters that are used to compute a hazard score that corresponds to the traversability measure of each cell. Furthermore, the authors complement the traversability analysis with gradient analysis from [Batavia and Singh, 2002] presented in section 2.2 in order to improve their results.

The plane fitting process performed in this work has a large computational cost and this is a main concern in real-time obstacle detection for mobile robots. Although it compensates some weaknesses of each individual algorithm, the option of combining two different kinds of analysis for obstacle detection increases the complexity of the system. This system is also more expensive than other approaches because it makes use of two laser scanners and a sweeping system.

2.4 Representations of the environment

Another well-known method to perform obstacle detection consists in creating representations of the environment and using them to detect obstacles or safe paths for the robot.

Lacaze et al. [Lacaze et al., 2002] propose the creation of an elevation map of the terrain in order to detect the support surface for the vehicle and to avoid obstacles. First, the measurements of a LADAR are geometrically transformed into an elevation map centered on the vehicle and this is done as the vehicle moves through the terrain. During this process each tile of the map is assigned with the number of times it has been seen by the sensor. A height threshold is applied to the elevation map to determine the support surface for the vehicle. Then, the authors try to predict safe trajectories for the vehicle along the elevation map through the computation of cost functions for each potential trajectory. These cost functions depend on several parameters such as existence of protruding objects, roughness of the terrain, number of times each cell has been seen by the sensor and pitch and roll along each trajectory. The authors use a vehicle model to predict pitch and roll along each path by placing vehicle masks along each potential trajectory in the elevation map.

Instead of assessing the content of each tile to try to find obstacles, the authors use the elevation map to estimate safe trajectories for the robot by calculating cost functions. This makes the method computationally heavy, especially because of the multiple placing of vehicle masks, and causes it to have difficulties to deal with real-time constraints.

2.5 Statistical analysis

Lalonde et al. [Lalonde et al., 2006] propose a different technique to perform obstacle detection. In this work the authors present a method that employs statistical analysis of a 3D point cloud that is built incrementally as the robot navigates through the terrain.

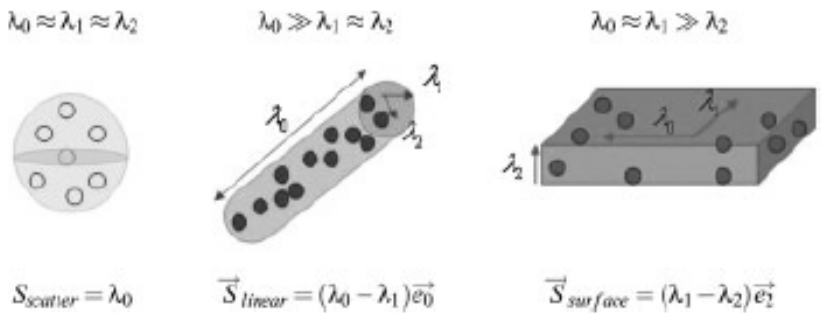


Figure 2.3 - The three classes used by [Lalonde et al., 2006] to classify the 3D point cloud

This analysis tries to classify the 3D point cloud into three classes: surfaces (ground surface, rocks), linear structures (wires, branches) and scattered regions (vegetation). The classification is based on the comparison of the eigenvalues obtained from the calculation of a covariance matrix for all the points within a neighborhood of a certain point.

As shown on Figure 2.3, scattered regions have no dominant eigenvalue whereas linear structures present one main eigenvalue and surfaces have two eigenvalues that prevail.

This obstacle detection algorithm is suitable for vegetated terrains and the authors present good results using measurements from laser scanners but, nonetheless, for structured environments simpler approaches can be used.

2.6 Geometrical relationships

A method that examines geometrical relationships between points of a 3D point cloud was developed by Manduchi et al. [Manduchi et al., 2005]. These authors try to detect obstacles by analyzing slant and altitude of visible surface patches directly in the range image domain. A visible surface patch is considered an obstacle if its slope is larger than a certain value θ (the maximum slope a robot can climb) and if it spans a vertical interval larger than a threshold H (the minimum height an obstacle must have to block robot's passage). Slant and altitude measures are taken from the search for pairs of compatible points in the 3D point cloud.

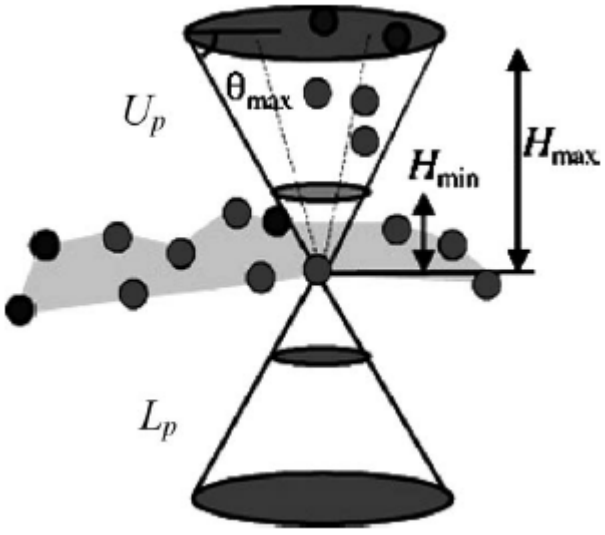


Figure 2.4 - Compatibility relationship defined by [Manduchi et al., 2005]

This compatibility relationship is illustrated in Figure 2.4 and is expressed by the authors as follows: *"The points compatible with a surface point p are those belonging to the two truncated cones U_p and L_p with vertex in p, axis oriented vertically, and limited by the two planes of equation y = Hmin and y = Hmax respectively "* [Manduchi et al., 2005].

Each point is considered an obstacle point if it has at least one compatible point that belongs to the same surface. The authors use stereo-vision images and, for each image pixel, a search for compatible points is executed with the aim of finding obstacle points.

This obstacle detection system has attracted particular interest by several researchers, mainly due to the distinct definition of an obstacle, which is based on the concept of compatibility between 3D points. However, this approach has limitations regarding real-time obstacle detection because its computational cost easily becomes a problem as the number of 3D points increases. Also, this method is very sensitive to incorrect 3D points generated by stereo-vision, commonly known as outliers.

3. Supporting concepts

This chapter summarizes some concepts that helped on the development of this mapping and obstacle detection system. Section 3.1 describes some coordinate systems and coordinate transformations that are useful in mobile robotics. Section 3.2 presents the Player/Stage Project, which is an open-source framework that simplifies the development of control architectures for several applications, including mobile robots.

3.1 Coordinate Systems and Coordinate Transformations

There are many ways of representing a location in the world by a set of coordinates. This section presents some coordinate systems that are often used in navigation with Inertial Navigation Systems [Grewal et al., 2001]. Coordinate transformations that are useful in mobile robotics and, in particular, in this work, are also exposed in this section.

3.1.1 LTP coordinates

Grewal defines Local Tangent Plane (LTP) coordinates as *"(…) local reference directions for representing vehicle attitude and velocity for operation on or near the surface of the earth"* [Grewal et al., 2001]. A frequent orientation for this kind of coordinates has one horizontal axis (the east axis) in the direction of increasing longitude and the other horizontal axis (the north axis) in the direction of increasing latitude. A common LTP coordinate system is the North-East-Down (NED).

In NED, the direction of a clockwise turn is in the positive direction with respect to a downward axis. This coordinate system is used in many applications because its coordinate axes coincide with vehicle-fixed roll-pitch-yaw (RPY) coordinates when the vehicle is level and headed north [Grewal et al., 2001].

3.1.2 RPY coordinates

Grewal [Grewal et al., 2001] defines roll-pitch-yaw (RPY) coordinates as *"(…) vehicle-fixed, with the roll axis in the nominal direction of motion of the vehicle, the pitch axis out the*

right-hand side, and the yaw axis such that turning to the right is positive". Figure 3.1 illustrates this coordinate system.

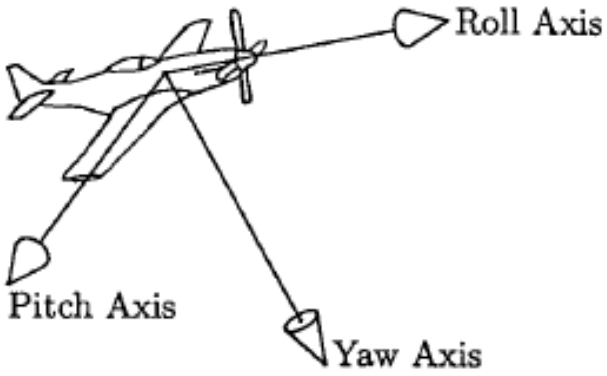


Figure 3.1 – Roll, Pitch and Yaw axes [Grewal et al., 2001].

The angles of rotation about the vehicle *roll*, *pitch* and *yaw* axes are called the *Euler* angles [Grewal et al., 2001]. These angles can specify the attitude of the vehicle body with respect to local coordinates.

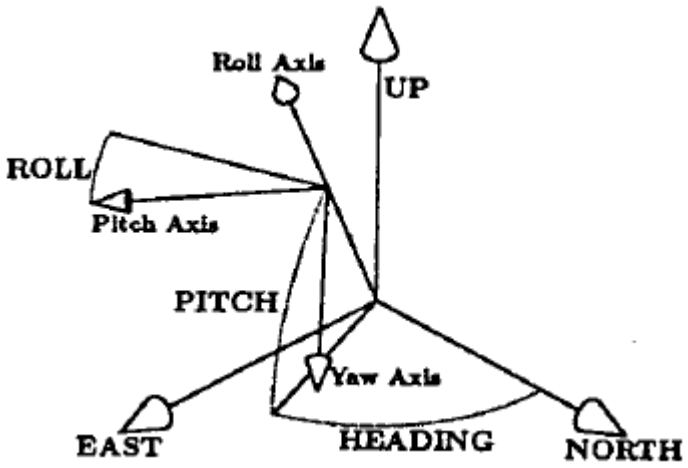


Figure 3.2 – Vehicle Euler Angles defined by Grewal [Grewal et al., 2001].

A common convention for Euler angles is illustrated on Figure 3.2 and is defined by Grewal [Grewal et al., 2001] as a set of three rotations, starting with the vehicle level with *roll* axis pointed north, as follows:

- First rotate through the *yaw* angle (*Y*) about the vehicle *yaw* axis to the intended azimuth (heading) of the vehicle *roll* axis. The azimuth is measured clockwise (east) from north;

- Then, rotate through the *pitch* angle (*P*) about the vehicle *pitch* axis to bring the vehicle *roll* axis to its intended elevation. Elevation is measured positive upward from the local horizontal plane;

- Finally, rotate through the *roll* angle (*R*) about the vehicle *roll* axis to bring the vehicle attitude to the specified orientation.

Figure 3.3 illustrates how the rotations through these angles can affect the orientation of an object, in this case an airplane.

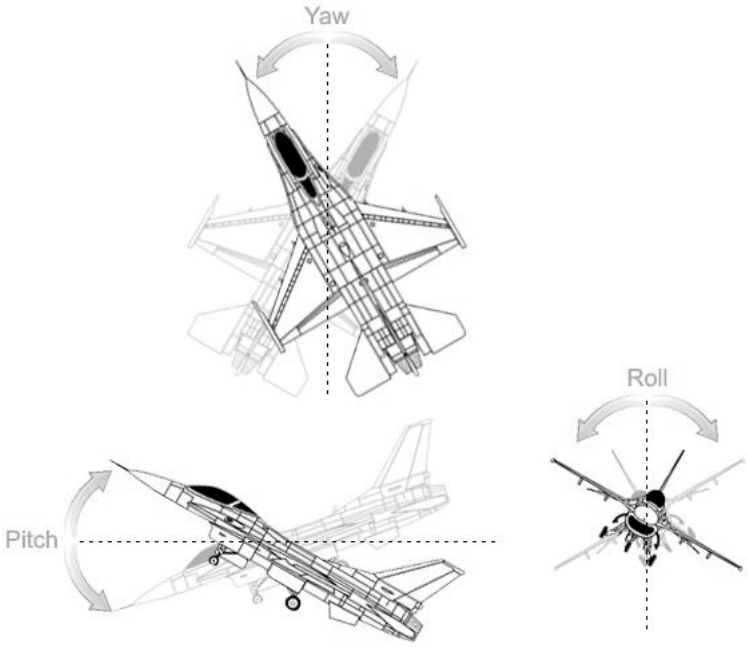


Figure 3.3 – Rotations through Roll, Pitch and Yaw angles [ACME, 2009].

This set of three angular rotations is often used to define a coordinate transformation to bring one coordinate frame to coincide to another. To achieve this, the rotation of each axis is specified by a rotation matrix. For example, the coordinate transformation from RPY coordinates to NED coordinates is given by the product of three rotation matrices, as shown on Figure 3.4 [Grewal et al., 2001].

$$C_{\text{NED}}^{\text{RPY}} = \overbrace{\begin{bmatrix} C_Y & -S_Y & 0 \\ S_Y & C_Y & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{Yaw}} \overbrace{\begin{bmatrix} C_P & 0 & S_P \\ 0 & 1 & 0 \\ -S_P & 0 & C_P \end{bmatrix}}^{\text{Pitch}} \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_R & -S_R \\ 0 & S_R & C_R \end{bmatrix}}^{\text{Roll}}$$

$$= \overbrace{\begin{bmatrix} C_Y P_P & -S_Y C_R + C_Y S_P S_R & S_Y S_R + C_Y S_P C_R \\ S_Y C_P & C_Y C_R + S_Y S_P S_R & -C_Y S_R + S_Y S_P C_R \\ -S_P & C_P S_R & C_P C_R \\ \text{(roll axis)} & \text{(pitch axis)} & \text{(yaw axis)} \end{bmatrix}}^{\text{in NED coordinates}},$$

Figure 3.4 - Transformation from RPY coordinates to NED coordinates [Grewal et al., 2001].

A similar convention for Euler angles is given by Craig [Craig, 2005]. Given two frames *A* and *B*, the order of rotations is as follows: starting with frame *B* coincident with a known frame *A*, rotate first *B* about Z_B by an angle α, then about Y_B by an angle ß, and, finally, about X_B by an angle γ. This set of Euler-angle rotations is exemplified on Figure 3.5.

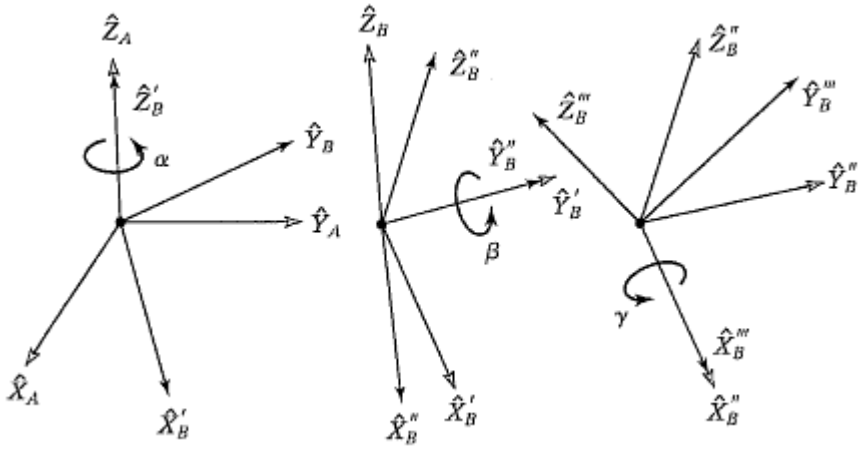


Figure 3.5 - Euler Angles defined by Craig [Craig, 2005].

The final orientation of *B* relative to *A* is also given by the product of three rotation matrices, as follows:

$$\begin{aligned} {}^{\text{A}}_{\text{B}}\text{R}_{\text{Z,Y,X}}(\alpha,\beta,\gamma) &= \text{R}_{\text{Z}}(\alpha)\text{R}_{\text{Y}}(\beta)\text{R}_{\text{X}}(\gamma) \\ &= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} = \end{aligned}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \sin\alpha\sin\gamma + \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \quad \textbf{Equation 3.1}$$

If one considers that the Euler angles of Craig's [Craig, 2005] convention, α, $ß$ and γ, correspond, respectively, to the Euler angles of Grewal's [Grewal et al., 2001] convention, *Y*, *P* and *R*, one can see that the results from Figure 3.4 and from Equation 3.1 are equivalent.

3.2 Player/Stage Project

The Player/Stage Project provides open-source tools that simplify controller development, particularly for multiple-robot, distributed robot, and sensor network systems [Vaughan et al., 2003]. This project includes the *Player* server, and the robot simulators *Stage* and *Gazebo*. According to its authors and developers, the *Player* server *"(…) is probably the most widely used robot control interface in the world."* [Player Project, 2010].

Player is a network server for robot control [Player Project Wiki, 2010]. When operating on a robot, *Player* provides a straightforward interface to the robot's actuators and sensors over an IP network. The user can create a *Client* program that configures devices, reads data from sensors and writes commands to actuators by interacting with *Player* over a TCP socket.

Stage is a 2D multiple-robot simulator from the Player project [Player Project Wiki, 2010]. It can simulate a group of mobile robots driving and sensing a two-dimensional environment. This simulator is prepared to provide virtual robots that make use of simulated devices instead of physical sensors, taking advantage of several sensor models including sonars, laser rangefinders, cameras, etc.

Gazebo is a multi-robot simulator for outdoor environments [Player Project Wiki, 2010]. Its basis is similar to *Stage*, but the robots and sensors are simulated on a three-dimensional environment.

Player server is the only component of the Player/Stage Project that will be used in this work and, thus, it's also the only one of the three components that is explored in more detail in this chapter.

Player defines a set of standard *interfaces*, each of which is a specification of the ways that you can interact with some class of devices [Player Manual, 2010]. For example, the *laser interface* specifies a format (messages and their contents) in which a laser range finder

can return its measurements (generally, a list of ranges and some scanning parameters). This *interface* can be used to interact with different kinds of laser range finders, such as SICK and Hokuyo Laser Range Finders.

Another key concept in *Player* is the concept of *driver*. It is defined by the authors of Player Project as "*A piece of software (usually written in C++) that talks to a robotic sensor, actuator, or algorithm, and translates its inputs and outputs to conform to one or more interfaces*" [Player Manual, 2010]. The *driver*'s job is to adapt the specific language of an equipment or algorithm to the format of the corresponding *interface*. This way, two different sensors of the same class can provide data in the same format to *Player*, using the same interface. Most *Player* drivers communicate directly with hardware, but it's also possible to use a different kind of driver - the *abstract driver* - that communicates with other drivers instead of hardware components. *Player* has a lot of *drivers* and *abstract drivers* already developed and that are ready to be used by the developer.

The concept of *device* is closely linked with the concepts above mentioned. A *device* represents a connection between a *driver* (or an *abstract driver*) and an *interface*. Each *device* is assigned with a specific address that is used for message exchange which occurs between *devices* and with the help of *interfaces* [Player Manual, 2010]. This address can be composed by several fields such as host, robot (port), interface or index. Only the last two fields are mandatory [Owen, 2010].

The connection between a *driver* and an *interface* is specified in a configuration file, named *config file*. The user must write this file containing all the information that *Player* must know about the equipment that will be used. This file tells *Player* which *drivers* will be used and which *interfaces* they provide or require. The declarations of *drivers* or *abstract drivers* on the *config file* can also contain some parameters related to the sensor or algorithm that corresponds to the *driver* or *abstract driver*, respectively.

The relationships between these three important concepts of *Player* (*interface*, *driver* and *device*) can be easily understood with the help of an example: As mentioned before, there are several drivers that came with *Player*. One of them is the *sicklms200 driver*. This *driver* controls a SICK LMS200, which is a laser range finder that is popular in mobile robotics applications. This *driver* is able to communicate with the SICK LMS200 over a serial port and receive range data from it. Moreover, the *sicklms200 driver* translates the range data received in a specific SICK format to the format defined by the *laser interface*. Thus, *Player* must know that this *driver* provides a *laser interface*. To achieve this, the *sicklms200 driver* and the *laser interface* are associated to create a *device*. This is performed in a *config file*. An example of a declaration of a *driver* on a *config file* can be given by:

```
driver
(
        name "sicklms200"    // name of the driver
        provides ["laser:0"]    // [“interface:index”]
        scanning frequency 50          // parameter
)
```

On this example, a *driver* with the name *sicklms200* is declared. The device address is defined on the ‘provides’ section. This address is composed by the *laser interface* (the *interface* through which the *driver* provides data) and the *index* of the *device* which is *0*. It also has a parameter named scanning frequency with the value 50.

The Player/Stage Project has also a set of libraries, named Client Libraries, which allow the user to communicate with the *Player* server from an external program. This communication can be achieved by using *Proxies* that are defined in the Client Libraries. *Proxies* are C++ classes that offer methods to request data and/or send commands from and to the *Player* server and they are closely related to the *interfaces* defined on *Player*, i.e. for each *interface* there should be a corresponding *Proxy* in the Client Libraries. The user must build a *Client* program that uses *Proxies* to communicate with *Player*, in order to request data from sensors and send commands to actuators.

The global architecture of the Player/Stage Project is illustrated on Figure 3.6.

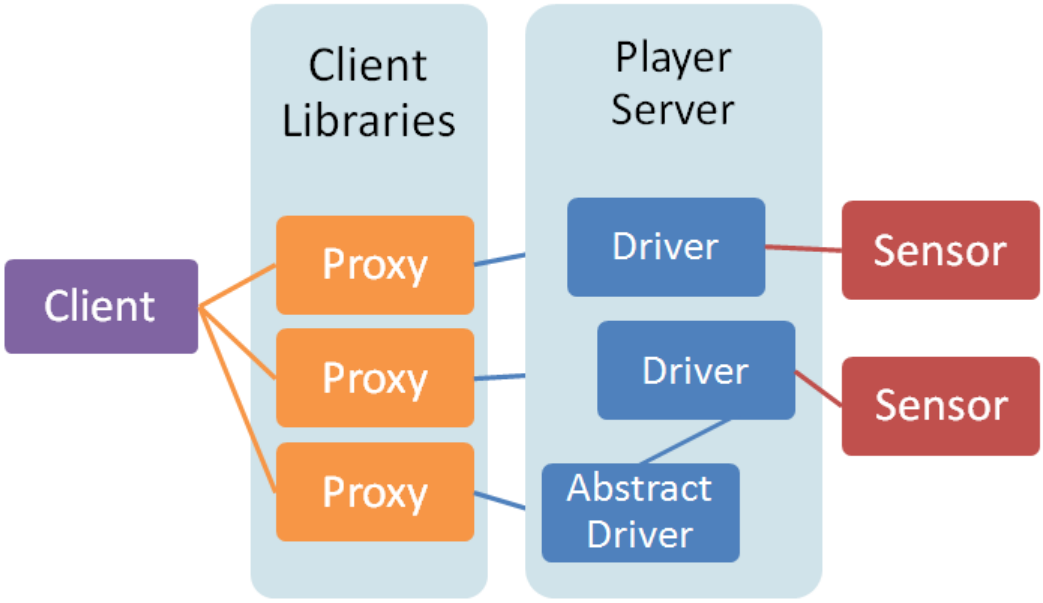


Figure 3.6 - Global architecture of Player/Stage Project

This figure shows that the *Client* program built by the user requests data or sends commands through the methods contained in *Proxies*. These methods offered by *Proxies* are prepared to send messages to *drivers* in order to deliver those requests and/or commands. These messages are routed by *Player Server* that knows which *interfaces* each *driver* supports and which *Proxies* correspond to those *interfaces*. Finally, the drivers communicate with the hardware of the robot to perform the actions desired by the *Client*.

As referred before, *Player* has several *drivers* and *abstract drivers* already developed. However, the user may not have the hardware or may not want the algorithms that these *drivers* control. In these situations, the user should develop its own *driver*. For this, it's important to know the methods contained on a *driver* and its run-time process. The methods that should be present on a *driver* are:

- **Driver**(ConfigFile* cf) : The constructor of the *driver*. It reads all the information present on the *config file* that is related to this *driver*, including the *interfaces* it provides and/or requires and some parameters that may have been defined. The parameter *cf* represents the *config file* that loads the *driver*;

- **MainSetup**() : This method allocates resources before entering the main loop. It is also useful to perform initializations or error checking before the main loop becomes active. In general, the developer should put here everything that only needs to be done once;

- **pthread_testcancel**() : This function is prepared to check whether the driver's thread should be killed and to cause the driver to break out of the **Main**() loop and go to the **MainShutdown**() method;

- **ProcessMessages**() : A very important method that processes the messages present on the message queue. In this method, the *driver* can receive data (e.g. from other *drivers*) and requests (e.g. from *Clients*). The *driver* can publish its data here, if it receives requests for it;

- **Main**() : The core of *driver*'s functioning. This method should contain a main loop and have specific function calls to critical methods, such as

ProcessMessages() or **pthread_testcancel**(). If the *driver* doesn't need a request to publish its data, it can be published in the Main function;

- **MainShutdown**() : This method is called when the *driver* is about to be stopped by *Player*. It's useful to deallocate resources, disconnect from ports, etc;

- **~Driver ()** : The destructor of the *driver*.

The run-time process of a *Player driver* is exemplified on Figure 3.7 [PSU Robotics RoboWiki, 2010].

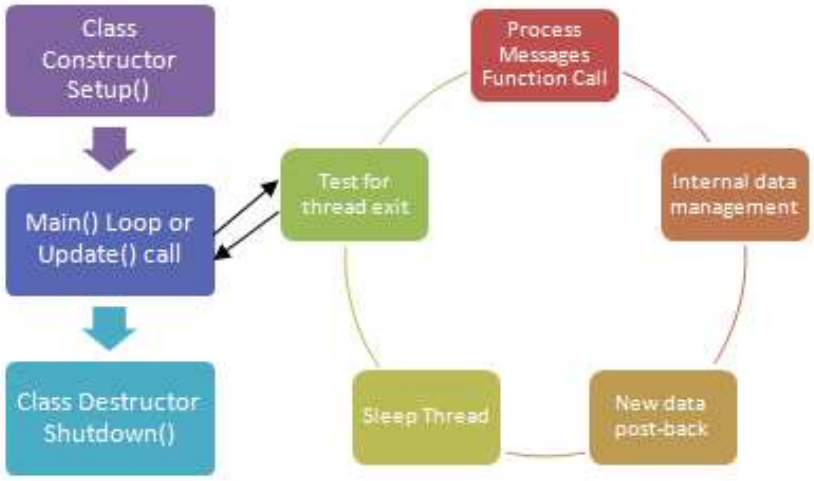


Figure 3.7 - Run-time process of a *Player* driver [PSU Robotics RoboWiki, 2010].

First, the *Constructor* and the *MainSetup* methods are executed. Then, the *driver* enters in its core function, i.e. the *Main* loop. In this loop, the *driver* continuously verifies if *Player* wants it to be stopped and also checks for new messages, executes the corresponding actions and publishes new data. Finally, when the *driver* is stopped, the *MainShutdown* and the *Destructor* methods are performed.

4. Mapping and Obstacle Detection system

This chapter presents a mapping and obstacle detection system for service robots that work in indoor/outdoor structured environments and that are equipped with, mainly, a LADAR and an AHRS sensor. Section 4.1 shows how the LADAR is positioned in this system, in order to build *elevation maps* of the environment. Section 4.2 exposes the mapping procedure that is responsible for the creation of the *elevation map*. On Section 4.3, it is described the obstacle definition used in this model. This definition and the previously obtained *elevation maps* are used by a simple obstacle detector, which is described on Section 4.4. This obstacle detection procedure generates a map (*obstacle map*) that represents only obstacles and freespace and that is more suitable to be used by a path planner than an *elevation map*. Section 4.5 presents the scrolling procedure that is used for the *elevation map* and depicts the overall functioning of this mapping and obstacle detection system. Section 4.6 describes the attitude compensation procedure used on this system. This algorithm uses an AHRS sensor and allows this system to be more robust against variations on robot's *pitch*, *roll* and *yaw* angles. Finally, Section 4.7 describes the implementation of this mapping and obstacle detection system on a platform for robotic applications named *Player/Stage*.

4.1 LADAR's positioning

The LADAR is a range sensor that is based on the "time of flight" principle, i.e. each distance provided by this sensor is computed from the propagation time that a pulse of light takes to travel from the source to the target and back to the receptor. At each scan, the LADAR provides a set of distances computed along its field of view (see Figure 4.1). Usually, the emitted laser beams are deflected using a mirror and, thus, the LADAR scans the surroundings in a circular manner. The measurements are detected at regular angular steps and each scan proceeds counterclockwise about the LADAR, i.e. from d_1 to d_n on Figure 4.1. This figure illustrates some of the features involved on LADAR's operation. This kind of range sensors is also called LIDAR (LIght Detection And Ranging).

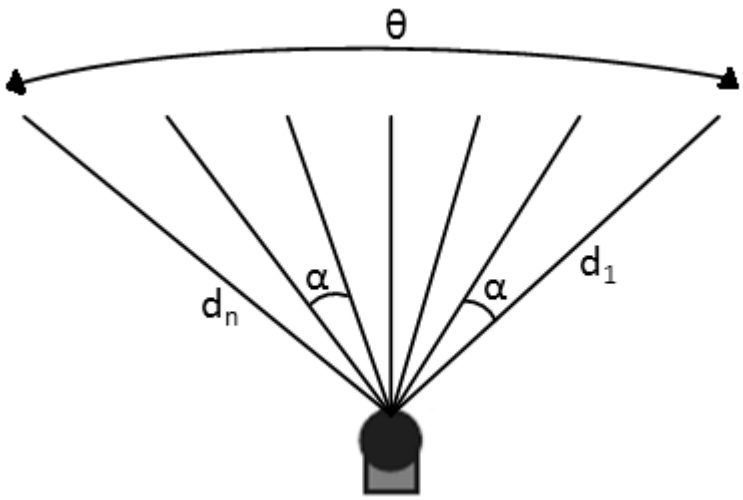
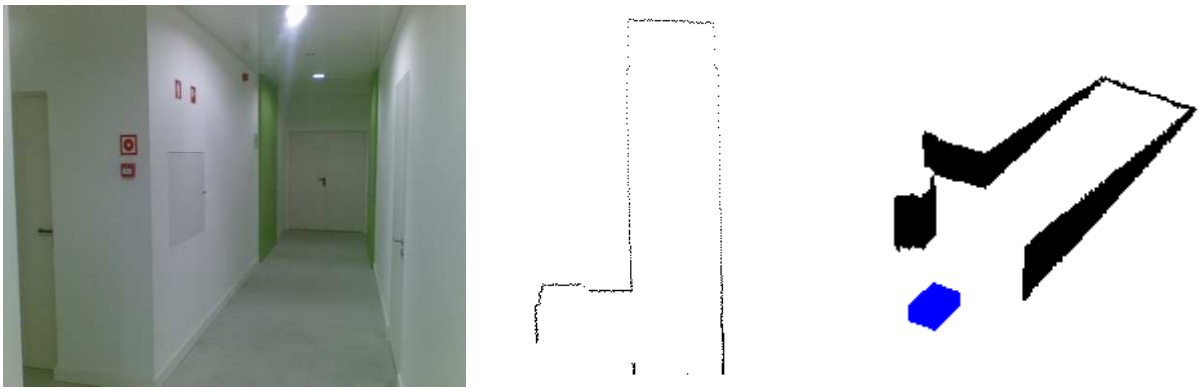


Figure 4.1 – LADAR features. d_1 and d_n are measured distances, α is the angular step and θ is the field of view.

Typically, the LADAR can be used to build two dimensional or three dimensional maps of the environment. The 2D approach is normally achieved by setting the scanning plane parallel to the floor. If we consider the hallway presented in Figure 4.2 (a), a typical map obtained with this method is shown in Figure 4.2 (b).



(a) (b) (c)

Figure 4.2 – Two types of maps obtained using a LADAR. (a): Hallway; (b): 2D map of (a); (c): 2.5D map of (a).

This kind of maps is acceptable for a two dimensional world where all the objects should have specific shapes and should be tall enough to be detected by the scanning plane. However, this method faces many problems presented by the real world such as objects below or above the height of the LADAR, tables, sidewalks, fencings, ditches, among others. In this dissertation the LADAR is used in a way suitable for a three dimensional world. With this

approach it is possible to build two and a half dimensional maps, similar to the one presented on Figure 4.2 (c).

As stated before, this model uses a single LADAR which is placed on top of a mobile robot, with the scanning plane angled down towards the direction of motion, as shown in Figure 4.3. As the robot moves forward, the laser sweeps some space in front of the robot and a 2D ½ map is being built.

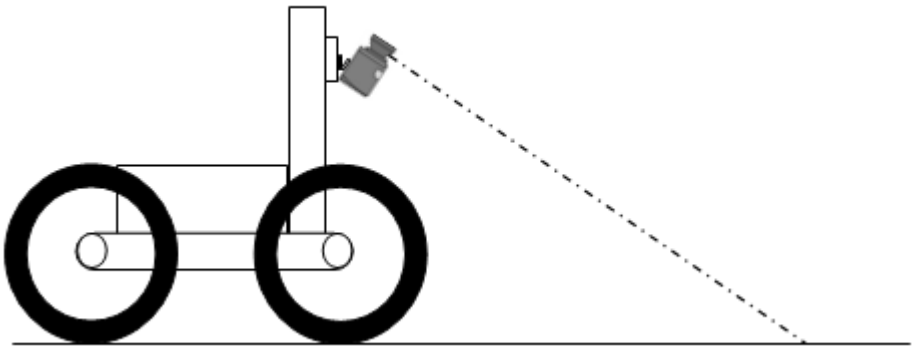


Figure 4.3 – LADAR's positioning for this model.

4.2 Mapping

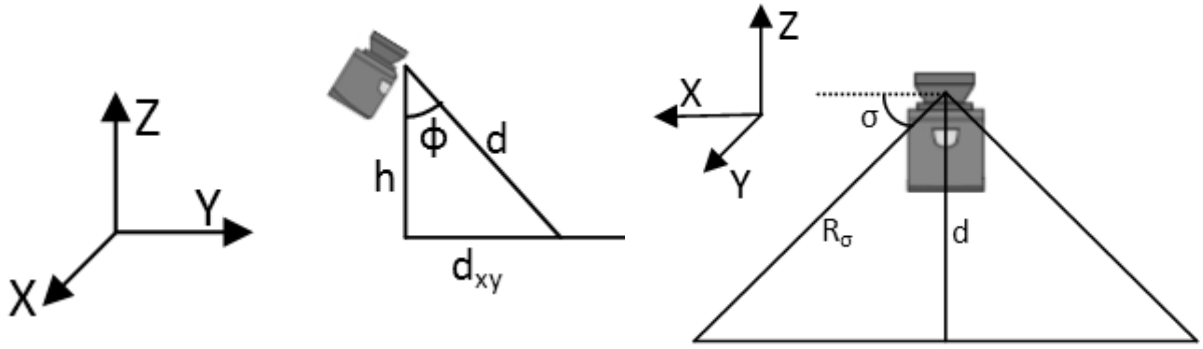
This section exposes the procedure that is behind the creation of the *elevation map*. It describes the computation of the *reference plane* (section 4.2.1) and also how to get the information that will be added to the *elevation map* and how the map is represented (section 4.2.2).

4.2.1 Computation of the reference plane

One of the first operations of the mapping procedure is the computation of a *reference plane*, which is the plane that the LADAR "sees", positioned as on Figure 4.3, without having any object inside its field of view. In other words, it's an "empty plane" composed by a set of ranges that the LADAR would provide if it only scans the surface where the robot is standing and no other object or surface is detected in its field of view. This *reference plane* is

continuously updated due to changes in LADAR's attitude and is used to compare with the *real plane*[4] provided by the LADAR in order to obtain information about terrain's elevation.

The calculation of the *reference plane* is based on some simple trigonometry concepts, which are illustrated on Figure 4.4. In these calculations it is assumed that the LADAR's lens is on the position (0, 0, *h*) according to the coordinate systems of Figure 4.4. Distance *d* represents the theoretical range provided by the LADAR's central beam, i.e. the beam that is emitted at $\sigma = 90°$. LADAR's height is represented by *h* and ϕ is an angle obtained from LADAR's tilt ($\phi = 90^{\circ}$ - tilt).



(a) (b)

Figure 4.4 – Calculation of the reference plane: (a) side view, (b) front view.

On Figure 4.4 (a) it's possible to see that distance *d* can be computed by the following equation:

$$d = \frac{h}{\cos \phi}$$

LADAR's height and tilt remain always with the same value, since the LADAR is fixed in a fix position. Thus, it would be expected that distance *d* would always remain the same as long as *h* and ϕ remain constant and no obstacles are detected. In fact, this will not happen because *d* will have to be updated due to changes in robot's attitude, but this problem will be explored on section 4.5.

Then, distance *d* is used to compute the LADAR's theoretical range at each angle σ (R_σ) as follows:

[4] In this dissertation, the term "real plane" refers to the plane that contains the set of ranges measured by the LADAR at each scan.

$$R_\sigma = \frac{d}{\sin \sigma}$$

The angle σ represents each angle where the LADAR emits a beam. These angles are obtained by iteratively adding the angular step α (see Figure 4.1) to the starting angle of the field of view.

The *reference plane* will be composed by the set of R_σ ranges computed along the field of view of the LADAR. Summarizing, the procedure to compute the *reference plane* consists of first computing distance d and then using this distance to determine the range R_σ for each angle σ of the field of view.

4.2.2 Map building

As stated before, the *reference plane* is used to obtain information about terrain's elevation by comparing it with the *real plane* provided by the LADAR. Hence, every time a scan is performed, and for each angle σ, the real range provided by the LADAR ($R_{real(\sigma)}$) is compared to the corresponding range R_σ of the *reference plane* to compute the height of a hypothetical obstacle. This procedure is illustrated on Figure 4.5 and Figure 4.6.

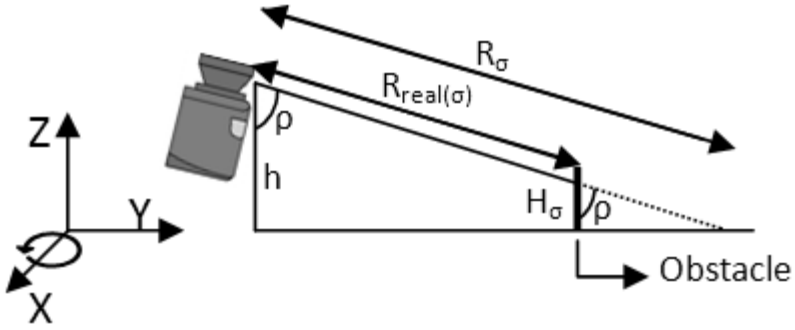


Figure 4.5 - Side view of the height computing process. Example for an obstacle placed on the positive Z axis. H_σ is the height in meters and ρ is an angle in degrees.

By looking at Figure 4.5 it's possible to see that the height of a hypothetical obstacle (H_σ) can be computed by the following equation:

$$H_\sigma = \cos(\rho) \times \left(R_\sigma - R_{real_{(\sigma)}}\right)$$

The angle ρ is obtained from laser's height h and each range R_σ as follows:

$$\rho = \cos^{-1}\left(\frac{h}{R_\sigma}\right)$$

For "negative" obstacles, i.e. obstacles placed on the negative Z axis, the procedure is similar but, in this case, $R_{real(\sigma)}$ has a bigger value than R_σ, as shown on Figure 4.6. Hence, as expected, the value obtained to H_σ is negative.

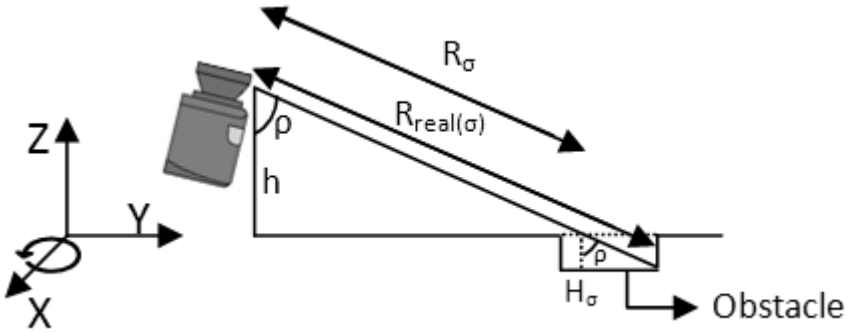


Figure 4.6 - Side view of the height computing process. Example for an obstacle placed on the negative Z axis. H_σ is the height in meters and ρ is an angle in degrees.

If there isn't any obstacle at a given angle σ, $R_{real(\sigma)}$ and R_σ will have approximately the same value and the value computed for H_σ will be very close to zero.

Finally, the computed height (H_σ) is saved in an *elevation map* (system's internal representation of the terrain's elevation) on a cell corresponding to the range provided by the LADAR. A grid map was adopted to represent terrain's elevation in a suitable way (see Figure 4.7). This map can be represented by the following expression:

$$\sum_{i=0}^{i=m}\sum_{j=0}^{j=n}(avg_{height})_{i,j}$$

The *elevation map* is a two-dimensional array of cells ($m \times n$) containing information based on the measures taken from the LADAR and it is LADAR-centered, i.e. the sensor is placed on the center cell of the map, which is also considered as the origin of the XY plane. Each cell in the *elevation map* represents an area, ($c_h \times c_w$) m^2. Therefore, the total area of

the map is $(m \times c_w) \times (n \times c_h)$ m^2. Each cell contains the average of the heights of all computed points that correspond to that cell. A representation of the *elevation map* used in this work is shown on Figure 4.7.

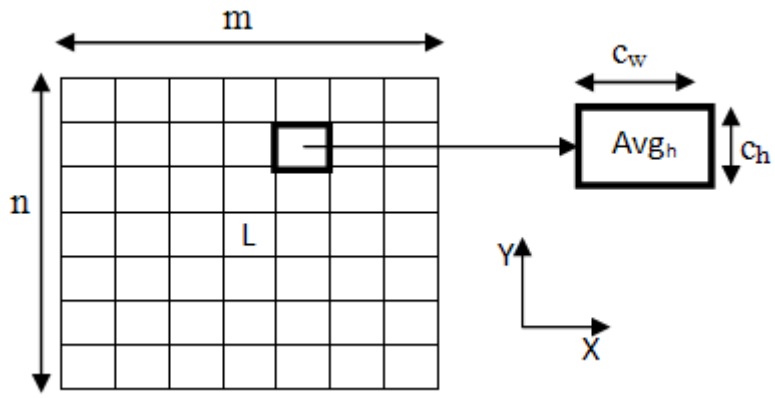


Figure 4.7 - Elevation map representation. *n* represents the number of height cells and *m* is the number of width cells. c_w and c_h represent cell's width and height, respectively, measured in meters. *L* symbolizes LADAR's position on the map.

Every time a LADAR scan is performed, the *elevation map* is updated with new information obtained from the scan. To achieve this, the average height of each cell that contains new information is updated with the corresponding height values computed from the new scan, as explained before.

4.3 Obstacle definition

As previously stated, this model intends to be suitable for structured environments. These environments usually have large planar surfaces where, generally, obstacles are distinguishable. Thus, in this case, an obstacle can be understood as an object that stands above or below those large planar surfaces where the service robot is based and that can prevent a wheeled service robot from passing through. In this model, it is assumed that the surface where the robot stands is considered as the ground plane and obstacles are classified in terms of their heights in relation to this ground plane. The obstacle definition used is:

Definition 1: A cell (i, j) of the *elevation map* is considered an obstacle if the following condition is met:

1. $Avg_h(i,j) > H_{pos} \cup Avg_h(i,j) < H_{neg}$

where $Avg_h(i,j)$ is the average height of the cell, H_{pos} is the minimum height that an object that stands above the ground plane must have to be considered as an obstacle and H_{neg} is the minimum height that an object that stands below the ground plane must have to be considered as an obstacle. All the cells that have average heights with values between H_{pos} and H_{neg} are considered as free space. Consequently, the values for parameters H_{pos} and H_{neg} must be chosen according to the service robot's dimensions in order to prevent the robot being damage when travelling cells considered as free space.

4.4 Obstacle detection

The obstacle detection procedure of this model is very simple and consists in applying the obstacle definition of section 4.3 to each cell of the *elevation map*. The information obtained on the obstacle detection procedure is saved on an *obstacle map* (see Figure 4.8) that is more suitable to be transferred to and used by a path planner.

The *obstacle map* is similar to the *elevation map* in terms of their appearance. Nonetheless, the content of each cell is different. In the *obstacle map*, each cell holds a value (c_v) that represents an obstacle or free space.

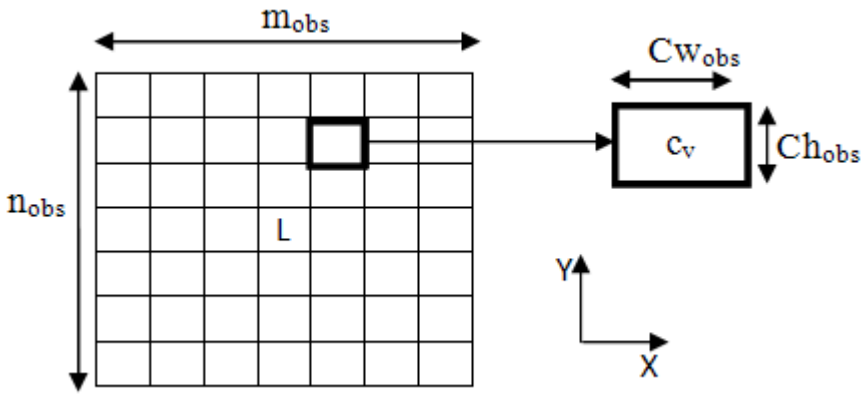


Figure 4.8 - Obstacle map. n_{obs} represents the number of height cells and m_{obs} is the number of width cells. Cw_{obs} and Ch_{obs} represent cell's width and height, respectively, measured in meters. *L* symbolizes LADAR's position on the map. c_v represents each cell's value.

Also their heights and widths can be different, because if the *obstacle map* is intended to be used by a path planner its size could be smaller than the *elevation map's* size in order to reduce the amount of information that is transferred. For example, the *obstacle map* can represent only a "window" of the *elevation map* maintaining the same resolution (cell's size). However, these details must be taken in consideration according to the chosen strategy for the path planning.

As previously mentioned, the value assigned to each cell of the *obstacle map* depends on the results of applying the obstacle detection definition to the *elevation map*. This value is assigned according to the following conditions:

$$c_v = \begin{cases} -1, & H_{neg} < Avg_h(i,j) < H_{pos} \\ 1, & Avg_h(i,j) > H_{pos} \cup Avg_h(i,j) < H_{neg} \end{cases}$$

If a cell of the *elevation map* is considered an obstacle, the corresponding cell of the *obstacle map* is assigned with the value 1. If the cell is considered as free space, the corresponding cell of the *obstacle map* is assigned with the value -1. This procedure is performed only for the cells of the *elevation map* that contain information in order to increase the computational efficiency of this procedure. The cells of the *obstacle map* that correspond to those of the *elevation map* that have no information are assigned with the value 0, which means "unknown terrain".

4.5 Map scrolling

Before the calculation and addition of new information, there is the possibility of *elevation map*'s cells being repositioned according to LADAR's displacement, but only when this displacement is larger than a cell's size. In this scrolling procedure, cells are only repositioned across the Y axis, i.e. each cell remains in the same column, changing only its line. Therefore, in order to decide if the scrolling procedure must be executed, the Y axis projection of LADAR's displacement is computed (using *yaw* angle) and this value is the one that is compared with the cell's size. Also, only the cells that contain information are repositioned, which increases the computational efficiency of this procedure.

In summary, for every LADAR scan the mapping and obstacle detection system is executed as exemplified by Figure 4.9.

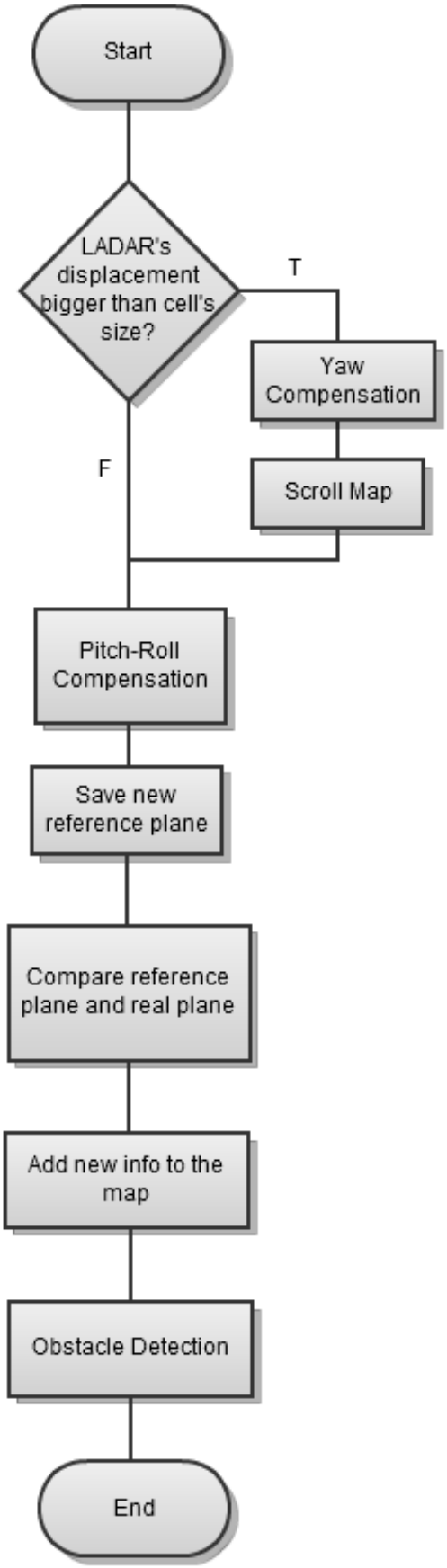


Figure 4.9 - Flowchart of the mapping and obstacle detection system.

First, and only if LADAR's displacement is bigger than the size of a cell, the *elevation map* is scrolled according to this displacement and *Yaw compensation* is performed. The procedure for *Yaw compensation* is addressed on section 4.6. After that, the *reference plane* is updated according to the LADAR's *pitch* and *roll* angles. This procedure, named on Figure 4.9 as *Pitch-Roll compensation* is also explored on section 4.6. Then, the new *reference plane* is stored so it can be used the next time that this whole procedure, represented on Figure 4.9, is performed. Afterwards, the new *reference plane* and the *real plane* are compared in order to obtain information about terrain's elevation and this new information is added to the *elevation map*. Finally, the obstacle detection procedure is performed in order to build the *obstacle map*, as explained on section 4.4.

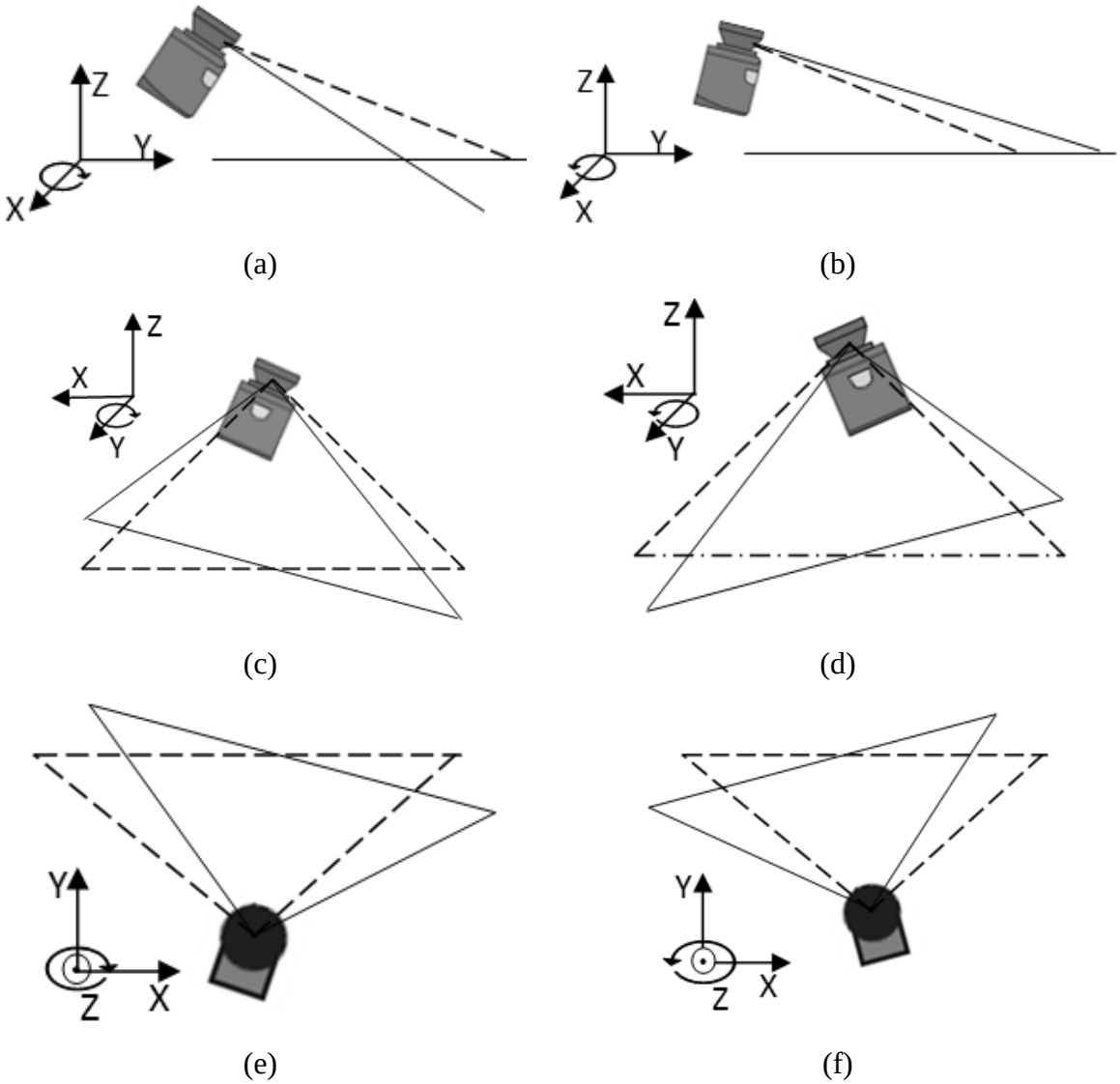


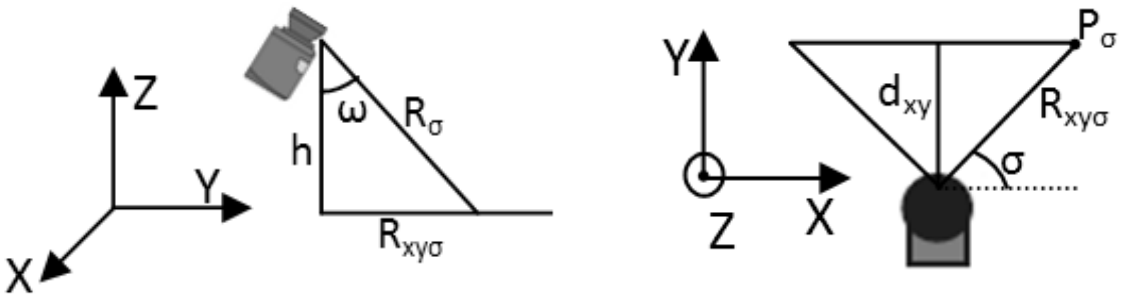
Figure 4.10 – Effects on the *reference* plane caused by variations on robot's attitude. (a) and (b) - side view of pitch changes. (c) and (d) – front view of roll changes. (e) and (f) – top view of yaw changes.

4.6 Robot's Attitude compensation

This section explains the procedures Pitch-Roll compensation and Yaw compensation of Figure 4.9. As stated before, changing robot's attitude (pitch, roll and yaw angles) may affect the mapping system and increase the number of false positives on obstacle detection. Namely, pitch and roll variations will affect the reference plane and, consequently, the elevation map, whereas yaw variations will affect only the elevation map. Figure 4.10 presents examples of how these variations may or may not affect the reference plane.

In this figure the ground plane is represented by the XY plane and the original *reference planes* are represented by dashed lines. By observing Figure 4.10 one can see that whenever LADAR's *pitch* or *roll* change, the *reference plane* must be updated because these changes affect the distances that belong to this plane. On the other side, *yaw* variations only change the orientation of the *reference plane* but they don't interfere on its distances. Therefore, whenever there are variations on *pitch* or *roll* values, the distances that comprise the *reference plane* must be updated so that the correct *reference plane* is used on the comparison with the *real plane*.

The update of the *reference plane* is achieved as follows: each distance R_σ at each angle σ is converted into an *XY* plane point (for example, point A of Figure 4.12), as shown on Figure 4.11.



(a) (b)

Figure 4.11 – Computation of each XY point of the reference plane. (a) – side view, (b) top view.

Assuming that the LADAR lens is on the position (0, 0, *h*) according to the coordinate systems of Figure 4.11, the distance $R_{xy\sigma}$ can be obtained by the following equation:

$$R_{xy\sigma} = R_\sigma \cdot \sin \omega$$

The angle ω is obtained from LADAR's height h and each range R_σ as follows:

$$\omega = \cos^{-1}\left(\frac{h}{R_\sigma}\right)$$

Finally, point $P_\sigma(x_{ref_\sigma}; y_{ref_\sigma})$ is computed using distance $R_{xy\sigma}$ and angle σ, as follows:

$$x_{ref_\sigma} = R_{xy\sigma} \cdot \cos(\sigma)\ ;\ y_{ref_\sigma} = R_{xy\sigma} \cdot \sin(\sigma)$$

Then, this point is transformed by a rotation matrix, containing the three axes (X, Y and Z) and the three rotation angles (*pitch*, *roll* and *yaw*), which is presented on Equation 3.1. As mentioned before, this step is only performed for *pitch* and *roll* variations, thus the transformations are performed using zero value for *yaw*. Also, *pitch* and *roll* angles used in these transformations are not absolute, i.e. they are given by the variation between the current angle and the angle used for the previous scan of the LADAR.

The transformed point will then be obtained by the product between the rotation matrix of Equation 3.1 and the point (x_{ref_σ}, y_{ref_σ}, 0), as follows:

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} =$$

$$\begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \sin\alpha\sin\gamma + \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \cdot \begin{bmatrix} x_{ref_\sigma} \\ y_{ref_\sigma} \\ 0 \end{bmatrix}$$

, where α, ß and γ represent, respectively, *yaw*, *pitch* and *roll* angles.

The next step consists of finding the line segment between the transformed point (x_t, y_t, z_t) and the point where the LADAR lens is placed $(0,0,h)$ and its intersection with the *XY* plane. The point where this line segment intersects the *XY* plane (assumed to be the ground plane) will then be used to compute the new distance for the *reference plane*. The calculation of this line segment is performed for two planes – *ZY* and *ZX* – and uses the characteristic equation of a line which is given by:

$$z = m_{zy} \cdot y + b \quad \text{, for plane } ZY;$$
$$z = m_{zx} \cdot x + b \quad \text{, for plane } ZX;$$

In these equations, m_{zy} and m_{zx} are the slopes of the line segments and b represents the y-intercepts. In this case, the slopes and y-intercepts are computed as follows:

$$m_{zy} = \frac{(z_t - h)}{y_t}$$

$$m_{zx} = \frac{(z_t - h)}{x_t}$$

$$b = h$$

The intersection of each line segment with the *XY* plane is computed through the following equations:

$$z = 0 \leftrightarrow m_{zy} \cdot y + b = 0 \leftrightarrow y_{int} = -\frac{b}{m_{zy}} = \frac{h \cdot y_t}{(z_t - h)}$$

$$z = 0 \leftrightarrow m_{zx} \cdot x + b = 0 \leftrightarrow x_{int} = -\frac{b}{m_{zx}} = \frac{h \cdot x_t}{(z_t - h)}$$

The point $(x_{int}, y_{int}, 0)$ is then used to compute the new distance for the *reference plane*. This distance will be given by the calculation of the Euclidean distance between point $(x_{int}, y_{int}, 0)$ and point $(0,0,h)$ as follows:

$$new_{dist(\sigma)} = \sqrt{(x_{int} - 0)^2 + (y_{int} - 0)^2 + (0 - h)^2}$$

This last step can be easily understood with the help of an example of a *pitch* variation presented in Figure 4.12. Point B is obtained by computing a 3D transformation of point A with the matrix presented in Equation 3.1. But, to have a new *reference plane* that is consistent with real LADAR measurements, it's necessary to know point C, because the ground plane (*XY* plane) is often the "lower limit" of the distances returned by the LADAR (laser beams can't "drill" through the ground). Knowing points B and D (considered to be the point of coordinates (0, 0, *h*)) it's possible to find point C, as explained above.

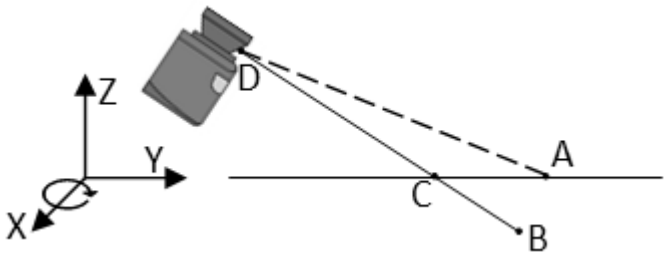


Figure 4.12 – Example of point transformation.

The new distance that will appear on the *reference plane* is the Euclidean distance between points C and D. It is worth mentioning that, in some cases such as *roll* variations, some transformed points have bigger *z* coordinates than the *z* coordinate of point D. In these situations, the previous operations are not performed and the new distance of the *reference plane* will be the maximum distance that the LADAR can measure. The procedure explained so far is performed for each distance of the *reference plane* previously stored (the one that was updated the last time the procedure of Figure 4.9 was carried out) and only for variations in *pitch* and *roll*, as stated before. As previously explained (Figure 4.9), *Yaw compensation* procedure is called to action right before the scrolling procedure but, however, these two procedures are not always carried out together, i.e. whereas the scrolling procedure is always performed when the displacement of the LADAR is bigger than the size of a cell, *Yaw compensation* is carried out only when necessary. For example, when the robot is driving straight ahead *Yaw compensation* will not be performed. However, when both procedures are performed simultaneously, *Yaw compensation* is always made first and, after that, the scrolling procedure is done, i.e. first the map is rotated (*Yaw compensation*) and, subsequently, cells are transferred across the Y axis of the map (scrolling).

On *Yaw compensation*, the *reference plane* is not modified and the compensation is performed on the *elevation map*, i.e. the map is rotated according to the variation of *yaw* angle, because it must maintain one of its main characteristics, which is being LADAR-centered. Just as on *Pitch-Roll compensation*, the *yaw* angle used in the rotation of the map is not absolute, i.e. it is given by the variation between the current angle and the angle used for the last time that *Yaw compensation* was carried out.

The timing for the rotation of the map depends on the size of its cells, otherwise there will be occasions when the variation of the *yaw* angle is not sufficient for cells to change position inside the map. Thus, for the map to be actually rotated, *Yaw compensation* should be made only when the angle corresponding to the variation of *yaw* is such that the XY points

corresponding to each cell change significantly their position inside the map when rotated with this angle, i.e. when the rotation of a XY point corresponding to a cell allows this point to be moved to another cell. Thus, the system follows the next procedure in order to decide when *Yaw compensation* is ready to be made: first, it searches for the first cell of the central column of the map (i.e. the central column's cell that is closest to the LADAR) that has information. Then, this cell is converted to an XY point which is rotated by the rotation matrix of Equation 3.1 with α = -yaw, and ß and γ with zero value. Finally the distances between the X and Y coordinates of the original point and the corresponding X and Y coordinates of the transformed point are computed and, if one of these distances is bigger than the size of a cell, then the information contained by this cell is ready to be moved to a different cell. In this model, it is assumed that, if the cell of the central column of the map that is closest to the robot is ready to be rotated, then so is the rest of the map. The rotation of the whole map is achieved by the following procedure: each cell of the map is converted to an XY point that is rotated by the rotation matrix of Equation 3.1 with α = -yaw, and ß and γ with zero value. Then, the information present in the original cell is copied to a new cell that corresponds to the transformed XY point. Finally the information that remained in the original cell is erased.

In summary, *pitch* and *roll* compensation consists of continuously updating the distances of the *reference plane* so that the comparison with the *real plane* can be reliable, whereas *Yaw compensation* is responsible for the rotation of the *elevation map* in order to maintain this map centered on the LADAR.

As aforementioned, this whole mapping and obstacle detection system is based on the assumption that the robot that carries the LADAR stands on a flat surface, which is considered to be the ground plane. Therefore, a drawback for this system happens when the ground plane changes its orientation, as for example when the robot tries to go down a ramp. In this situation, *Pitch-Roll compensation* will not help because it will try to adjust the *reference plane* when it is not supposed, since the attitude variation that affected the LADAR was caused by a modification on the orientation of the ground plane and not by an object through which the robot was passing. This can cause the emergence of false positives on obstacle detection and, therefore, an additional heuristic is necessary to prevent these problems and to make this system more robust to be used on navigation and path planning, since it is mainly in this area that this system can be useful. Thus, each time the variation over time in LADAR's orientation, since the start of its operation, reaches a significant value, i.e. whenever the total variation of the *pitch* angle exceeds a threshold ε, the *reference plane* is recalculated as explained on section 4.2.1 and the *elevation* and *obstacle* maps are erased.

Maps are erased in order to delete all the erroneous elevation information obtained and all the false obstacles detected during the transition between a ground plane and the other, and the *reference plane* is recalculated to prevent that fictional obstacles show up in the next scans. However, this is only performed in those cases where the *pitch* angle didn't vary abruptly since the last iteration (i.e. when *pitch* variation since last scan didn't exceed a certain threshold η), since this is what happens when the robot's wheels pass through an object. This second condition has to be verified, so that the first condition doesn't interfere with the *Pitch-Roll compensation* by giving order to clear the maps and re-compute *the reference plane* in situations where variations on robot's attitude happened due to the passage of the robot through some object and not due to variations on the orientation of the ground plane.

4.7 Implementation on Player Server

As stated on section 1.2 of Chapter 1, this model was implemented on a framework for mobile robotics applications named *Player/Stage Project*. Section 3.2 of Chapter 3 has further information about this framework and the concepts that will be addressed in this section.

In particular, this model was incorporated on a component of this framework, named the *Player* Server. To achieve this, it was developed an *abstract driver* to perform the tasks related to this mapping and obstacle detection system and a *driver* to acquire measures from a LADAR.

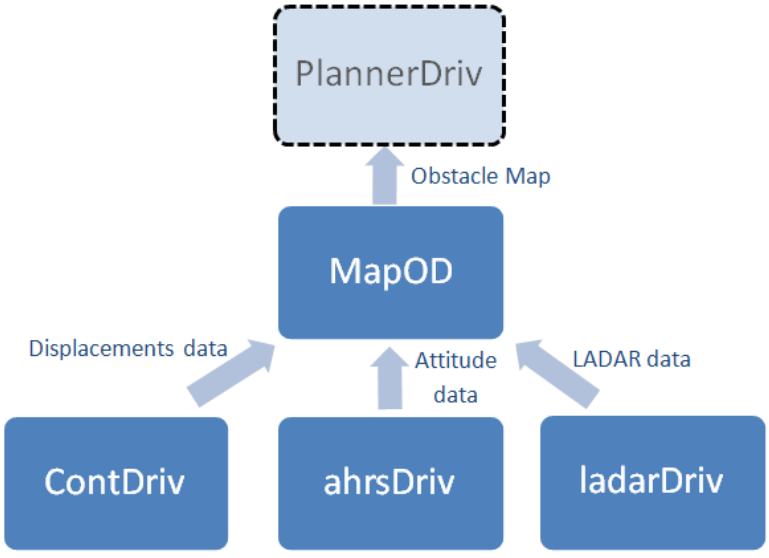


Figure 4.13 - Implementation of this system on Player Server

Figure 4.13 illustrates the hierarchy of the *Player* server architecture used and where the developed *drivers* fit into this architecture. The components of this architecture that were developed for this work are the *ladarDriv* and *MapOD drivers*.

The *ContDriv* and *ahrsDriv drivers* were not developed specifically for this work but they are also used to provide their data to *MapOD*. These two *drivers* are responsible for the computation of robot's displacements through the odometry data provided by two optical encoders (*ContDriv*) and for the acquisition of data from the AHRS sensor (*ahrsDriv*).

The *driver ladarDriv* is responsible for the communication with the LADAR sensor and provides the acquired measurements to the *MapOD driver*. Its operation is based on a simple sequence of actions (Figure 4.14): when it's launched, it reads some configuration parameters of the LADAR (measurement frequency, angular step, IP address, port, etc.) from the *config file* and connects to the LADAR on the *Constructor* method. Then, every *x* milliseconds the *Main* loop is executed. Here *ladarDriv* gives orders to the LADAR to perform a scan, receives the set of distances given by this sensor, adapts this information to the *laser interface* format and provides it to the drivers that may use it, in this case the *MapOD driver*. This *driver* can also receive requests to send the current configuration parameters of the LADAR (scanning frequency, angular step, etc), through the *ProcessMessages()* method.

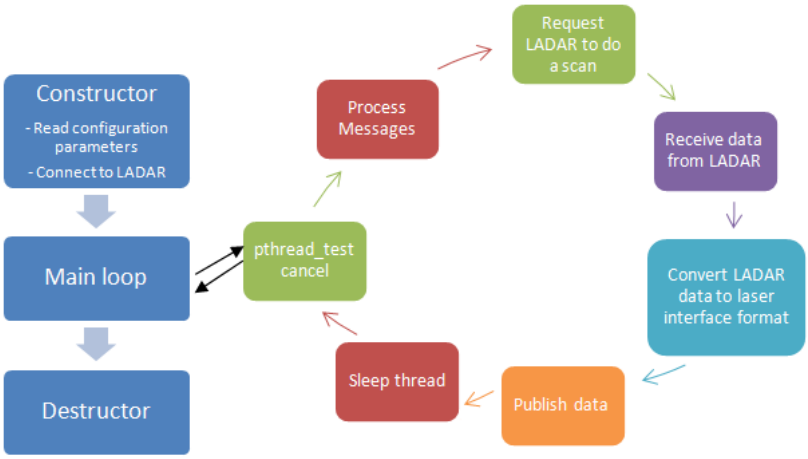


Figure 4.14 – Run-time process of *ladarDriv* driver.

The *MapOD driver* is responsible for all the mapping and obstacle detection system and, thus, it needs to receive measures from the LADAR and information about robot's attitude (AHRS) and robot's displacements (Optical Encoders). This *driver* provides an *elevation map* and also an *obstacle map*, which is more suitable for a Planner that may exist on the

architecture. Therefore, Figure 4.13 shows *MapOD* providing an *obstacle map* to an *abstract driver* that represents a Planner (*PlannerDriv driver*). *MapOD* starts its operation by reading from the *config file* some parameters for the maps (resolution, width and height) and the identification numbers of the drivers *ContDriv*, *ahrsDriv* and *ladarDriv* (Figure 4.15). The driver *MapOD* needs these identification numbers in order to establish connections with the corresponding *drivers* through the *Player Server*, so that it can receive data from them. These connections are established in the *MainSetup()* method. The main action of this *driver* is present on the *ProcessMessages()* method. Here, it receives LADAR measures from *ladarDriv* and attitude and displacements data from *ahrsDriv* and *ContDriv*, respectively. Whenever a message from *ladarDriv* is received by *MapOD*, the whole mapping and obstacle detection system illustrated on Figure 4.9 is executed. Each map obtained (*elevation map* and *obstacle map*) is provided in tiles (as specified by the *map interface*), as soon as *MapOD* receives the corresponding request on the *ProcessMessages()* method.

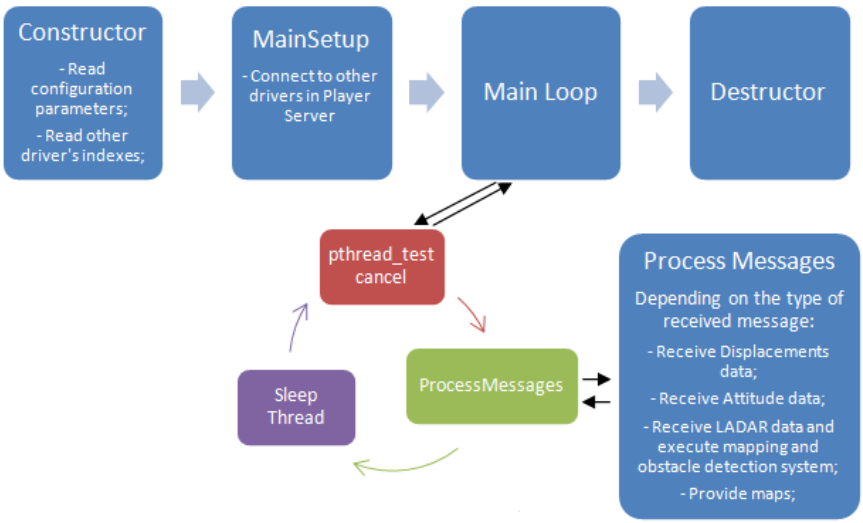
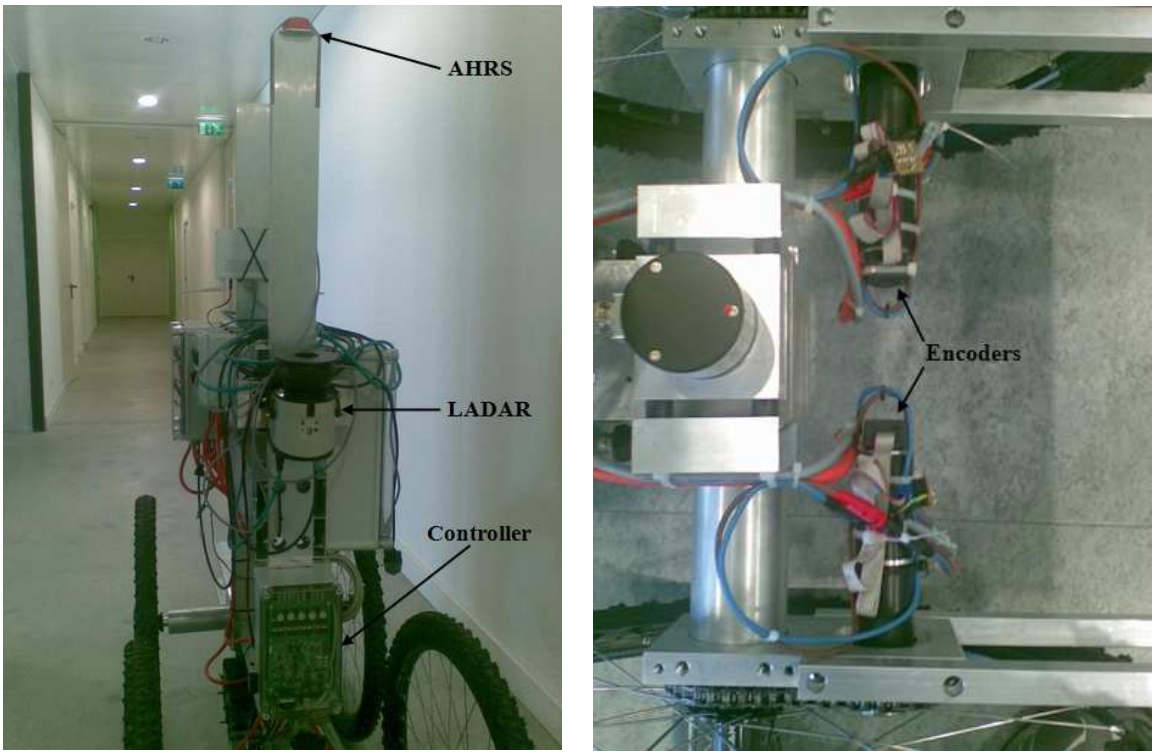


Figure 4.15 - Run-time process of *MapOD* driver.

Besides the developing of these *drivers*, it was also created a *Client* program that uses some *Proxies* to interact with the corresponding *drivers*. This *Client* allows the user to view the maps in real-time and also, if desired, to control the movements of the robot through the computer's keyboard. Its operation is quite simple and basically consists of periodically requesting (through the *MapProxy*) the *elevation* and *obstacle maps* to the *MapOD driver* and display them using OpenGL [Shreiner et al., 2007].

5. Experimental Results

This chapter presents a set of experiments that were carried out in order to demonstrate the capabilities and weaknesses of the developed mapping and obstacle detection system and its applicability to indoor/outdoor structured environments. This system was developed to be part of a service robot that is being developed by Holos, S.A. The experiments were conducted in real environments and with the developed system incorporated on this service robot, which is illustrated on Figure 5.1.



(a) (b)

Figure 5.1 – The robot with which these experiments were performed. This robot is being developed by Holos, S.A., the mapping and obstacle detection system being the work of the author.

This robot carries several sensors, including a LADAR which is the main sensor used in this system. It also has an AHRS sensor and two optical incremental encoders, which are also necessary for full operation of the developed system. In the context of these experiments, the AHRS sensor is used to obtain *roll*, *pitch* and *yaw* angles and the encoders are a means for capturing the distance travelled by the robot.

The LADAR used in these experiments is a SICK LMS111 which has a maximum field of view of 270° and a scanning range up to 20 meters. It can be configured to scan with angular steps of 0.25° or 0.5° and with a measurement frequency of 25 or 50 Hz [SICK, 2008]. The communication is achieved through Ethernet connection.

The AHRS sensor is an Xsens Mti that is composed by an accelerometer, a gyroscope and a magnetometer, all with three axes [Xsens, 2008]. The communication protocol is RS-232 and it has an RS-232/USB converter. The Xsens Mti can handle 5G accelerations and has a rate of turn of, approximately, 300º/s.

The optical incremental encoders have 500 *ppr* (points per revolution) and output pulses as they rotate. By counting these pulses it is possible to obtain how many revolutions (or fractions of) the motor has turned and thereby compute the displacement of the robot. These encoders are connected to two 150W Maxon motors and the number of pulses of each encoder is provided by a Roboteq AX3500 motor controller [Roboteq, 2007].

If nothing is said otherwise, in the following experiments the LADAR is configured with a field of view θ = 120º (between 30º and 150º), a measurement frequency of 25 Hz and an angular step α of 0.25º. It is placed on the top of the robot at an approximate height h of 1.08 meters and with a tilt angle of approximately 7.5 degrees, i.e. ϕ is set to 82.5 degrees.

The widths, heights and cell's sizes of the *elevation* and *obstacle* maps have also been set to the values presented on Table 5.1, if nothing is said otherwise.

Name	Description	Value
m	Number of width cells of the *elevation map*	101
n	Number of height cells of the *elevation map*	101
c_w	Width of a cell of the *elevation map*, measured in meters	0.2
c_h	Height of a cell of the *elevation map*, measured in meters	0.2
m_{obs}	Number of width cells of the *obstacle map*	101
n_{obs}	Number of height cells of the *obstacle map*	101
Cw_{obs}	Width of a cell of the *obstacle map*, measured in meters	0.2
Ch_{obs}	Height of a cell of the *obstacle map*, measured in meters	0.2

Table 5.1 - Widths, heights and cell's sizes of the *elevation* and *obstacle* maps.

Therefore, the *obstacle* and *elevation maps* have a range of approximately 20x20 meters. H_{pos} and H_{neg} have been set to +10 cm and -10 cm, respectively.

As described in the previous chapter, this system was implemented on the *Player Server* and a *Client* program was developed to display the results of the experiments. Both *Player Server* and *Client* run on the operating system Linux with Ubuntu distribution. The *Player drivers* responsible for all the mapping and obstacle detection system, as well as the *Client* program, have been implemented in C++. The *Player Server* runs on the robot's side, on a ZOTAC GeForce 9300 – ITX with an Intel Q9650 Core 2 Quad 3 GHz and 4GB RAM. The *Client* runs on a laptop equipped with an Intel Core 2 Duo 1.66 GHz processor and 1GB RAM.

The results of all experiments are presented by snapshots or measurements taken on the *Client* side. In these experiments, in order to allow the reader a better visualization and understanding of the results, each type of cell content is drawn in a different color. Therefore, and if nothing is said otherwise, cells from the *elevation map* with average heights between -3 and +3 centimeters are considered as belonging to the ground plane and thus are drawn in blue. Also, cells that are below the *ground plane* are drawn in yellow and cells that are above the *ground plane* are drawn in black and grey. In the *obstacle map*, cells considered as freespace are drawn in green and cells considered as obstacles are drawn in red. The cell drawn in orange in each map represents LADAR's position. Moreover, every map image contains a line segment of 1 meter length (also drawn in orange) that intends to be a scale for the map in order to enable the reader a better perception of the distances between the robot and the obstacles and, also, of the displacements of the robot.

5.1 Changing resolution and size of maps

This experiment intends to demonstrate the ability to customize the size and resolution of *elevation* and *obstacle maps* and the effects of this customization on the detail of the information presented in these maps and also its impact on the algorithms that use these maps.

As explained on chapter 4, both maps built by this system are represented by two-dimensional arrays of cells. Therefore, the number of cells of each map is always given by $m \times n$, in the *elevation map*'s case, and by $m_{obs} \times n_{obs}$ in the *obstacle map*'s case. The amount of memory occupied by these maps and the processing time of an algorithm that uses these maps depend on their size (number of cells). For example, in order to draw the *elevation map* on OpenGL, the *Player Client* runs through every cell of the map. Thus, the number of iterations of the drawing algorithm can be given by the number of cells of the map. Table 5.2 represents this number of iterations for four different map sizes.

Map width, *m*	Map height, *n*	Number of iterations, $it = m \times n$
41	41	1681
101	101	10201
201	201	40401
401	401	160801

Table 5.2 – Number of iterations for the algorithm that draws the maps with four different map sizes.

By observing Table 5.2 it is clear that the larger the map, the greater the number of iterations of the drawing algorithm. The processing time of this drawing algorithm is closely connected with the number of iterations needed to complete the algorithm.

As aforementioned, the amount of memory occupied by these maps also depends on their size. Considering that the content of each cell of the *elevation map* occupies 4 bytes of memory and the content of each cell of the *obstacle map* occupies 1 byte of memory and that both maps have the same size, i.e. $m_{obs} = n_{obs} = m = n$, Table 5.3 shows the number of bytes occupied by each map with four different map sizes.

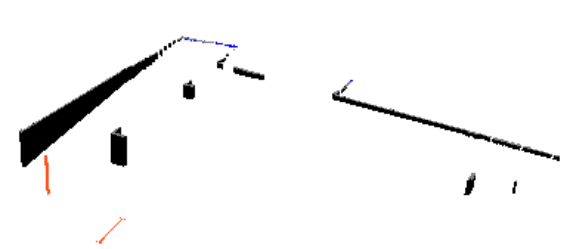
Map width, *m*	Map height, *n*	Memory occupied by elevation map $= m \times n \times 4$ (bytes)	Memory occupied by obstacle map $= m \times n \times 1$ (bytes)
41	41	6724	1681
101	101	40804	10201
201	201	161604	40401
401	401	643204	160801

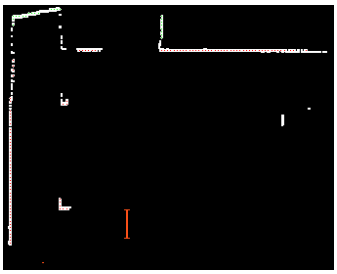
Table 5.3 – Amount of memory occupied by the elevation and obstacle maps with four different map sizes.

The values presented in Table 5.3 are also in accordance with what was said previously, i.e. as the map size increases, the amount of memory used also increases.

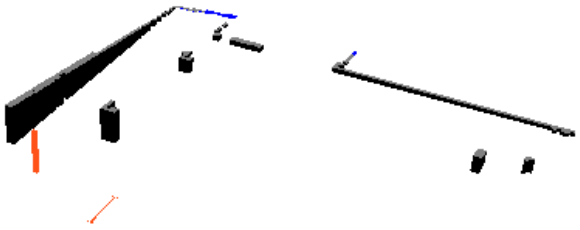
In order to evaluate the effects of the configuration of these maps on the detail of the information, it was performed an experiment where it was taken a set of *elevation maps*, and the corresponding *obstacle maps*, representing the same scene - the hall from Figure 5.2 (a) - but with different sizes and resolutions. All of these maps have a range of approximately 20 x 20 meters but each one achieves this range with a certain size and resolution.

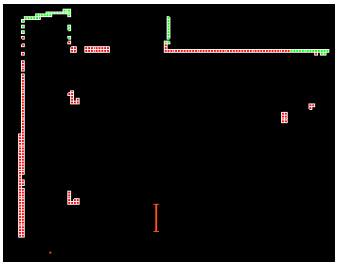
(a)



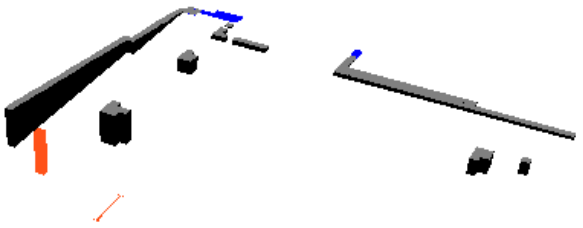


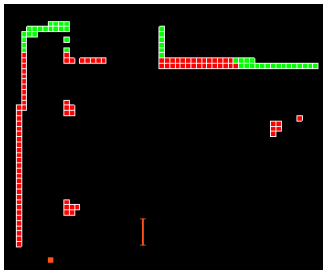
(b) (c)



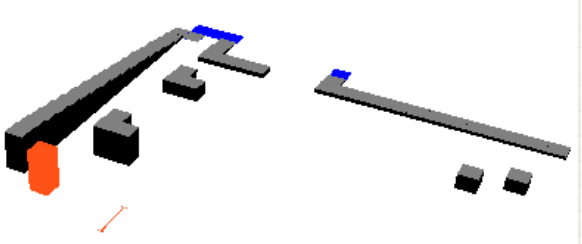


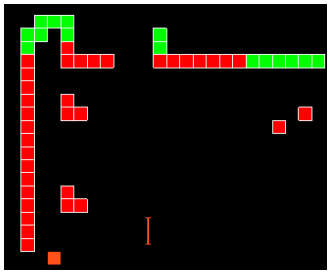
(d) (e)





(f) (g)





(h) (i)

Figure 5.2 – Elevation and obstacle maps with different sizes and resolutions, all built in the same hall. (a) – real image of the hall. (b) and (c) – elevation and obstacle maps with 401x401 cells and resolution of 0.05 m. (d) and (e) – elevation and obstacle maps with 201x201 cells and resolution of 0.1 m. (f) and (g) – elevation and obstacle maps with 101x101 cells and resolution of 0.2 m. (h) and (i) – elevation and obstacle maps with 41x41 cells and resolution of 0.5 m.

Elevation maps are illustrated in the left column of Figure 5.2, from images (b) to (h), by increasing order of cell's size and descending order of number of cells. The corresponding *obstacle maps* are represented in the right column of Figure 5.2.

By observing Figure 5.2 one can clearly see that resolution decreases as the size of the cells increases. On the one hand, the map should have small cells in order to present a good resolution because in this way the area occupied by the objects and its position relative to the LADAR is more consistent with the real scene. On the other hand, if the map has small cells it is necessary to have a big number of cells in order to maintain the range of the map and this can be computationally heavy, as discussed earlier in this section. In fact, in the *elevation* and *obstacle maps* of Figure 5.2 (b) and (c) there is a good accuracy on the information as can be seen, for example, by the distance between the first pillar and the left wall that is consistent with Figure 5.2 (a) (approximately 1.7 meters). However, these maps have a large number of cells which can make them too heavy to be used by this mapping and obstacle detection system or to be transferred by its *Player driver* to some other *drivers* of the architecture, such as a Planner *driver*, or to a *Client* that wants to draw the maps. The *elevation* and *obstacle maps* of Figure 5.2 (h) and (i) have a much smaller size but the resolution decreases a lot and the information's accuracy is affected, as shown, once again, by the distance between the first pillar and the left wall, which is now 1 meter in these maps. The maps from Figure 5.2 (d), (e), (f) and (g) can achieve a much better accuracy than the ones from Figure 5.2 (h) and (i) presenting a much smaller size than the ones from Figure 5.2 (b) and (c).

Another experiment was made to measure the processing time of the algorithm that draws the *elevation maps* of Figure 5.2. Table 5.4 represents the results of this experiment.

Map width (cells), *m*	Map height (cells), *n*	Resolution (meters)	Processing time (milliseconds)
41	41	0.5	0.863
101	101	0.2	4.838
201	201	0.1	24.095
401	401	0.05	85.074

Table 5.4 – Processing time (on *Client* side) for the algorithm that draws the elevation map with four different map sizes.

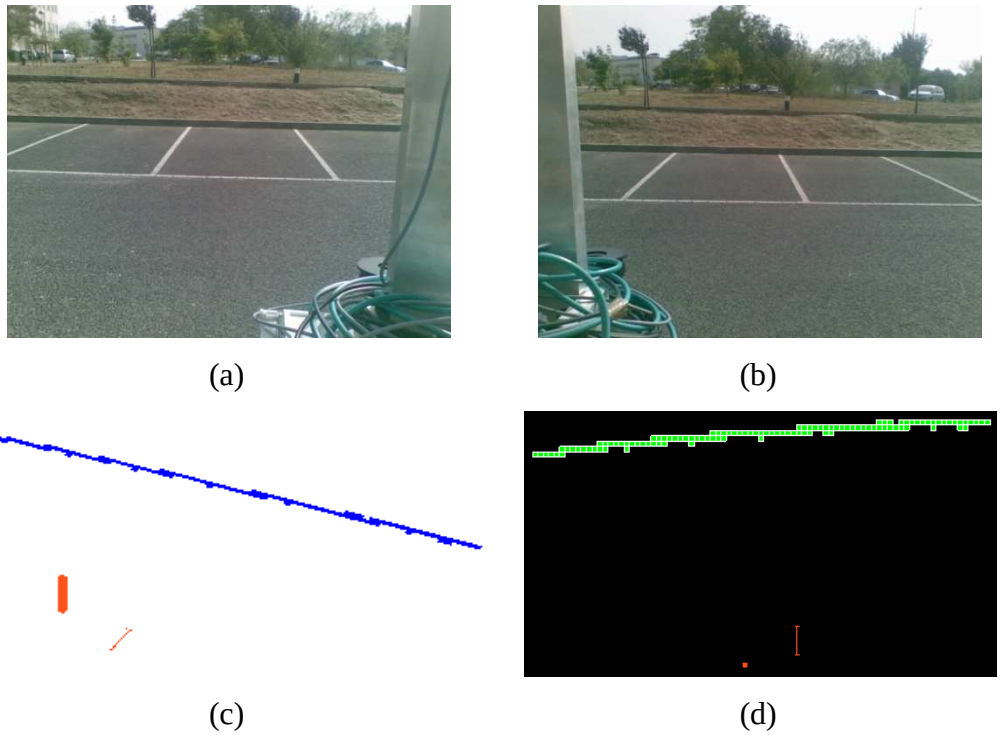
The results of Table 5.4 show that, as expected, the processing time of the algorithm increases with the size of the map. As also expected, these results are consistent with the information from Table 5.2.

In this experiment, the maps that show more balance between the accuracy of the information and the computational load are the ones presented on Figure 5.2 (d), (e), (f) and (g) which have a resolution of 0.1 meters and a size of 201x201 cells ((d) and (e)) and 0.2 meters of resolution and a size of 101x101 cells ((f) and (g)). These maps have good accuracy and they don't imply a very high computational load. Therefore, in all the remaining experiments of this dissertation, the maps will have one of these two configurations.

5.2 The reference plane

The purpose of this experiment is to prove the correct computation of the *reference plane*. As explained on chapter 4, the *reference plane* is composed by a set of ranges which the LADAR, positioned as on Figure 4.3, would provide if it only scans the ground plane and no other object or surface is detected inside its range and field of view. The *reference plane* is very important for the creation of the *elevation map* because it is used on the computation of the heights of each cell by being compared to the *real plane* provided by the LADAR.

In this experiment the robot was positioned in a place wide enough so that the LADAR can "catch" the ground plane in its entire field of view. This place can be seen on Figure 5.3 (a) and (b).



(a) (b)

(c) (d)

Figure 5.3 – Reference plane experiment. (a) and (b) – real scene. (c) – elevation map. (d) – obstacle map.

The field of view of the LADAR, θ, was configured with an angle of 90 degrees (between 45 and 135 degrees) in this experiment. Also, the limits among which it is considered that a cell belongs to the ground plane were extended to average heights between -6 and +6 centimeters.

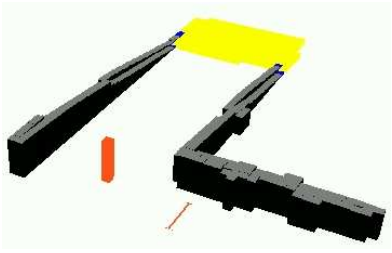
If the *reference plane* is correct, the *elevation map* should only show cells from the ground plane, because, at each angle where a beam is emitted, both distances from the *reference plane* and the *real plane* have similar values and the computed heights should be close to zero. As expected, the *elevation map* from Figure 5.3 (c) presents only blue cells, which together approximately make up a horizontal line. This line represents the portion of the ground plane which is detected by the LADAR. The *obstacle map* (Figure 5.3 (d)) is also in agreement considering all cells as freespace.

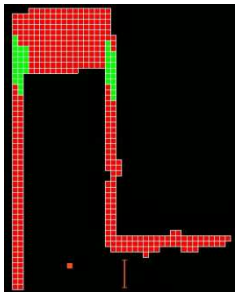
5.3 Changing orientation of ground plane

As explained on section 4.6, situations where the ground plane changes its orientation bring problems to this mapping and obstacle detection system. An additional heuristic that uses two thresholds for the *pitch* angle (also explained on section 4.6) was implemented to try to solve these problems. The purpose of this experiment is to evaluate the usefulness of this additional heuristic, placing the robot in a situation where the orientation of the ground plane is altered. In this experiment the robot is asked to travel straight on towards north (top of maps) at a speed of 0.2 ms^{-1}. During its path, the robot will descend a ramp, which is illustrated in Figure 5.4 (a). Prior to the experiment, the threshold for *pitch* total variation ε was set to 0.5 degrees and the threshold for *pitch* variation relative to the last scan η was set to 0.1 degrees. The snapshot of Figure 5.4 (b) and (c) was taken before the robot start to go down the ramp and the snapshot of Figure 5.4 (d) and (e) was taken at a time when the robot was already walking down the ramp.

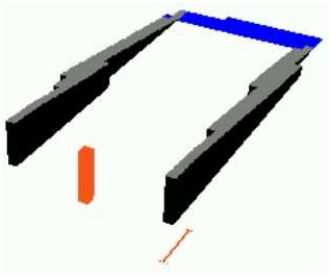
In the *elevation map* of Figure 5.4 (b) the main problem caused by changes on the orientation of the ground plane is evident. In this map there are a large number of yellow cells, which are cells with negative average heights. This is not incorrect because, in fact, the points where LADAR's beams touch these surfaces are below the ground plane where the robot is standing. However, this will mean that many of these cells are regarded as obstacles on the *obstacle map* (Figure 5.4 (c)), leaving the robot in a dead-end situation. Thus, it becomes necessary that, when the robot is on the ramp, this situation can be corrected.

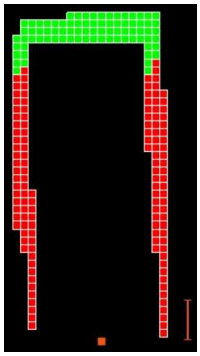
(a)





(b) (c)





(d) (e)

Figure 5.4 – Experiment in which the robot goes down a ramp. (a) – real scene representing the ramp. (b) and (c) – elevation and obstacle maps taken before the robot start to go down the ramp. (d) and (e) - elevation and obstacle maps taken when the robot was already going down the ramp.

During the passage of the robot to the ramp, the total variation of the *pitch* angle reaches the threshold ε because, in the ramp, the robot presents a *pitch* angle of +1 degree. Therefore, during this transitional period the *reference plane* is recalculated and the two maps are erased and, once on the ramp, the robot builds a new *elevation map* that is much more consistent with the real scene, as can be seen on Figure 5.4 (d). As a consequence, the *obstacle map* is now much more realistic because it has several traversable cells that correspond mostly to the portion of ground plane "swept" by the LADAR. Unlike the former, this *obstacle map* is now able to be provided to a planner to be used for path planning or navigation and this was the main concern that led to the incorporation, in this system, of this heuristic that deals with variations on the orientation of the ground plane.

5.4 Map scrolling

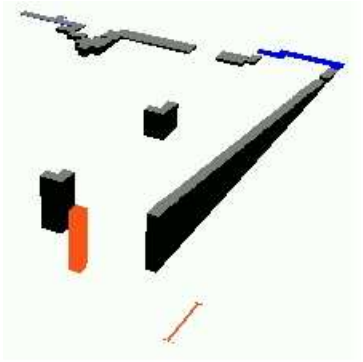
This section exposes the results of some experiments carried out to evaluate the performance of the scrolling procedure. As mentioned on chapter 4, this procedure is executed only when robot's displacement is greater than the size of a cell of the map, i.e. each time the displacement of the robot reaches the size of a cell, the map is scrolled by one cell. In these experiments, the measures of two optical incremental encoders are used to compute the displacements of each front wheel of the robot and, after that, it is computed the average of these two displacements. Since the robot doesn't have any sensor that provides its curvature angle, it is not possible to compute the displacement of the central point of the front axle of the robot through kinematic equations. Therefore, it is assumed that the average of the displacements of each wheel corresponds to the displacement of the central point of the front axle. As explained on chapter 4, cells are only repositioned across the Y axis. Therefore, in order to decide if the scrolling procedure must be executed, the Y axis projection of the displacement of the central point is computed (using *yaw* angle) and this value is the one that is compared with the cell's size. If the robot is driving straight ahead, the displacements for the two wheels are almost equal, as also will be the displacement of the central point, which, in turn, will be approximately equal to its Y axis projection.

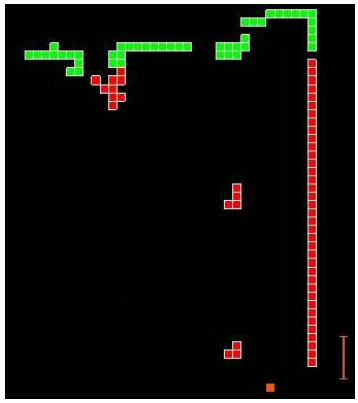
In these experiments the robot is asked to travel towards north (top of maps) at a speed of 0.2 ms^{-1}. It will be expected that the robot walks straight on, not making any turns, so that the scrolling procedure can be analyzed without the influence from Yaw compensation.

In the first experiment, the robot is asked to travel a small path with a total extent of 4.2 meters, which is a little more than the distance between the two pillars of Figure 5.5 (a). This figure shows images of the real scenes, and most of the data contained on *elevation* and *obstacle maps* captured by the LADAR at the beginning and at the end of this path. Prior to this experiment, the threshold for *pitch* total variation ε was set to 0.5 degrees and the threshold for *pitch* variation relative to the last scan η was set to 0.1 degrees.

By comparing the *elevation* and *obstacle* maps at the beginning of the path with the same maps at the end of the path, one can see that the distances between the various surfaces of the scene suffer a few changes, mainly due to inconsistencies between the scrolling procedure and the addition of new information to the maps. Despite that, the *elevation map* remains essentially consistent with the real scene.

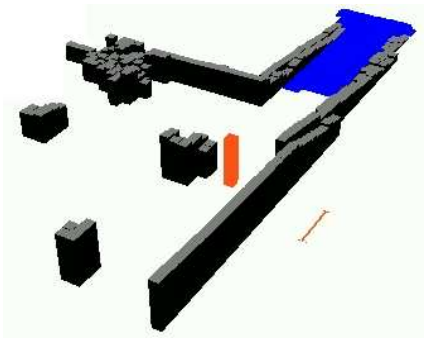
(a)

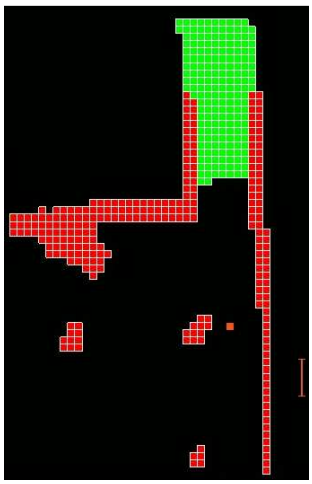




(b) (c)

(d)





(e) (f)

Figure 5.5 – Scrolling Procedure, first experiment. Beginning of the path: (a) – real scene, (b) – elevation map, (c) – obstacle map. End of the path: (d) – real scene, (e) - elevation map, (f) – obstacle map.

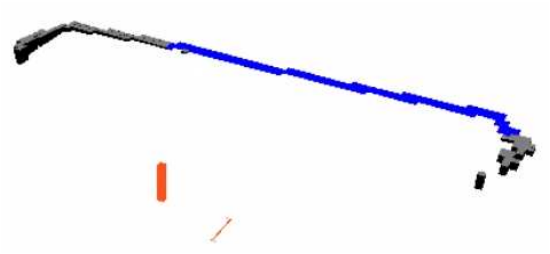
Moreover, the positioning of the robot in relation to the several existing surfaces is also in agreement with the real scenes, as can be seen on Figure 5.5 (b) through the position of the robot in relation to the first pillar at the beginning of the path, and also on Figure 5.5 (e) where one can see that, at the end of the path, the robot is positioned next to the second pillar, which is in line with the real scene of Figure 5.5 (d). There is also a small deviation on the right side wall due to the fact that the robot did not exactly travel in a straight line, i.e. there was a small deviation in its path and, because these cells have a considerable size (20x20 centimeters), this effect becomes more visible. The *obstacle maps* are consistent with the respective *elevation maps*, as can be seen by the *obstacle map* of Figure 5.5 (f) where the majority of cells considered as freespace belong to the portion of ground plane that was "swept" by the LADAR.

When the robot reached the end of the path, the displacement computed through the encoders had a value of 4.4 meters. Therefore, there is an error of 0.2 meters in relation to the total extent of the path (4.2 meters), i.e. in this experiment the total error of the encoders represents less than 5% of the total extent. However, as the reader will see in the next experiment, this error tends to increase over time, as the path increases its extension.

In the second experiment, the robot travelled a longer path between two marks drawn on the ground. The total extent of this path is 20 meters and, this time, the threshold for *pitch* total variation (ε) was set to 0.7 degrees. Also, the limits among which it is considered that a cell belongs to the ground plane were extended to average heights between -6 and +6 centimeters. Once again, Figure 5.6 shows images of the real scene, *elevation maps* and *obstacle maps* at the beginning and at the end of this path.

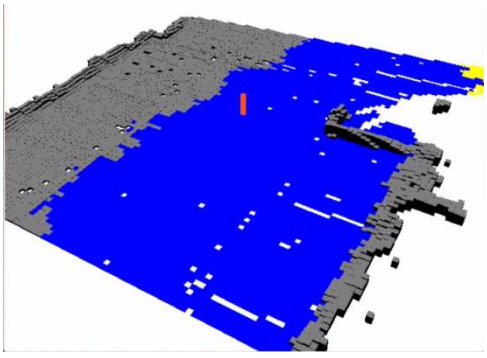
By looking at the *elevation maps* of Figure 5.6 there is one detail that gets our attention immediately. In these maps there is an area of grey cells (more visible on the map of Figure 5.6 (e)) on the left side, between the robot and the sidewalk at his left, where, at first, there should only be cells that belong to the ground plane (blue ones). Along this path the robot maintains a *roll* angle around -2 degrees and the *Pitch-Roll compensation* procedure continuously adjusts the *reference plane* due to variations on this angle and also on the *pitch* angle. However, the surface on the left side of the robot (where the parking lines are drawn) is not exactly on the same plane (it is slightly above) of the surface where the robot is standing. This means that, for each LADAR beam that touches this area, the real distance provided by the LADAR is smaller than the corresponding distance of the *reference plane*, because the latter is adjusted in relation to the surface where the robot is standing which, once again, has not exactly the same orientation of the surface on the left side of the robot.

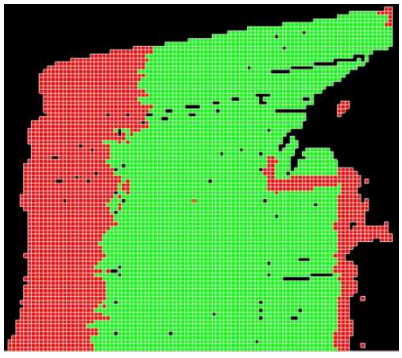
(a)



(b) (c)

(d)





(e) (f)

Figure 5.6 – Scrolling procedure, second experiment. Beginning of the path: (a) – real scene, (b) – elevation map, (c) – obstacle map. End of the path: (d) – real scene, (e) - elevation map, (f) – obstacle map.

The comparison between these two distances will give rise to cells with a reasonable average height. In fact, there is a slight gradient of elevation between the robot and the left sidewalk. This situation could pose a problem to the robot because the *obstacle map*, in line with *elevation map*, believes that most of this area is an obstacle. Nevertheless, this is not a serious problem because, if, by chance, the robot turned left towards the sidewalk, the threshold for *pitch* total variation (ε) would be exceeded (because, for the robot, the ground plane changes its orientation) and the maps would be erased and the *reference plane* recomputed. Thus, these obstacles, which in reality do not exist, would probably disappear.

Moreover, in the *elevation map* obtained at the end of the path it is possible to see that there are some cells that were left without information (white cells). This may be due to small errors in the discretization of LADAR's measures, i.e. when the elevation information computed based on LADAR's measures is added to the *elevation map*, and also to inconsistencies in *Yaw* compensation, which, in turn, should not be executed but, since the robot does not travel exactly in a straight line, there are always small deviations in its path and *Yaw compensation* turns out to be made, leading to small errors in the transfer of information between cells. The existence of these "empty" cells on the *elevation map* causes the corresponding cells of the *obstacle map* to be considered as "unknown" terrain.

Despite that, the *elevation map* remains essentially consistent with the real scene, as the majority of the cells belong to the ground plane. Also, the LADAR "catches" other surfaces on the right side of the robot, such as some vegetation and two vehicles which can be seen on Figure 5.6 (a). On the *elevation map* of Figure 5.6 (e) there are some blue cells behind the ones that represent the vehicle that is next to the robot. This is not incorrect because, the LADAR first detects the ground surface that is beneath the vehicle (represented by those blue cells) and, only when it moves a little forward is that it starts to detect the vehicle. The *obstacle maps* are consistent with the respective *elevation maps*, as can be seen by the *obstacle map* of Figure 5.6 (f) where the majority of cells considered as freespace belong to the large area of ground plane that was "swept" by the LADAR and other surfaces such as the two vehicles, the sidewalk on the left side, some vegetation on the right side and also a part of the ground plane on the left side of the robot (as explained before) are labeled as obstacles.

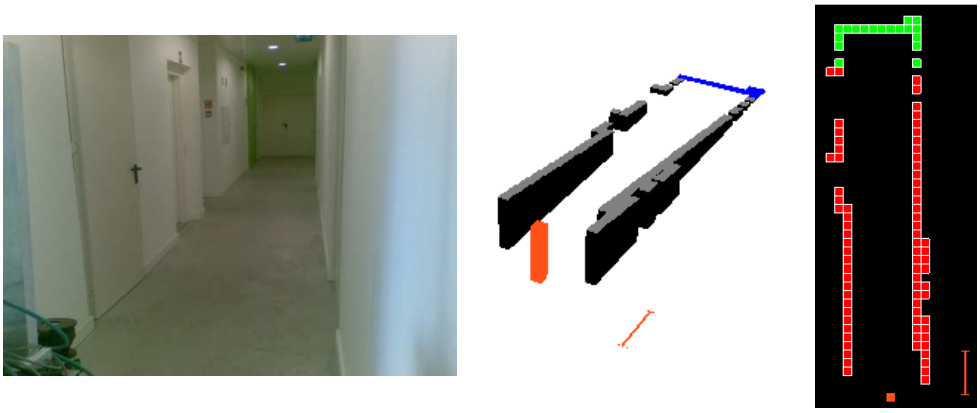
When the robot reached the end of the path, the displacement computed through the encoders had a value of 21.41 meters. Therefore, there is an error of 1.41 meters in relation to the total extent of the path (20 meters), i.e. in this experiment the total error of the encoders represents approximately 7% of the total extent of the path. Thus, as mentioned earlier in this section, there is an increase in the total error of the encoders as the distance travelled by the robot becomes larger. Finally, the positioning of the robot in relation to the several existing

surfaces is not totally in agreement with the real scenes, due to this error of the encoders. For example, on the real scene at the end of the path, although is not possible to see on Figure 5.6 (d), the vehicle is a little further away from the robot than the maps from Figure 5.6 (e) and (f) suggest.

5.5 Pitch-Roll compensation

This experiment was carried out to try to prove the adaptability of the mapping procedure to some changes in robot's attitude, so that the obstacle detection procedure can be more reliable. As explained on chapter 4, the main idea of *Pitch-Roll compensation* is to continuously adjust the *reference plane* to the current attitude of the LADAR, so that the comparison between the *reference plane* and *real plane* provided by the LADAR, which, in turn, is the basis for the creation of the map and the subsequent detection of obstacles, can be as much consistent as possible.

The first experiment of this section intends to simulate the passage of a robot through objects that interfere significantly in its orientation.

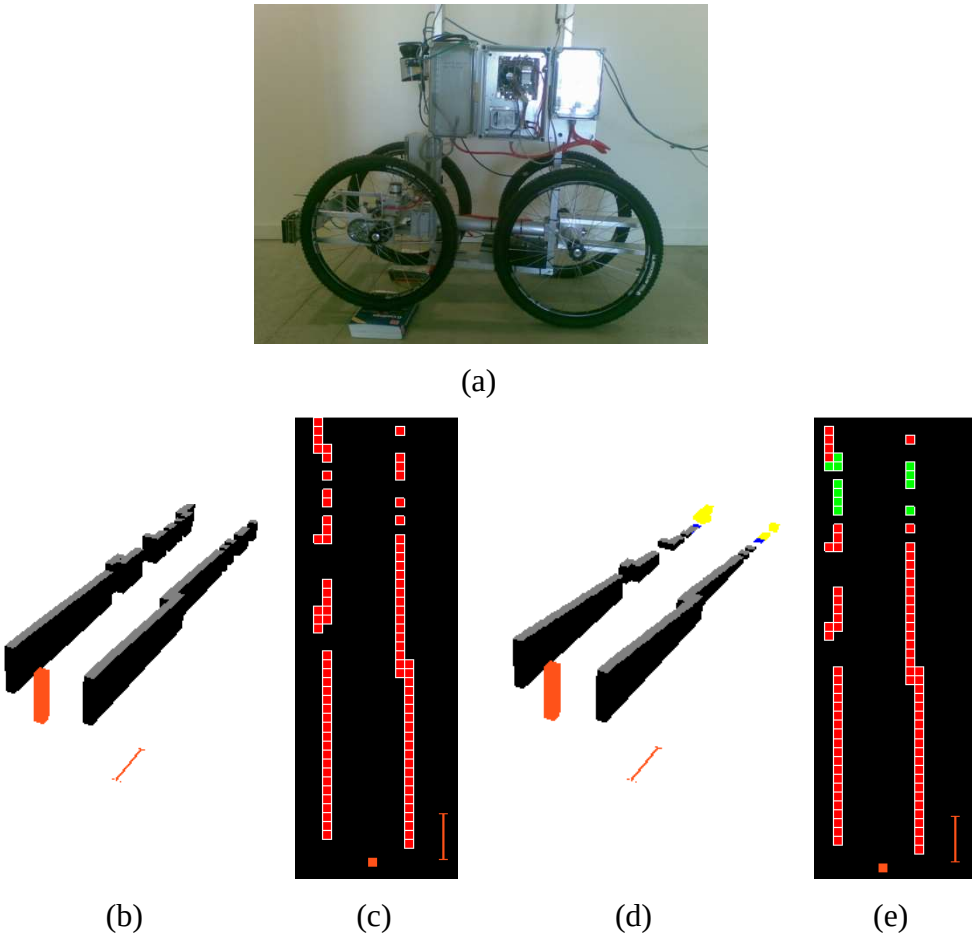


(a) (b) (c)

Figure 5.7 - Corridor where this experiment was performed. (a) is the real scene, (b) the elevation map and (c) the obstacle map.

Figure 5.7 represents the real scene where this experiment was performed as well as its *elevation* and *obstacle maps*. In this experiment, the threshold for *pitch* variation relative to the last scan η is set to 0.1 degrees and the threshold for *pitch* total variation ε was set to 0.5 degrees.

Figure 5.8, Figure 5.9 and Figure 5.10 represent three different cases of attitude changes caused by the objects through which the robot was passing. By passing through these objects the robot was positioned at, respectively, a *pitch* of -4.5 degrees, a *pitch* of 2.5 degrees and a *roll* of -5.2 degrees. In this experiment, each case is analyzed through maps built with and without *Pitch-Roll compensation*, so that the reader can have a better perception of the advantages of using *Pitch-Roll compensation*. Moreover, the reader should notice that the snapshots presented on the following figures of this section were taken after the maps have been erased and redrawn at the exact moment when the robot is on top of objects, so that the reader can clearly see the effects of this element on the whole mapping and obstacle detection system.



(a) (b) (c) (d) (e)

Figure 5.8 - Negative Pitch example. (a) – robot with negative pitch. (b) and (c) - elevation and obstacle maps with Pitch-Roll compensation. (d) and (e) - elevation and obstacle maps without Pitch-Roll compensation.

In the first case, the two front wheels of the robot are passing through the objects (Figure 5.8 (a)). In this situation, the attitude of the robot is affected and *pitch* has a negative value, which in this case is -4.5 degrees. With this positioning, the LADAR is able to “see” further

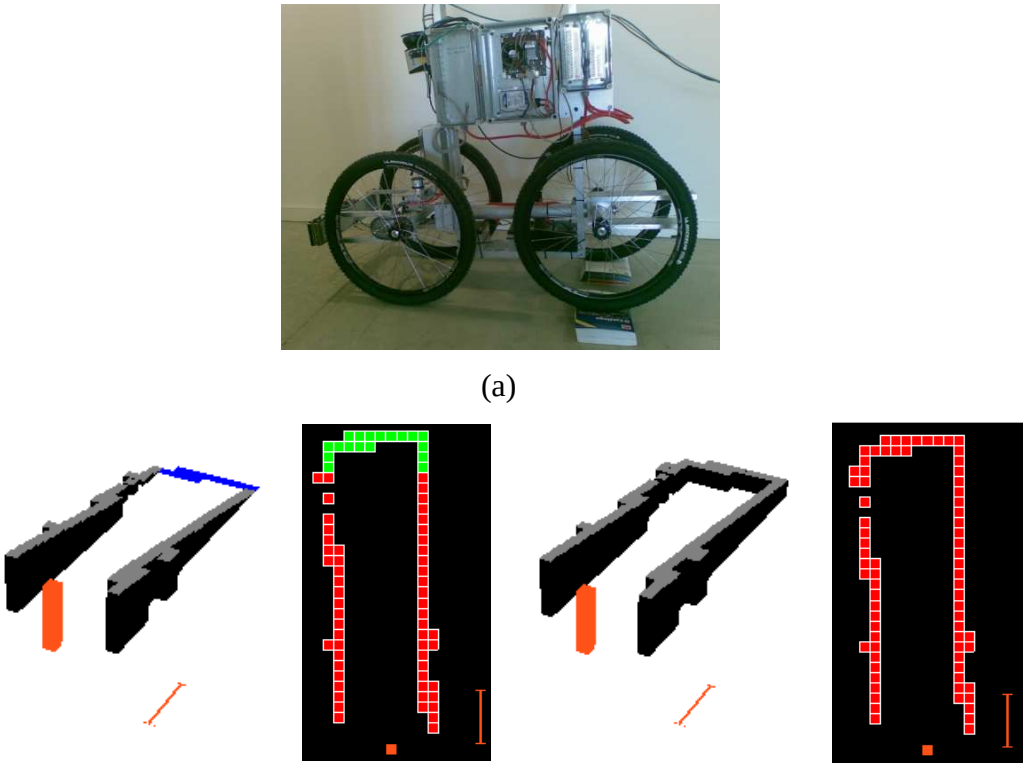
ahead and, consequently, its beams “touch” the ground few meters ahead and “touch” the walls a little higher. Therefore, if *Pitch-Roll compensation* correctly adapts the *reference plane* to the *real plane*, these changes should be reflected on the *elevation map*. By looking at Figure 5.8 (b), one can see that the heights of left and right walls increased in comparison to the *elevation map* of Figure 5.7 (b). The ground plane is not detected on this map because it is outside its range when the robot has this positioning. Thus, the map doesn’t show any blue cells. In line with the *elevation map*, the corresponding *obstacle map* of Figure 5.8 (c) only presents red cells, which means that all the detected surfaces are obstacles for the robot.

If the *elevation map* is built without *Pitch-Roll compensation*, the *reference plane* is not updated in order to be aligned with the *real plane* and this will origin the emergence of fictitious obstacles and errors on the computed heights of the detected surfaces. The *elevation map* of Figure 5.8 (d) presents some of these problems. The heights of left and right walls didn’t increase and some surfaces with negative heights (yellow cells) and some other surfaces considered to be from the ground plane (blue cells) appeared. In yellow cell’s case this happens because the distances from the *real plane* are bigger than the distances from the *reference plane*, due to the fact that the two planes are not aligned, and the computed height is negative. Negative and ground plane surfaces shouldn’t appear because, as mentioned earlier, at this positioning the LADAR cannot detect the ground plane inside the range of the map. As a consequence, the *obstacle map* of Figure 5.8 (e) considers some cells as freespace where, actually, it should only “see” obstacles.

In the second case, there is the reverse process. The two rear wheels of the robot are passing through the objects (Figure 5.9 (a)) and the attitude of the robot is also affected but, this time, *pitch* has a positive value of 2.5 degrees. With this positioning, LADAR’s range decreases when compared to Figure 5.7. Therefore, the ground plane should be detected closer to the robot. By looking at Figure 5.9 (b), one can confirm this circumstance because blue cells are drawn much closer to the robot. In line with the *elevation map*, the corresponding *obstacle map* of Figure 5.9 (c) has some green cells (freespace), mainly across the ground plane’s extension, considering the remaining cells as obstacles.

On the *elevation map* without *Pitch-Roll compensation*, once again the *reference plane* is not aligned with the *real plane* and the *elevation map* (Figure 5.9 (d)) presents a small wall where it should show the ground plane. This small wall is obtained because, across this set of angles, the distances of the *real plane* are much smaller than the distances of the *reference plane*, which leads to objects with positive and insurmountable heights for the robot. The *obstacle map* (Figure 5.9 (e)), in line with the *elevation map*, considers all the surfaces as

obstacles and puts the robot in a dead-end situation, forcing him to back off to avoid obstacles that do not actually exist.



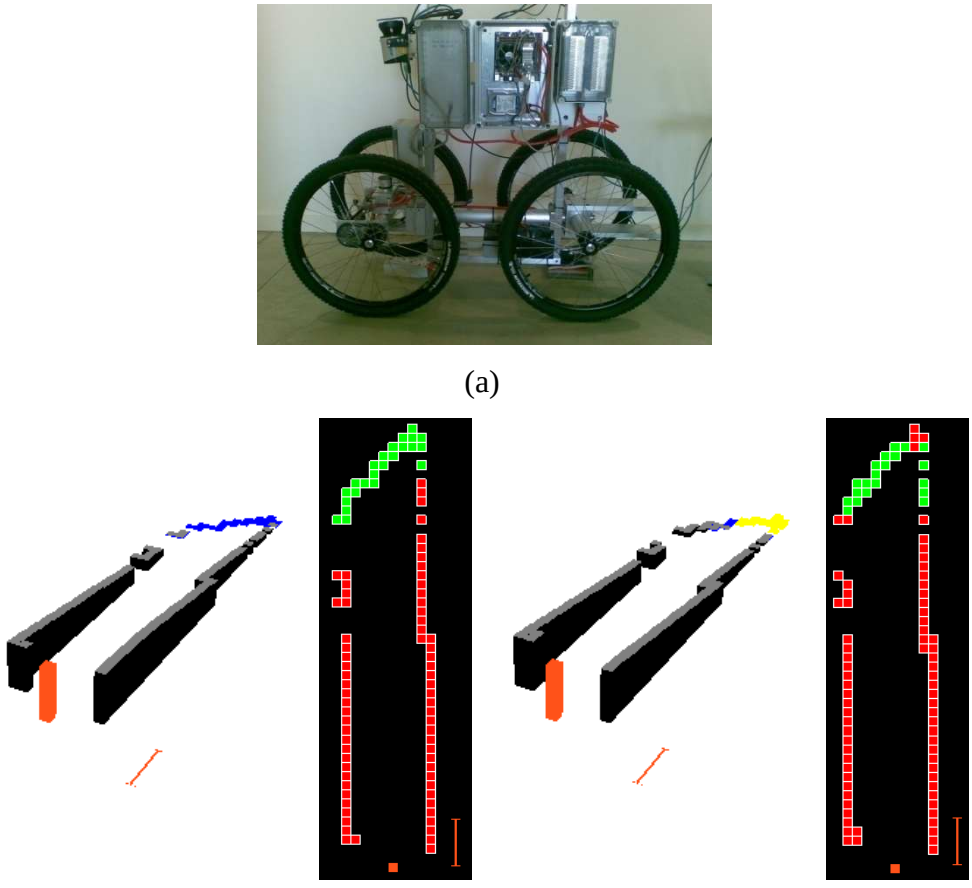
(a) (b) (c) (d) (e)

Figure 5.9 – Positive Pitch example. (a) – robot with positive pitch. (b) and (c) - elevation and obstacle maps with Pitch-Roll compensation. (d) and (e) - elevation and obstacle maps without Pitch-Roll compensation.

The third case shows a different situation. Here, the right front and rear wheels of the robot are the ones that pass through the objects, as shown by Figure 5.10 (a). In this situation the robot has a *roll* of -5.2 degrees. In this position, LADAR's beams emitted to the right of the central beam "touch" the ground a little ahead of the central beam, whereas LADAR's beams emitted to the left of the central beam "touch" the ground a little behind the central beam. Also, the walls of the right side are also detected a little higher than left side walls. Thereby, instead of having a horizontal line where the ground plane is detected, the *elevation map* has now a diagonal line, as shown by Figure 5.10 (b) and by the green cells considered as freespace on the *obstacle map* of Figure 5.10 (c).

On the maps without *Pitch-Roll compensation*, the *reference plane* was not updated and is positioned as if the robot was still placed as on Figure 5.7. Therefore, the comparison

between the *reference plane* and the *real plane* will not be correctly performed and the *elevation map* will have errors, as can be seen on Figure 5.10 (d).



(a) (b) (c) (d) (e)

Figure 5.10 - Negative Roll example. (a) – robot with negative roll. (b) and (c) - elevation and obstacle maps with Pitch-Roll compensation. (d) and (e) - elevation and obstacle maps without Pitch-Roll compensation.

In this figure one can see that, on the right side of LADAR's central beam, some cells with negative heights show up because *real plane*'s distances are bigger than *reference plane*'s distances. On the left side of LADAR's central beam, *real plane*'s distances are smaller than *reference plane*'s distances and some cells with positive heights show up. The ground plane almost disappears and it is only represented by a few blue cells that are positioned on the direction of LADAR's central beam, which is correct because the central beam is not affected by *roll* variations. The *obstacle map* of Figure 5.10 (e), as expected, incorrectly considers some cells located on the diagonal line of the ground plane as obstacles but still considers several cells as freespace, due to the fact that the *roll* angle caused by these objects is not very high. If this angle had a greater value, the errors committed by the

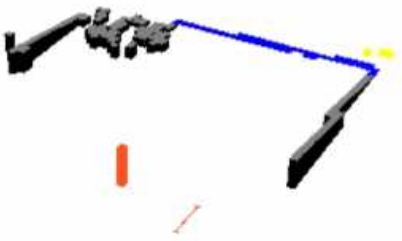
elevation map should be clearer on the *obstacle map*, because the heights of the cells that belong to those fictitious objects would increase and only the cell corresponding to the central beam would be considered as freespace.

5.6 Yaw compensation

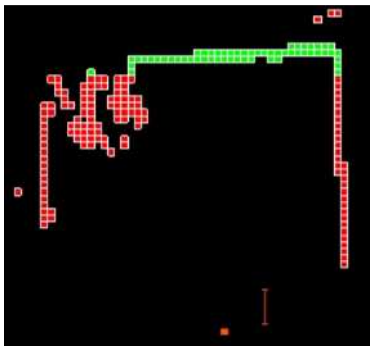
This section presents the results of an experiment that was carried out in order to assess the performance of the *Yaw compensation* procedure. As explained on section 4.6, *Yaw compensation* is performed by the rotation of the *elevation map* according to the variation of *yaw* angle. In this experiment, first the robot is asked to travel straight ahead at a speed of 0.2 ms^{-1} and, at a certain moment, the robot is asked to make a left turn of, approximately, 90 degrees, finishing his path after the end of the curve. At the end of this path, the total variation for the *yaw* angle given by the AHRS sensor is approximately 92 degrees.

The path taken by the robot is shown on Figure 5.11 (a), (d) and (g), which represent, respectively, the beginning, middle and end of the path. The *elevation* and *obstacle maps* taken at each of these moments are represented on Figure 5.11 (b) and (c) (beginning), Figure 5.11 (e) and (f) (middle) and Figure 5.11 (h) and (i) (end).

(a)

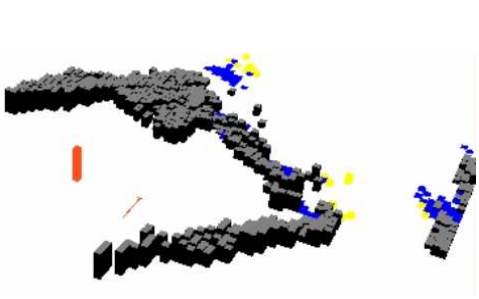


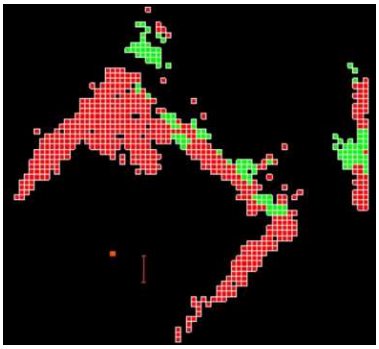
(b)



(c)

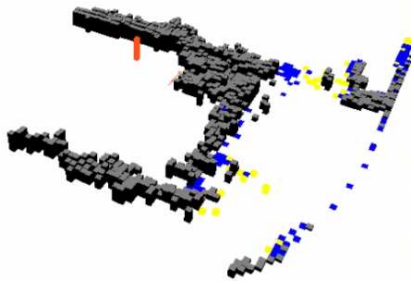
(d)

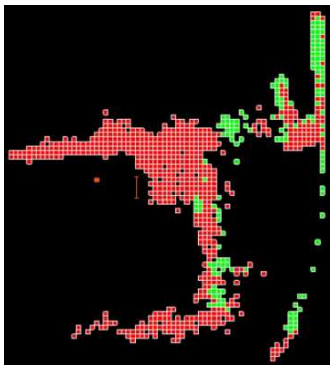




(e) (f)

(g)





(h) (i)

Figure 5.11 – Yaw compensation example. Beginning of the path: (a) – real scene, (b) – elevation map, (c) – obstacle map. Middle of the path: (d) – real scene, (e) - elevation map, (f) – obstacle map. End of the path: (g) – real scene, (h) - elevation map, (i) – obstacle map.

In this experiment, the threshold for *pitch* total variation (ε) was set to a high value (0.9 degrees), so that the maps were not deleted and, thus, one can observe the whole map rotation. Also, the limits among which it is considered that a cell belongs to the ground plane were set to average heights between -5 and +5 centimeters.

In the *elevation map* obtained at the beginning of the path one can see several aspects of the real scene of Figure 5.11 (a), such as the wall on the right side of the robot, a portion of the ground plane and a wall and some vegetation on the left side of the robot. Also, the LADAR captures some cells beyond the portion of the ground plane that was also captured. This portion of ground plane is placed immediately before the walls and door which are in front of the robot. Therefore, the only reason why the LADAR captures points that are beyond these two surfaces is because these are made of glass and this is a kind of material that causes problems to the LADAR. In the next two situations this detail will be clearer. The *obstacle map* is consistent with the *elevation map* as it considers all surfaces as obstacles, except for the portion of ground plane, which is correctly considered as freespace.

After the robot has traveled half way, the *elevation map* has been significantly rotated (Figure 5.11 (e)). Although there is some drag when the map rotates (which leads to an increase on the areas of the surfaces previously mapped), the different surfaces maintain their position relatively to the robot. At this point, the real scene (Figure 5.11 (d)) shows that the robot is facing the vegetation (the same vegetation which, on the beginning of the path, was on the left side of the robot), and the *elevation* and *obstacle maps* are consistent with this fact. Also, the portion of ground plane previously "caught" by the LADAR is now overlapped by cells corresponding to the glass wall that is now on the right side of the robot (to avoid confusion, from now on this wall will be referred to as "main glass wall") and which, meanwhile, was also captured by the LADAR. As mentioned before, the number of cells that correspond to surfaces which are beyond the main glass wall is now much higher than at the beginning of the path. The *obstacle map* now considers almost all cells of the scene as obstacles (except for some of those surfaces that are beyond the main glass wall) due to the fact that the portion of ground plane previously captured is now almost entirely covered. In the same place where previously were freespace cells that corresponded to this portion of ground plane, are now several obstacle cells which represent a part of the main glass wall.

When the robot reaches the end of the path, the *elevation map* (Figure 5.11 (h)) suffered a rotation of approximately 90 degrees relative to the initial *elevation map* (Figure 5.11 (b)). The areas of the previously captured surfaces didn't change substantially in relation to the previous *elevation map* (Figure 5.11 (e)) and the positions of these same surfaces relative to the robot are consistent with the real scene (Figure 5.11 (g)). For example, the positions, on

the real scene, of the vegetation on the right side of the robot and, also, of the wall and the door in front of the robot are consistent with the positioning of the cells that represent those surfaces on the *elevation map*. In the second half of the path, the LADAR continued to detect surfaces that are beyond the main glass wall (which, once again, is the wall that the robot was facing at the beginning of the path) but, on the other hand, the LADAR didn't detect any surfaces beyond the wall (also made of glass) that is at his front at the end of the path, because, in the real scene, this glass wall has the blinds closed preventing the LADAR from detecting anything beyond that. The *obstacle map*, once again, is in line with the *elevation map*, as it has several obstacle cells and only a few freespace cells, which correspond mostly to the blue cells of the *elevation map*.

6. Conclusions and Future Work

This chapter summarizes this dissertation, discusses the main capabilities and weaknesses showed by the proposed mapping and obstacle detection system on the experimental results, provides a set of conclusions and suggests some improvements to be made in future work.

6.1 Conclusions

This dissertation presented a solution for the problem of mapping and obstacle detection in indoor/outdoor structured environments, with particular application on service robots equipped with a LADAR. This solution works in unknown environments and it is based on the assumption that the robot, which carries the LADAR and the mapping and obstacle detection system, is based on a planar surface which is considered to be the ground plane.

The mapping module of the system creates a terrain map, which is then used to detect obstacles. The obstacle detection module generates a map that represents only obstacles and freespace. An AHRS sensor is used to increase the robustness of the system to variations on robot's attitude, which, in turn, can cause false positives on obstacle detection. It were also developed a scrolling procedure that updates the maps in accordance with the displacements of the robot and an additional procedure to deal with situations where the ground plane changes its orientation. Moreover, the developed solution was implemented on a framework for mobile robotics applications named *Player/Stage Project* that improves the communications between different modules of the system and increases computational efficiency by executing several navigation tasks simultaneously.

Several experimental tests were conducted in real environments and, by analyzing the experimental results, some conclusions can be drawn.

The need for this system to comply with real-time constraints makes it very important to choose an appropriate size and resolution for the maps. Given this choice, it is also clear that there is a relationship between the detail of the information and the computational load of the maps (processing time and amount of memory occupied) that needs to be respected so that real-time constraints are met and robot's safety can be assured.

The computation of the *reference plane* is very important for the creation of the *elevation map* and it was shown that the *reference plane* is computed properly because, in a situation

where the LADAR only captured the ground plane, the *reference plane* corresponded very closely to the *real plane* provided by the LADAR and, as a result, the maps only showed cells that belong to the ground plane.

The component of this system that deals with situations where the ground plane changes its orientation makes this system more reliable and useful for navigation and path planning, because, when the transition between two ground planes occurs, it allows the removal of fictitious obstacles and the creation of new maps much more consistent with the real scene. However, the use of two thresholds leads to the need of their adjustment depending on the environment where the robot will operate.

Experimental results also demonstrate that *Pitch-Roll compensation* is a very important element for this system because, as shown, it allows the construction of *elevation maps* which are more consistent with what is "seen" by the LADAR at each moment and, consequently, it helps to reduce the number of false positives on *obstacle maps*, which contributes to a greater efficiency and reliability on obstacle detection. It was also demonstrated that *Yaw compensatio*n works well in its essence, because the map is properly rotated and the captured surfaces change their position relative to the robot in agreement with what happens in reality. However, this procedure has a considerable drawback, because there is some drag when the map is rotated, especially at the beginning of the curves, which leads to an increase on the areas of the surfaces previously mapped. This may be caused by errors on the algorithm of rotation of the map (especially in the decision about when the map is ready to be rotated), but also on the acquisition of the *yaw* angle. In this system, the *yaw* angle is obtained through the AHRS sensor and, among the three angles provided by this sensor, the *yaw* angle is the one which is less accurate because it is computed by magnetometers, which are very sensitive to external influences. This lower accuracy of the *yaw* angle may have influence on the drag of the maps. This drawback can cause significant issues to the robot, especially if it intends to change direction in a tight space, such as making a turn to enter an indoor corridor.

The experiments also indicate that the map's scrolling procedure fulfills its mission but, still, it suffers from some errors related to changes on the distances between the various surfaces and objects of the real scene with the movement of the robot (mainly due to inconsistencies between the scrolling procedure and the addition of new information to the maps) and also errors related to the computation of the displacement of the robot through the encoders (that may arise in the calculation of the displacements, but also on odometry data provided directly by the encoders), which, in turn, tend to increase as the distance travelled by the robot becomes larger.

In general, the experimental results indicate that the proposed model provides an appropriate solution for the problem of mapping and obstacle detection in indoor/outdoor structured environments. Throughout the several experiments, most structured surfaces and objects were efficiently mapped and those which could be an obstacle for the robot were also correctly detected by the algorithm of obstacle detection. However, the experiments carried out seem to indicate that the system finds more problems in outdoor than in indoor environments, mainly due to changes (smooth, but still with influence) on the ground surface, which, in turn, can produce false positives in obstacle detection. The author believes that, as a first approach, these problems could be minimized by reducing the range of the LADAR (by increasing its tilt angle and/or lowering its height), which would reduce the area of the surface considered by the system as the ground plane. Thus, the probability of having changes on that same surface within the field of view of the LADAR could be reduced. Nevertheless, due to reasons related with the construction of the service robot for which this system was developed, it was not possible to change the positioning of the LADAR and perform more tests. This and other improvements will have to remain for future work.

6.2 Future Work

Below, this section suggests some possibilities for future work based on the experimental results obtained and on the conclusions of the previous section:

- Try to reduce the errors in the scrolling procedure of the maps through the use of a complete localization system to obtain the displacements of the robot. Instead of using robot's odometry, it would be used a more precise localization system, which could lead to a reduction on the errors, as well as on their propagation over time, as the distances traveled by the robot increase;

- To improve *Yaw compensation*. Experimental results revealed that there is some drag on the cells when the map rotates, which can be problematic for the robot when moving in tight spaces. As explained on the previous section, one cause for this drag may be related to the fact that the *yaw* angle provided by the AHRS sensor is not accurate. Thus, it would be interesting to explore a new method to obtain the yaw/heading of the robot to try to improve the results;

- To develop a mechanism to regulate and fix ("offline") the positioning of the LADAR, so that it is possible to modify its range depending on what is needed. For this system in particular, the possibility of reducing LADAR's range would reduce the area of the surface considered by the system as the ground plane, which could lead to improved results on outdoor structured environments, since these environments, although in a smooth way, are a bit rougher than the indoor environments. Within this subject, it could also be investigated the possibility of developing an algorithm for detection of smooth slopes on the *elevation map*, in order to improve obstacle detection when the robot is faced with such situations;

- To explore the detection of dynamic obstacles. So far, this system considers all surfaces as static, i.e. each detected surface remains on the maps until eventually the maps are erased. However, there are objects that can move in front of the robot (like a human that passes in front of the robot) and, accordingly, it would be interesting to develop a mechanism that could distinguish between objects that remain in front of the robot and those who are no longer there, so that the last ones could be removed from the map;

- To investigate the detection of some unstructured objects. Even in structured environments, there is often the existence of unstructured objects such as grass or low vegetation. In this system, such objects will often be considered as obstacles when, in many cases, they could be overcome by the robot. Therefore, it would be interesting that this system had the ability to distinguish some unstructured objects, either by the integration of new sensors (e.g., to have information about the color of the objects) as well as by the development of algorithms that, based on the maps built, could detect some unstructured surfaces.

Bibliography

[ACME, 2009] ACME Worldwide Enterprises, Inc., *F-16 Dynamic Motion Seat*, retrieved in December 2009, *http://www.acme-worldwide.com/dynamic_motion_seat_F16.htm*.

[Batavia and Singh, 2002] Batavia, P., and Singh, S. (2002). Obstacle detection in smooth high curvature terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3062-3067.

[Broten and Collier, 2006] Broten, G. and Collier, J. (2006). Continuous motion, outdoor, $2\frac{1}{2}$D grid map generation using an inexpensive nodding 2-D laser rangefinder. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 4240-4245.

[Chatila and Laumond, 1985] Chatila, R., and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 138- 145

[Chella et al., 2007] Chella, A., Liotta, M., and Macaluso, I. (2007). CiceRobot: a cognitive robot for interactive museum tours. *Industrial Robot: An International Journal*, 34(6): 503-511.

[Cheng et al., 2005] Cheng, Y., Maimone, M., and Matthies, L. (2005). Visual Odometry on the Mars Exploration Rovers. In *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, pages 903- 910

[Craig, 2005] Craig, J. (2005). *Introduction to Robotics – Mechanics and Control, third edition*. ISBN:0-13-123629-6. Pearson Education, Inc., New Jersey.

[Elfes, 1987] Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3): 249-265.

[Fischler and Bolles, 1981] Fischler, M. and Bolles, R. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*. 24: 381-395.

[Graf et al., 2004] Graf, B., Hans, M., and Schraft, R. (2004). Care-O-bot II—Development of a Next Generation Robotic Home Assistant. *Autonomous Robots*, 16(2): 192-205.

[Grewal et al., 2001] Grewal, M., Weill, L., and Andrews, A. (2001). *Global Positioning Systems, Inertial Navigation, and Integration*. ISBN:0-471-20071-9. John Wiley & Sons, Inc., New York.

[Hamner et al., 2008] Hamner, B., Singh, S., Roth, S. and Takahashi, T. (2008). An efficient system for combined route traversal and collision avoidance. *Autonomous Robots*, 24(4): 365-385.

[IFR, 2009] International Federation of Robotics (IFR), *Service Robot*, retrieved in November 2009, *http://www.ifr.org/service-robots/*.

[Konolige et al., 2008] Konolige, K., Agrawal, M., Bolles, R., Cowan, C., Fischler, M. and Gerkey, B. (2008). Outdoor mapping and navigation using stereo vision. *In Experimental robotics – the 10th international symposium on experimental robotic. Springer Tracts in Advanced Robotics.* 39: 179-190.

[Kuipers and Byun, 1991] Kuipers, B., and Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8: 47-63.

[Lacaze et al., 2002] Lacaze, A., Murphy, K. and DelGiorno, M. (2002). Autonomous mobility for the demo III experimental unmanned vehicles. *In Association of Unmanned Vehicle Systems International Conference on Unmanned Vehicles (AUVSI'02).*

[Lalonde et al., 2006] Lalonde, J-F., Vandapel, N., Huber, D., and Hebert, M. (2006). Natural Terrain Classification Using Three-Dimensional Ladar Data for Ground Robot Mobility. *Journal of Field Robotics*, 23(10): 839-861.

[Lee et al., 2009] Lee, Y.-C., Lim, J. H., Cho, D.-W., and Chung, W. K. (2009). Sonar Map Construction for Autonomous Mobile Robots Using a Data Association Filter. *Advanced Robotics*, 23: 185-201.

[Manduchi et al., 2005] Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2005). Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. *Autonomous Robots*, 18(1): 81-102.

[Matthies et al., 2002] Matthies, L., Xiong, Y., Hogg, R., Zhu, D., Rankin, A., Kennedy, B., Hebert, M., Maclachlan, R., Won, C., Frost, T., Sukhatme, G., McHenry, M., and Goldberg, S. (2002). A portable, autonomous, urban reconnaissance robot. *Robotics and Autonomous Systems*, 40(2-3): 163-172.

[Moghadam et al., 2008] Moghadam, P., Wijesoma, W., and Feng, D. (2008). Improving Path Planning and Mapping Based on Stereo Vision and Lidar. In *10th Intl. Conf. on Control, Automation, Robotics and Vision*, pages 384 – 389.

[Owen, 2010] Owen, J. (2010), *How to Use Player/Stage, 2nd Edition*, retrieved in April 2010, *http://www-users.cs.york.ac.uk/~jowen/player/playerstage-tutorial-manual.pdf*.

[Player Manual, 2010] Player Manual, *The Player Robot Device Interface*, retrieved in April

2010, *http://playerstage.sourceforge.net/doc/Player-2.0.0/player/index.html*.

[Player Project Wiki, 2010] Player Project Wiki, *About the Player Project*, retrieved in April 2010, *http://playerstage.sourceforge.net/wiki/Main_Page*.

[Player Project, 2010] The Player Project - Free Software tools for robot and sensor applications, *The Player Project*, retrieved in April 2010, *http://playerstage.sourceforge.net/*.

[PSU Robotics RoboWiki, 2010] PSU Robotics RoboWiki, *Player/Stage Drivers*, retrieved in April 2010, *http://psurobotics.org/wiki/index.php?title=Player/Stage_Drivers*.

[Roboteq, 2007] Roboteq (2007). *AX3500 Motor Controller User's Manual*. Roboteq, Inc.

[Santana et al., 2007] Santana, P., Barata, J., and Correia, L. (2007). Sustainable robots for humanitarian demining. *International Journal of Advanced Robotic Systems*, 4(2): 207-218.

[Shreiner et al., 2007] Shreiner, D., Woo, M., Neider, J., and Davis, T. (2007). *OpenGL Programming Guide – Sixth Edition*. ISBN:0-321-48100-3. Pearson Education, Inc., Boston.

[SICK, 2008] SICK (2008). *LMS100/111/120 Laser Measurement Systems. Operating Instructions*. SICK AG.

[Thrun, 2003] Thrun, S. (2003). Robotic mapping: a survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring artificial intelligence in the new millennium*, pages 1 - 35. ISBN:1-55860-811-7. Morgan Kaufmann Publishers, San Francisco.

[Vaughan et al., 2003] Vaughan, R., Gerkey, B., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317-323.

[Xsens, 2008] Xsens (2008). *MTi and MTx User Manual and Technical Documentation*. Xsens Technologies B.V.