Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

Phd Thesis

An Inverse-Perspective-based Approach to Monocular Mobile Robot Navigation

---

by

Francisco Jesús Bonin Font

Supervisors: Dr. Alberto Ortiz Rodriguez

and

Dr. Gabriel Oliver Codina

Dissertation sumbmitted in partial fulfillment of the requirements
for the degree of
DOCTOR EN INFORMÀTICA

Palma, January 2012

An Inverse-Perspective-based Approach to Monocular Mobile Robot Navigation

Francisco Jesús Bonin Font

March 15, 2012

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

Introduction

This part presents the motivation and the necessary background to introduce the reader in the main issues of this dissertation.

Chapter 1 proceeds with the general motivation that lead to understand the importance of the algorithms presented in this dissertation, circumscribed in the context of navigation strategies for autonomous robots. Next, the principal requirements for the new algorithms are outlined. The aim is overcoming several limitations encountered in other visual approaches, and designing a system suitable in environmental conditions as general as possible.

Chapter 2 examines in the first section the different types of control architectures for mobile robots, emphasizing and reviewing the concepts of deliberative, reactive, map-based or mapless systems. The second section of this chapter introduces the background of the localization problem.

Chapter 3 surveys the most outstanding approaches on visual navigation from the final eighties up to nowadays. The review of all these pieces of work places the reader into the main problems encountered during all these years concerning how to extract and interpret the environmental information from digital images. Reviewing a wide range of visual navigation approaches leads to a complete comprehension of what has been done until the moment and what is still remaining to be done. Of course, all these pieces of work have been an important source of inspiration for this research.

CHAPTER 1

INTRODUCTION

1.1 Motivation

In general, autonomous robots must be endowed with all the necessary capabilities to move safely through their operative areas while executing the intended tasks, such as efficiently proceeding towards a target point.

The optimization of routes is also linked with the ability of the autonomous mobile agents to detect obstacles, and to determine which is the best option to avoid them, bearing in mind that the final objective is to reach the target. Optimizing routes means optimizing time and energy resources.

The different navigation techniques addressed to mobile robots can be roughly divided in map-based and mapless systems [18]. Map-based systems plan routes and their performance on the basis of the map loaded in the robot core. Mapless systems have no prior knowledge of the environment, analyzing it on-line to determine the route to follow.

Both different techniques commonly need to deal with the robot position. In the case of map-based techniques, the robot must be self-located inside the map and along the planned path. In mapless systems, the robot needs to know its self global coordinates to control its route towards the different targets.

A third type of navigation system corresponds to some navigation approaches that do not assume any previous knowledge of the environment but they include the implementation of local occupancy maps that show the presence of obstacles

in the vicinity of the robot. These systems try to perform a symbolic view of the surrounding world and can be considered as hybrids because they combine some typical aspects from the map-based systems with some other from the mapless systems. These maps are updated on-line and used to navigate safely.

Obstacle detection and collision avoidance are crucial abilities that have to be included in any navigation system addressed to mobile autonomous agents. This is particularly important in mapless or hybrid navigation approaches, where the robot has no information about the environment where it has to operate. Range sensors, basically ultrasonic and laser, have been traditionally used for obstacle avoidance and mapless navigation in structured and non-structured scenarios [5] [136]. However, the former are only able to provide sparse sets of readings and thus aggregating a group of them is necessary to cover a relatively wide field of view, while the latter are still expensive if one wants to scan also a relatively wide range of orientations. Lately, visual solutions have emerged as competitive alternatives because of the low cost of cameras, the richness of the provided sensor data and the larger spatial and temporal resolution available. The increasing capabilities of nowadays computers, make it possible to successfully perform image processing in real-time, and thus to run on-line navigation, obstacle avoidance, mapping and localization algorithms.

As previously mentioned, another crucial task in mobile robotics is the continuous estimation of the robot pose, or self-localization for short. This task usually involves the use of proprioceptive sensors such as wheel odometers, gyroscopes or accelerometers. However, the information that is obtained by these sensors can not be entirely trusted during long paths since they are liable to drift. Then, exteroceptive sensors have to be used to correct position data and to close the localization loop. Similarly to navigation, range sensors constitute the traditional approach for robot localization [51], [26] but vision-based solutions have progressively been emerging over the rest, principally visual odometry approaches (for example [161]) and, more sophisticated, visual SLAM (see for example [152] and [30] among many other solutions). Many of this visual localization approaches make use of EKFs (*Extended Kalman Filters*). World-centric EKF-based localization algorithms deal with all pose information measured with respect to the global fixed coordinate frame. These systems present inconsistencies in the pose estimations caused by errors accumulated due to the linearization process inherent to the EKF [90]. Conversely, in robocentric approaches all coordinate measurements are computed with

respect to the robot position. This allows to deal better with linearization errors than traditional filter-like world-centric systems [29].

If a camera is already used for navigation and obstacle avoidance, it is desirable to take advantage of it when estimating the robot pose, so that the robot sensorial equipment can be reduced. Moreover, if the same data gathered to perform one of these tasks is also utilized to perform the other, then, the system might also simplify its software design and its computational charge. Consequently, one of the objectives of this work is the design of a system capable of performing the tasks of obstacle detection, collision avoidance, navigation in a reactive context and self-localization, using a single visual sensor and the same data set of environmental information.

Moreover, an exhaustive analysis of the state of the art in visual navigation would reveal some important deficits in the solutions proposed up to now, leading to the possibility of offering new or improving proposals. For example, there are a lot of solutions based on image segmentation, but they fail when the illumination conditions are deficient or in front of highly textured floors. As another example, other systems try to find relations between coplanar points in successive images, but this requires a method to discriminate the different planes of the scene and to assign each image point to the corresponding plane, which can be a challenging task in intricate scenarios. Therefore, the other principal objectives of this thesis will be: a) surveying the most of the visual navigation approaches, from the late nineties until nowadays, to analyze the main weak and strong points, and to see possible ways for new contributions, b) designing a new system that overcomes the principal vulnerabilities, at least, as many as possible, found in the surveyed pieces of work with the unique constraints of assuming a flat ground and knowing the camera world coordinates, its pitch and yaw angles and its speed of motion.

1.2 Document Structure

The rest of the document is organized as follows:

Part I [Chapters 2]: Reviews robot architectures and general concepts about navigation and localization are introduced to understand and to easily follow the subsequent presented thesis and developments.

[Chapter 3]: Extensively reviews the main research contributions on visual

navigation from the late nineties until nowadays. The final discussion included in this chapter reveals the strong and weak points of the main trends reviewed in the survey and proposes the general requirements and the general outline of the contribution described in this thesis, specially designed to overcome some of the described problems.

Part II describes the complete Navigation algorithm: the obstacle detection, the occupancy map building, the navigation strategy and the localization process.

[Chapter 4] details: a) the image feature classifier and its assessment using ROC (*Receiver Operating Characteristic*) curves, b) the obstacle detection process and the computation of range and angle with respect to the robot, c) the process of local map building and some preliminary tests to prove the suitability of the proposed obstacle detection algorithm in autonomous navigation missions.

[Chapter 5] exposes the experimental assessment of different feature descriptors and trackers and concludes which is the one that supplies the best results to our system.

[Chapter 6] states the overall navigation strategy and shows a set of experimental results in which the robot must go from a starting point to a goal point in different scenarios with different problems.

[Chapter 7] focuses on solving the localization problem using an EKF and presents an extensive set of real experiments conducted indoors and outdoors.

Part III concludes the present document, suggests the forthcoming work to extend the research described here and presents a list of all the related publications.

CHAPTER 2

BACKGROUND

2.1 The Navigation Problem: Control Architectures

Talking about the *navigation problem* one can get a little bit confused, since it is not clear how many concepts these terms can include. In autonomous mobile robotics, the *navigation problem* basically refers to the ability of the agent to move through the environment from a starting point to one or several targets, without collisions and in an efficient way in terms of mission achievement [104, 37]. Navigation abilities might need to increase in underwater or in aerial robots, since they must take into account the three dimensions, contrarily to the terrestrial robots, which are more likely to consider only two dimensions.

Navigation capabilities will be given by the software architecture installed and used to control the robot. The software architecture determines the general abilities of the robot, for example: building maps or interpreting them for navigating, detecting obstacles, learning from the environment, computing its own pose, or matching all it perceives with the data stored in memory.

In the literature there are many different descriptions refering to control architectures. People from different fields have their particular vision of how architectures have to be defined and which must be their competences in an autonomous agent. Let us propose a compilation of several well accepted definitions to compose a rough

description of this term:

- From the software point of view, a robot control architecture is a highly specific collection of software building blocks, their interfaces, usability and organization between them to provide the control system with a particular global functionality. All software components are installed in the robot core computer, and are specifically dedicated to manage the robot behavior. However, lately, distributed control architectures share resources and efforts between the robot and some external control units. The software architecture generates orders directly addressed to the actuators, according to, a) the environmental conditions captured via the different sensors, b) a collection of preprogrammed actions designed to respond to any different external stimulus, and c) a set of planned tasks.

- From a more functional point of view, a control architecture is the set of structural software components that provide the robot with the capability of performing its three main functions: to perceive, to reason and to act [7].

It is commonly accepted that robot architectures can be classified as deliberative, reactive or hybrids. Pure reactive or deliberative systems are each one in one extreme of the spectrum, while hybrid systems combine the advantages of both trying to minimize their disadvantages.

The main characteristics of these architectures are exposed in the following section.

2.1.1 Deliberative Control Architectures

General Outline

In deliberative architectures, navigation or action decisions are taken on the basis of pseudo-logical reasoning derived from: a) the previous analysis of environment models, or b) the recognition of landmarks or predefined patterns, or c) guided tours, or any kind of previously programmed plan of actuation based on the knowledge of an already explored scenario.

Deliberative agents usually need to be provided with some kind of representation of the environment to perform their programmed tasks. These world models can be

built either by a tele-operated agent, or by an autonomous mobile robot, previously to the autonomous navigation stage or simultaneously to it.

During the autonomous navigation phase, a deliberative agent must continuously contrast the data contained in the stored model with the data that the sensorial equipment captures online from the environment. By comparing the stored data with the perceived data, the robot is able to localize itself and, depending on its position and state, it is also able to run the corresponding planned actions. Information flows from the sensors to the world model and from the world model to the actuators, but never in the reverse manner. According to this, agents involved in map building tasks and map-based deliberative systems have to be specially accurate in:

a) translating the real world into a sufficiently accurate model with an adequate symbology to be useful for logical reasoning,

b) representing the complex, or sometimes dynamic, real world entities and events in order to be useful for itself or for others to work with them, and

c) analyzing the environmental information and generating the corresponding reasoning in time to be able to react online to the events produced in the environment.

The lack of precision in any of these three important points can lead to an inadequate world representation and thus to an inefficient robot, either in the planning or in the acting performance.

Then, it is reasonable to define three main logical and symbolic layers, known as the SPA (Sense-Plan-Act) approach, for the classic deliberative architectures [7]:

1. Sensorial module: this is the module dedicated to incorporate the data captured by the sensors into the world model.

2. Planner: this module must execute a plan for a determined goal, taking into account the perceived environment.

3. Execution system: its task is to translate the plan into navigation orders for the different actuators.

In consequence and summarizing, the main shortcomings of these kind of agents that can induce incorrect robot behaviors are:

- The inaccuracy of the sensors derive to errors in the world modeling process. These errors in the world model can affect the planning, in the same way that sensor errors affect the mapping task.

- Deliberative agents can be ineffective in dynamic environments when the frequency of the events occurred in the real world is higher than the time the robot needs to sense, plan and act.

Map Building

Although some visual reactive or hybrid systems can simultaneously build local maps and/or localize themselves, the construction and the use of maps of the complete environment is mostly inherent to deliberative architectures. Maps can be either needed for human guidance and thus they are built to be interpreted by human beings, or they can be used by an autonomous mobile robot for navigation tasks or path planning. In this case, maps need to be interpretable by the robot but not necessarily by a human being.

There exist two major paradigms for mobile robot mapping: *Metric* and *Topological* maps.

Metric Maps : Metric maps are built using fine metric information of the environment, that is, distances, measures, sizes, heights, angles, etc... and they are referenced to a global coordinate system (so-called the world coordinate system in this document). These maps are intended to use metric information as accurate as possible since they try to represent the environment as precisely as possible. To this end, agents dedicated to map-building must pay special attention to the sensors that are used to retrieve the environmental information. Sensors for metric map building must be specially calibrated and uncertainties in their measurements have to be taken into account to anticipate possible errors in the map. Metric maps have the additional problem of accumulating too much data, considerably increasing the storage needs of the system and the computational time.

The metric map building problem is strongly linked to the localization problem. On the one hand, for building maps as accurate as possible it is necessary to know the exact localization of the robot at each moment. And, on the other hand, for a correct localization, in many cases it is important to be provided

with a map as accurate as possible. Errors in the mapping procedure, caused basically for sensor inaccuracies are also translated into localization inaccuracies, and vice-versa. Concurrent or simultaneous localization and mapping has become in the last decade one of the most challenging problems in the robotics research, either using vision, ultrasounds or laser as the main sensor.

One of the most extensively used metric map configurations that deserves a special attention is the *Occupancy Grid*. *Occupancy grids* divide the environment in rectangular cells. Each cell is labeled with a certainty of being occupied by an obstacle or a part of it. As it is exposed in the next chapter, *Occupancy Grids* can represent a vast outdoor region, a complete indoor environment, or simply the vicinity of the robot (local maps). These local map-based systems can be considered to be in the borderline between the deliberative systems and the hybrid ones since they try to infer a map of the environment but only of the portion very close to the robot, which is a certain sign of reactiveness.

As it is shown in chapter 3, a lot of approaches performing *Occupancy Grids* are found in the literature. Ultrasounds, laser range finders and visual sensors have been indistinctly used for building them.

Topological Maps : Topological maps are coarse abstract representations of the environment, usually denoted by a graph with nodes and links between nodes. Nodes represent different points, sites, landmarks or features of the environment and links represent or can be labeled with actions or relationships between the different nodes. For example, links can represent time of traveling from one point to another or an approximate distance. Topological maps are, in general, not referenced to any coordinate system and they generate much less information to be stored than metric maps. Furthermore, the localization problem relaxes its constraints to a location recognition and these systems are less vulnerable to noise or sensor inaccuracies. However, topological maps are, in general, useless for obstacle detection and avoidance.

Some authors have analyzed both paradigms and explored hybrid (metric-topological maps) solutions where the benefits of both techniques are combined [142], [195], [199].

2.1.2 Reactive Control Architectures

General Description

Reactive systems usually do not need any previous knowledge or abstract representation of the navigating environment. Instead, navigation decisions are taken as the information of the surroundings is perceived, trying to couple perception to action in the best possible way. Purely reactive systems directly react as they perceive the world, avoiding obstacles or evaluating online the best choice for getting the destination point through the optimal path. These systems are often used in robot navigation approaches since they are usually faster than map-based systems.

Purely reactive architectures are an application of the *sense and act* paradigm. Environments where robot moves can be unstable or dynamic, and, in these cases, it is convenient to fit the robot with a set of high level rules to be able to overcome all unexpected situations. Another key issue of these systems is to appropriately sense the environment and often enough to be able to properly generate the navigation decisions. Therefore, putting all efforts in the sensing systems instead of in the planning systems is very important to design a useful reactive architecture. The sensors the robot is equipped with have to be convenient enough to cover all tasks in all environments. Remarkably, reactive systems can properly react in hazardous or highly dynamic environments, where a great level of unpredictability can cause the robot to get stuck.

Reactive architectures are usually composed by a set of preprogrammed rules, so called behaviors, which try to give fast and accurate response to the different situations appeared during the navigation process.

Behavior-based Architectures: An Overview

The core of a reactive architecture is usually formed by a collection of predefined directives which state each robot response for each possible external stimuli or situation that the robot is able to perceive.

These directives are commonly known in the robotics field as behaviors, and they are embedded in a bottom-up hierarchical collection of modules with different levels of abstraction. High level modules implement the abstract representation of behavioral rules and the low level modules implement the physical communication with sensors and actuators. Each behavior can be implemented by a software module and the system can add, remove, enable or disable behaviors as needed.

Robotic Behaviors can be designed according to different criteria, for example [7]:

- In the last decades, many researchers have focused their efforts in trying to extrapolate the behavior of some animals, basically some insects, to the robot behaviors. Animals provide powerful insights of how robot behaviors can be constructed.

- Designs are often based on the activities in which the robot is involved or on the situations in which the robot is found itself. The robot actuation is simplified to identify a determined situation and choosing the pre-programmed proper action to overcome it.

- Some other robot architecture designers prefer to firstly prove that it has been previously demonstrated that all the pre-programmed behaviors are suitable to give response to all the possible considered situations Behaviors are debugged in real world and modified or recycled until they exhibit satisfactory performance.

One of the most popular behavior-based technique found in the literature is the Potential Fields method [97]. This method was developed to generate smooth trajectories in navigation tasks and can be roughly defined as the method of appropriately combining different behaviors represented by repulsive or attractive action vectors. Vectors are related to obstacles and targets as a repulsive or an attractive force, respectively. Forces potential drop off with the square of the distance between objects and the robot. The robot is affected by the repulsive forces coming from nearby obstacles and by the attractive force generated by the goal point. The different forces (repulsive and attractive) are combined to yield a single field. The resulting force vector determines the trajectory direction and speed.

The techniques based on potential fields is not exempt of problems. Its main pitfalls are [100]:

1. The method is vulnerable to local minima, which makes the robot stay trapped in between certain obstacle configurations.

2. Attempting to pass through closely spaced obstacles or door apertures is not properly solved.

3. The system can get unstable if there are irregularities in narrow passages or in the presence of multiple obstacles.

When a behavior-based control architecture is in operation, behaviors can be individually applied or can be fused to obtain different or more appropriate responses to a vaster range of unpredictable situations. This is commonly known as behaviors coordination. The coordinator usually acts as the mapper between stimulus and motor orders. There exist two main techniques to combine (coordinate) different behaviors, namely, competitive and cooperative methods:

Competitive methods : When in a determined situation several behaviors are simultaneously active and get in conflict, these conflicts are resolved by the coordinator choosing the most appropriate behavior for each situation. For example, each behavior can obtain a different number of votes depending on the results obtained in previous experiences, or the coordinator response can be the result of the application of a set of rules.

Cooperative methods : The different behaviors are combined in such a way that the system takes advantage of the best of all concurrent behaviors and rejects their hazards. This method permits to concurrently use more than one behavior at the same time. Each behavior has a relative gain, which is used to multiply the behavior output module before they are combined. The Potential Fields method can be considered one of the approaches where different actors cooperatively participate in the final robot steering decision.

2.1.3 Hybrid Control Architectures

On the one hand, when an agent is moving in scenarios that can guarantee a high level of stability or when the estimation of the location of the agent in the world is absolutely necessary, then, systems with a higher degree of deliberativeness are preferred (planning, positioning, landmark detection and following, local mapping performance, among other functionalities). Furthermore, it is difficult for reactive agents to organize themselves or coordinate their behaviors with other agents in a non-trivial way.

But, on the other hand, even if a complete map of the environment is provided, robots need some kind of rules to apply when unexpected situations appear,

unlikely detectable with some kind of sensor configurations or architectures. Deliberative systems dedicated to dynamically build maps or locate themselves need an implementation of some kind of reactive behaviors to detect and avoid obstacles. Furthermore, the knowledge of the world model may help to avoid potential hazardous situations.

Consequently, many researchers feel that hybrid systems, which combine deliberative reasoning with reactive behaviors, are necessary to maximize the potential of the automatic agents. In hybrid architectures, robot behaviors are designed or built in accordance with the tasks that have to be executed, while planned schemas can anticipate environmental conditions so as to apply the behavior that best fits at each situation. A priory world knowledge or dynamically acquired environmental information can be even used to efficiently reconfigure some of the existing behaviors.

Hybrid systems are normally designed in a hierarchical structure. Lower levels are occupied by reactive modules acquiring information of the environment and interacting with the sensors and actuators, and deliberative modules lie in the upper level for goal determination, planning, decision making, localization or map building tasks.

2.2 The Localization Problem

2.2.1 Overview

The majority of applications involving autonomous robots need to know or deal with the robot position in the environment. For instance, in missions between defined or preprogrammed points, the robot pose is needed to evaluate if the goal points have been reached or not. In surveillance, exploration or mapping operations, the location of the robot is crucial because it is the only way to know which parts of the environment have been covered and which sites are still unexplored.

Depending on the nature of the sensorial robot equipment, localization techniques can be roughly classified as:

- Open-loop methods which are also known as dead-reckoning. The robot position is estimated from the data given by the proprioceptive sensors such as wheel odometers, gyroscopes or accelerometers. These sensors provide relative measurements with respect to the last captured information and the pose

is obtained integrating this information with any observation of the environment. Due to this integration process, boundless errors in the pose estimations are accumulated with time and length.

- Close-Loop methods which make use of exteroceptive sensors to observe the environment. Pose estimations are computed from data given by proprioceptive sensors corrected by data observed and captured from the environment. Since dead-reckoning data is continuously being corrected by observations, pose errors can be controlled.

Mobile robot localization is not the main concern of this thesis, but it is developed as a complement of the navigation approach. The aim is to correct the robot pose given by its proprioceptive sensors. In our system, inaccuracies in the robot pose can affect the obstacle localization and, thus, the navigation task.

There are different interpretations of the localization problem. Some systems need to know their position in a qualitative manner, analogously to a topological map. In these cases it is enough to know if the robot has been in a certain labeled place as obstacles are approximately located in the environment representation.

Other localization approaches compute accurate numerical pose estimates with respect to a reference frame. In these type of localization systems, the problem thus consist on calculating the robot coordinates with respect to a fixed world coordinate system: $(x, y, z, \theta, \varphi, \phi)$ (translation and rotation) if the robot as a total of 6 degrees of freedom, for example in underwater or aerial robotics, or (x, y, θ) if the robot is moving in a plane (3 degrees of freedom), as for terrestrial robots.

If the initial robot pose is unknown, this is called the *global localization* problem. Contrarily, the *pose tracking* or *local localization* refers to those systems where the initial pose is previously known. Normally, all localization problems start being global and continue being local once the robot has located itself in the environment.

General pose tracking approaches compute the position at one time t from the position at instant $t-1$ and some additional information captured from the environment, such as the position of natural or artificial landmarks.

2.2.2 Closed-loop Localization

Concerning closed-loop localization, one of the most relevant methods is the one based on *landmarks*.

A landmark is an element of the imaged scene that can be detected or recognized by the robot from its sensors and used as a reference point to compute its own position. The common localization or mapping models assume that the robot can measure the range and the bearing of all the landmarks, with respect to the robot's current pose.

Landmarks can be passive or active. Active landmarks continuously transmit information of location to the robot. They have the advantage of being easy to detect and that the location information is sent directly to the robot. However, they need a power supply and additional technical infrastructures which increase cost and maintenance. Passive landmarks do not emit any sign, and they have to be recognized on-line. They are cheaper and easy to design and maintain, but require more sophisticated software in the robot side. Furthermore, passive landmarks have to be invariant to scaling, rotation, and viewing angle to facilitate their successive detection and identification. From now on, we will refer only to passive landmarks.

Landmarks can also be natural or artificial. Artificial Landmarks are specifically designed to be recognized by the robot. They can be deliberately placed on the environment or simply be on it, but in all cases the robot has a certain knowledge about their characteristics which normally rely on some kind of structured information.

Traditional visual localization approaches based on passive artificial landmarks try to identify colors, shapes, patterns, textures, or, for instance, barcodes (see for example, [156], [31], [213]).

Navigation and localization systems based on natural landmarks [17] [175] have emerged over the ones based on artificial landmarks. These systems do not have any previous knowledge of the landmark characteristics or location. A particular type of visual localization algorithms search for distinctive regions in the image, considering them as the set of referenced natural landmarks. Image distinctive regions are normally labeled with a descriptor based on surrounding image data. These points, so called *features or corners*, are image points with significant visual characteristics that can be easily tracked across consecutive images. As a matter of fact, image feature detection and tracking has become itself in a research area of growing interest. Image features are used for visual odometry, localization, mapping and navigation approaches. Section 3.3.2 overviews the general principles of feature detection, lists their most significant solutions found on the literature and supplies a set of visual-navigation outstanding approaches based on tracking image features.

Many localization approaches assume that the robot has been equipped with

a map of the environment. Inversely, many map building approaches need the pose information to self-locate in the map. In these cases, they must assume that the localization problem has been previously solved. The SLAM *(Simultaneous Localization and Mapping)* techniques are devoted to implement systems able to incrementally build maps from unknown environments and to simultaneously self-locate within these maps.

Early SLAM techniques were performed from data provided by range sensors. But, analogously to navigation and localization, interest in visual SLAM has been significantly growing during the last years.

Some visual localization, odometry or visual SLAM powerful solutions track uniquely identifiable features in consecutive frames and integrate them in an EKF *(Extended Kalman Filter)* ([154, 209, 39]). The filter continuously corrects and stabilizes the information included in its state vector and it can provide future estate estimations. Theory and applications of EKF localization and SLAM are extensively detailed in [198].

Once the basis for contextualizing all the work presented in this thesis have been given, next sections will introduce the related work and proceed with the navigation and the localization algorithms.

CHAPTER 3

VISUAL NAVIGATION: STATE OF THE ART

3.1 Introduction

Different types of sensors can be used for navigation purposes, deriving into a varied spectrum of solutions. In particular, in the last three decades, visual navigation for mobile robots has become a source of countless research contributions. Navigation strategies based on vision can increase the scope of applications of any kind of autonomous mobile vehicles (land, aerial or underwater). Among the different proposals, this chapter surveys the most outstanding ones. Since in many cases the performance of a good navigation algorithm is deeply joined to an accurate robot localization approach, some vision-based localization solutions have also been included in this chapter.

Regarding the type of control architecture, navigation systems can be roughly divided in those that need previous knowledge of the whole environment (deliberative) and those that perceive the environment as they navigate through it (reactive). Visual systems that need or produce a map can be included in one of these main groups:

- *Metric map-using systems*, which need to be provided with a complete metric map of the environment before the navigation task starts.

- *Metric map-building systems*, which build the whole metric map of the environment by themselves and use it in the subsequent navigation stage.
- *Topological map-based systems*, which build or use topological maps for navigating.

Mapless visual navigation systems mostly include reactive techniques that use visual clues derived from the segmentation of an image, optical flow, or the tracking of features among frames. No global representation of the environment exists; the environment is perceived as the system navigates, recognizes objects or tracks landmarks.

Concerning visual sensors, most configurations are based on monocular and binocular (stereo) systems, although others based on trinocular configurations also exist. Omnidirectional camera systems are another type of structures that are gaining popularity because of their advantages: i) omnidirectional cameras have a 360° view of the environment, ii) with this kind of cameras it is easier to find and track features, since they stay longer in the field of view. Omnidirectional cameras are usually obtained combining a conventional camera with a convex conic, spherical, parabolic or hyperbolic mirror.

The progress made in vision-based navigation and localization for mobile robots up to the late 90's was widely surveyed by DeSouza and Kak in [49]. After the late 90's, some authors have hardly surveyed this area: examples are Kak and DeSouza [95], whose work is restricted to *navigation in corridors*, and Abascal and Lazcano [1], whose work is restricted to *behaviour-based indoor navigation*. A remarkable outline of navigation and mosaic-based positioning solutions for AUVs can be found in [43, 44] and a wide list of underwater vision tracking techniques was surveyed in [202].

The survey presented in this chapter mostly covers the work performed from the late nineties until the present day, and includes all the topics related to visual navigation.

3.2 The primary techniques until the late 90's

De Souza and Kak in [49] structure robot visual navigation in two main subjects: *indoor navigation* and *outdoor navigation*. *Outdoor navigation* is in turn subdivided in *structured* and *unstructured environments*, while *indoor navigation* is subdivided

in *map-building-based navigation* and *mapless navigation*. Tables 3.1 and 3.2 summarizes all visual navigation techniques until the late 90's included in [49].

3.3 From the late 90's up to present

In the last decade, the techniques included in tables 3.1 and 3.2 have matured into more refined versions, or have evolved into other more accurate and efficient systems. This variety of old and new techniques have extended the amount and quality of research in this area and their applications. This section surveys most of these studies distinguishing between *map-based* and *mapless solutions*.

3.3.1 Map-based Systems

This section considers techniques that build and/or use metric or topological maps.

Metric Map-using and -building Navigation Systems

This group includes systems that need a complete map of the environment before the navigation starts.

Metric map-using systems are unable to map the environment and they need to be equipped with it, while *metric map-building* systems explore the environment and automatically build its map. The navigation phase starts only if the map of the environment is available for the robot or after the map has been built. The map information can be either directly used for navigation, or it can be post-processed to improve its accuracy, and thus to calculate more precise localizations. This is the navigation technique that requires more computational resources, time and storage capability. Since outdoor environments can be large in size and extremely irregular, visual navigation techniques based on maps are in most occasions applied to indoor environments.

Map building and *self-localization* in the navigation environment are two functionalities that deliberative systems tend to incorporate. In *map-building* standard approaches, it is assumed that the localization in the environment can be computed by some other techniques, while in pure *localization* approaches, the map of the environment is presumably available. Robots using pure *localization* approaches need to track their own position and orientation in the environment in a continuous way. Accurate metric maps are essential for good localization, and precise localization

Table 3.1: Summary of the most outstanding visual navigation approaches from 1987 to late 1990's (1)

Authors	Indoor-Outdoor	Category	Method
[20, 99]	Indoor	Map based	Force Fields
[22, 149]	Indoor	Map based	Occupancy Grids
[33]	Indoor	Map based	Occupancy Grids
[188, 58, 196]	Indoor	Map based	Absolute Localization
[8]	Indoor	Map based	Absolute Localization
[128]	Indoor	Map based	Incremental Localization
[205]	Indoor	Map based	Incremental Localization
[38]	Indoor	Map based	Incremental Localization
[101, 132, 131, 150]	Indoor	Map based	Topological Map. Incremental Localization
[94]	Indoor	Map based	Landmark Tracking
[84]	Indoor	Map based	Landmark Tracking
[138]	Indoor	Map building	Stereo 3D reconstruction
[192]	Indoor	Map building	Occupancy Grid
[21]	Indoor	Map building	Occupancy Grid
[195]	Indoor	Map building	Grid and Topological Representation
[167]	Indoor	Mapless	Optical Flow

Table 3.2: Summary of the most outstanding visual navigation approaches from 1987 to late 1990's (2)

Authors	Indoor-Outdoor	Category	Method
[15]	Indoor	Mapless	Optical Flow
[50]	Indoor	Mapless	Optical Flow
[124]	Indoor	Mapless	Appearance-based Navigation
[93]	Indoor	Mapless	Appearance-based Navigation
[146]	Indoor	Mapless	Appearance-based Navigation
[204]	Outdoor	Mapless	Road Following
[71, 70, 72, 73]	Outdoor	Mapless	Road Following
[203]	Outdoor	Mapless	Road Following
[194] , [193]	Outdoor	Mapless	Road Following
[155], [91]	Outdoor	Mapless	Road Following
[208]	Outdoor	Mapless	Random Exploration
[103]	Outdoor	Map building	Given Mission Exploration
[127]	Outdoor	Mapless	Random Exploration

becomes necessary for building an accurate map. If the exploration and mapping of an unknown environment is automatically done online, the robot must accomplish three tasks: safe exploration/navigation, mapping and localization, preferably in a simultaneous way. SLAM *(Simultaneous Localization and Mapping)* and CML *(Concurrent Mapping and Localization)* techniques search for strategies to simultaneously explore, map and self-localize in unknown environments.

Davison and Kita discuss in [48] about sequential localization and map building, review the state of the art and expose the future directions that this research domain should take. Furthermore, they present a tutorial of first-order relative position uncertainty propagation and a software to perform sequential mapping and localization.

Sim and Dudek [177] proposed a framework to learn a set of landmarks and track them across the sequence of images maximizing the correlation of the local image intensity. Landmarks are characterized with position parameters and subsequently used by the robot for self-localization. Sim and Dudek [178] extended their previous work with a new strategy for environment exploration and map building, maximizing coverage and accuracy and minimizing the odometry uncertainties. This proposal maped image features instead of performing a geometrical representation of the environment, operating and managing the framework presented in [177] and adapting an *Extended Kalman Filter* localization framework described in [182] and [106]. In this approach, exploration policies were chosen among a great number of possibilities: (1) *seed spreader*, by which the robot followed a predefined navigation pattern throughout the environment; (2) *concentric*, where the robot followed concentric circular trajectories, with their center in the starting point, and the direction of movement changing at every circle; (3) *figure eight*, by which the robot followed eight-shaped concentric trajectories; (4) *random*, where the robot moved randomly; (5) *triangle*, by which the robot moved in concentric closed triangular trajectories; (6) *star*, where the robot moved along a set of rays that emanate from the starting point. Experimental results in [178] showed that, exploration efficiency (measured in observed images definitely inserted in the map) divided by the total number of processed images, was maximum for the *concentric* policy, and minimum for the *star* policy. Besides, the mean error in odometry was maximum for the *random* policy and minimum for the *concentric* policy.

Sim *et al* [179, 180] solved the SLAM problem with a stereo pair of cameras and a Blackwellised particle filter. The system implemented a hybrid approach consisting

of 3D landmark extraction for localization, and occupancy grid construction for safe navigation.

AQUA is a visually guided amphibious robot developed by Dudek *et al* [53, 67]. This system runs on land and swims into the water. Using a stereo trinocular vision system, it is capable of creating 3D maps of the environment, locates itself and navigates.

In [46], Davison reported a new Bayesian framework that processed image information of a single standard camera to perform localization. Weak motion modeling is used to map strong distinguishable features, which are used to estimate the camera motion.

Wide angle cameras present a much wider field of view than standard lens cameras. Therefore, features are visible longer and are present in more frames. Due to the distortion introduced by a wide angle lens, a previous calibration process has to be performed in order to get corrected images from original frames. In [47], Davison *et al* extended their previous work by substituting the 50° standard camera with a 90° calibrated wide angle camera, leading to a significative improvement in movement range, accuracy and agility in motion tracking. Camera calibration improves the calculation of relative positions, and consequently improves the accuracy of the localization process. On the other hand, the Shi and Tomasi algorithm [173] was adopted in [47] to extract the position of the image features, which were used as landmarks to guide the navigation process. Experimental results proved that with a wide angle camera some aspects were improved: i) the camera motion could be better identified, with particular improvements on rotational and translational movements estimation, ii) the range of movements increased, and large motions or motions with great acceleration were better dealt with, since they appeared much less abrupt.

Schleicher *et al* [168] used a top-down Bayesian method-based algorithm to perform a vision-based mapping process consisting in the identification and localization of natural landmarks from images provided by a wide-angle stereo camera. Simultaneously, a self-localization process was also performed by tracking artificial known features (landmarks). The position of these features was determined through the combination of the *epipolar line* concept, characteristic from stereo theory, and the calculation of the *fundamental matrix*. The authors proved that using the redundancy of the information extracted from the images of both cameras increases the robustness and accuracy, and decreases the processing time of the procedure. To

prove the improvements of the stereo-based solution. the system with a wide-angle stereo camera was compared with a SLAM approach that used a single wide-angle camera.

Some researchers have focused their work on approaches to recover 3D environment structures and/or to estimate robot motion models from vision information [129, 201].

Manessis *et al* addressed the 3D environment reconstruction problem using image sequences captured from n different camera views [117]. The two main contributions of this proposal are: i) a new geometric theory for surface recovery from 3D sparse data and ii) an algorithm based on a recursive *structure from motion* (SFM) method, which is used to estimate the location of 3D features and then to reconstruct the scene.

The classic process of building a 3D map using stereo images was refined by Wooden [211] under the DARPA-sponsored project Learning Applied to Ground Robots (LAGR), and particularly applied on its robot LAGR. The map building process consisted of four main steps:

- the captured stereo images were transformed into a three-dimensional representation by matching small patches in the two images,
- the real possible position of image points were deduced from the geometrical characteristics of the camera,
- a derivative was applied to the 3D map points to detect abrupt changes in slope, as for example, trees, rocks, etc..., and,
- in order to decrease the resolution of the map and smooth some variations, the result of the derivative was transformed into a cost map, where every point value was the average of the values over a defined 1.2 m $\times$ 1.2 m region.

Once the map had been created, a process of path planning was used to navigate through the environment.

When a robot explores an environment and constructs an occupancy grid, it makes approach of where the free space is. In this case, the object shape is not important, only the certainty that a fixed location is occupied by an object. In some cases, it is important to recognize the objects because they have to be picked up or manipulated, and, in other cases, it is paramount to recognize if the objects

are on a table or lying on the floor. Following this trend, Tomono [200] proposed a high density indoor map-based visual navigation system on the basis of on-line recognition and shape reconstruction of 3D objects, using stored object models. A laser range finder was also used to complement the information provided by the camera. The proposed method contemplated three main issues:

- advanced objects model creation, before the navigation starts,
- on-line object recognition and localization, during the navigation stage and,
- placement of recognized objects in the 3D map of the environment.

Other *map-based navigation techniques* are those that impose a human-guided pre-training phase. Kidono *et al* [98] developed an approximation to this type of systems. In their contribution, a human guided the robot through an environment and during this guided route, the robot recorded images with a stereo camera and incrementally (frame by frame) constructed the 3D map on-line. After the map was built, the robot was able to repeat the same route from the starting point to the goal point, tracking features and computing the shortest safe path. In this solution, the robot odometry was used to support the stereo vision sensor.

An outstanding evolution of this technique using a calibrated wide angle camera came up from Royer *et al* [164]. The robot was guided by a human in a pre-training navigation stage, recording images from the trajectory. A complete 3D map of the environment was constructed off-line, using the information extracted from the pre-recorded images. A collection of useful landmarks and their 3D position in a global coordinate system were used for localization purposes, during the autonomous navigation stage. At the beginning of the navigation process, the robot had to self-localize in the starting point where it had been left, by comparing the current image to all stored key frames to find the best match. The selected subsequent images had to present a certain movement perception between them, to provide the system with trackable feature information. Losing perceptual movement caused problems to the algorithm. In these terms, the robot was able to follow the same complete pre-recorded trajectory, saving a lot of time in the positioning process. This approximation was basically directed to city navigation, rich in visual features, and where kinematic GPS visibility can be hidden in a lot of places.

Several undersea map construction techniques combined with a proper and accurate algorithm of position estimation can also be considered to belong to the

CML category. In major cases, undersea bottom mosaics can be used by AUVs for navigation purposes. Haywood designed a system to mosaic underwater floors using images attached with accurate position coordinates [85]. Marks *et al* [118] developed a technique to implement real-time mosaics using correlation between on-line images and stored images. In a subsequent work, and following the same trend, Fleischer *et al* [60] improved the previous work [118] focusing on dead-reckoning error reduction. Previous systems often assumed that the seafloor was plane and static, and that the camera was facing it, making the image plane almost parallel to the seafloor plane. Gracias *et al* [69] proposed a method for mosaicing and localization that did not make any assumption on the camera motion or its relative position to the sea bottom. The system was based on motion computation by matching areas between pairs of consecutive images of a video sequence. Finally, Xu and Negahdaripour presented in [212] an interesting contribution to underwater mosaicing and positioning. The vehicle position was computed integrating the camera motion from consecutive frames using Taylor series of motion equations, including the second order terms, which in previous research was usually ignored.

Topological Map-based Navigation Systems

Topological maps are suitable for long distance qualitative navigation, and specially for path planning. In general, they do not explicitly represent obstacles, walls or free space so that obstacle detection and avoidance must be performed on line by other means. Visual topological maps are simple and compact, take up less computer memory, and consequently speed up computational navigation processes.

Winters and Santos-Victor [210] use an omnidirectional camera to create a topological map from the environment during a training phase. Nodes are images of characteristic places and links are sequences of various consecutive images between two nodes. During the navigation, the position is determined matching the online image with previously recorded images. The matching process is performed with an appearance-based method which consists of projecting every online image onto an eigenspace defined by the covariance matrix of a large image training set.

More recently, Gaspar *et al* use [210] to map indoor structured environments and emulate insect vision-based navigation capabilities [66]. The robot must be able to advance along corridors, recognize their end, turn into the correct directions and to identify doors. The division of the map into nodes allows splitting the navigation

task into sub-goals. Every sub-goal is recognizable with landmarks and covers the movement between two nodes; for instance, two doors joined by a corridor. Navigation between two nodes works through detection of the corridor parallel sides and generation of the adequate control signals.

Another topological map-based navigation strategy for indoor environments comes from Košecka *et al* [102]. In a previous exploration stage, video is recorded and, for each frame, a gradient orientation histogram is computed. After that, a set of view prototypes are generated using Learning Vector Quantization over the set of histograms gathered. Each histogram corresponds to a node in the topological map. During the navigation phase, the gradient orientation histogram of each frame is compared with the view prototypes to determine the location it most likely comes from using the nearest neighbor classification. In case the quotient of the distances with the nearest and the second closest histograms/views is below a certain threshold, the classification is considered correct and a location is obtained; otherwise, the classification is refined by comparing sub-images of the new image and the images in the database closest to the view prototypes.

Remazeilles *et al* proposed a system based on environment topological representation and a qualitative positioning strategy [158]. Nodes were represented by views captured in a training phase and edges represented the possibility of moving from one scene towards another. The robot navigated tracking landmarks over consecutive frames and keeping them inside the field of view. The localization strategy used in this approach is qualitative since it informs that the robot is in the vicinity of a node, instead of giving exact world coordinates.

Museum guiding robots is one of the map-building applications that has proved to be greatly useful, in contrast to other solutions that need the museum map to navigate. These robots need to be autonomous in their missions, recognize people, guide them through different environments and also avoid static and dynamic obstacles, such as chairs, bookcases or other people. Because of the growing interest on this application, two relevant contributions are reviewed in the following. Thrun *et al* [197] developed MINERVA, a robot that used two cameras combined with a laser sensor to build a complete map of the environment for the navigation process. Shen and Hu [172] presented ATLAS, a museum guiding robot that combined topological map building and appearance-based matching algorithms for localization. ATLAS also incorporated a human face detection algorithm [14] used to actively approach to new visitors.

One of the problems in topological SLAM is to decide when to add a new node in the map when new images are provided. Systems must be able to detect when new images correspond to already visited locations, so to existing nodes, or to new ones. This is the loop-closure detection problem particularized for topological map building and localization. Approaches focused in solving this problem have to be specially accurate since different images taken from different viewpoints or at different distances can represent the same node. In [3], Bayes filters were employed to calculate the probability of loop-closure detection each time a new image was acquired. If the matching process was successful, the node information was updated with the information retrieved from the last image. If not, a new node was added to the topological map. Nodes were characterized with words. Word were formed by incrementally combining similar SIFT features [112] stored in a visual vocabulary (database of SIFT features with their descriptors). These words were also used to determine the matching between acquired images and existing nodes.

More recently, Liu *et al* [108] described a novel scene recognition appearance-based method for omnidirectional vision. Viewed scenes were compared with stored scenes representing topological-map nodes. The method permits to recognize an already registered place (as a map node) or to add a new node to the topological map in case the scene has not been identified. For each scene, dominant vertical lines define the regions for segmentation. For each region, the average color value in the U-V space is extracted. This U-V average value and the width of the region delimited between two lines form the region descriptor. The descriptors were invariant in rotation and to some illumination changes. Scene matching between new scenes and existing nodes was performed computing the 2D euclidean distance between color descriptors and recursively comparing the widths of the regions according to an empirically determined inequality ($\frac{1}{3} < \frac{Width_1}{Width_2} < 3$). Every node was characterized with more than one image and it was assumed that occlusions caused by dynamic obstacles would not occupy more that a 30% of the total image.

Local Map-building Navigation Systems and Obstacle Avoidance

The strategies seen so far base their strength in a global description of the environment. This model can be obtained automatically by the robot, or in a previous human guided stage, but it has to be acquired before the robot begins the navigation. Since the early nineties, some authors have developed solutions where visual

navigation processes are supported by the on-line construction of a local occupancy grid. In vision-based navigation, the local grid represents the portion of the environment that surrounds the robot and the grid size is determined by the camera field of view. This local information can be used for a subsequent complete map construction or simply updated frame by frame and used as a support for on-line reactive navigation. Since robot decisions depend, to a large extent, on what the robot perceives at every moment in the field of view, these navigation techniques arise a disscusion about what can be considered *deliberative* and what can be considered *reactive* vision-based navigation techniques.

Badal *et al* reported a system for extracting range information and performing obstacle detection and avoidance in outdoor environments based on the computation of disparity from the two images of a stereo pair of calibrated cameras [9]. The system assumed that objects protrude high from a flat floor that stands out from the background. Every point above the ground was configured as a potential object and projected onto the ground plane, in a local occupancy grid called Instantaneous Obstacle Map (IOM). The commands to steer the robot were generated according to the position of obstacles in the IOM.

Gartshore *et al* [63] developed a map building framework and a feature position detector algorithm that processed images on-line from a single camera. The system did not use matching approaches. Instead, it computed probabilities of finding objects at every location. The algorithm started detecting the object boundaries for the current frame using the Harris edge and corner detector [82]. Detected features were back projected from the 2D image plane considering all the potential locations at any depth. The positioning module of the system computed the position of the robot using odometry data combined with image feature extraction. Color or gradient from edges and features from past images helped to increase the confidence of the object presence in a certain location. Experimental results tested in indoor environments set the size of the grid cells to 25 mm $\times$ 25 mm. The robot moved 100 mm between consecutive images.

Goldberg *et al* [68] introduced a stereo vision-based navigation algorithm for the rover planetary explorer MER, to survey and map locally hazardous terrains. The algorithm, first, computed epipolar lines between the two stereo frames to check the presence of an object, second, computed the Laplacian of both images and, third, correlated the filtered images to match pixels from the left image with their corresponding pixels in the right image. The work also included a description of the

navigation module GESTALT, which packaged a set of routines able to compute actuation, direction, or steering commands from the sensor information.

Gartshore and Palmer presented in [64] a novel approach for complete unknown environment visual exploration and map construction with a limited field-of-view vision system. Afterwards they extended this work to more complex environments [65]. No landmarks or way-markers were used, and once the navigation had started, there was no human interaction. The exploration agent had to act as a human might do, observing the current view of the environment, exploring it, and deciding in which direction to advance to explore new areas. The main issues of the incremental map building process were:

- Vertical edges were extracted from the current frame to define obstacle boundaries. In some cases, these edges did not correspond to obstacles, but to shadows or specularities.

- To discriminate shadows or specularities from real obstacles, a confident measure was assigned to every edge point. Such a measure was a function of the number of times the object had been seen and the number of times the same area had been viewed.

- Features were connected with lines. These lines could either correspond to objects or just be connecting lines traced for triangulation purposes.

- Lines were also labeled with a confidence of being an obstacle. According to [116]: a candidate point to be labeled as an obstacle can not intersect the line that joins the camera with a real obstacle. The confidence measures were recalculated when points labeled as obstacles were viewed from another point of view as occluding other real obstacles.

- Obstacles and triangulation information are stored in discrete grids.

Visual Sonar

In recent years, *visual sonar* has become an original idea to provide range data and depth measurements for navigation and obstacle avoidance using vision in an analogous way to ultrasound sensors. Therefore, the originality of the concept is not in the navigation process itself, but in the way the data is obtained.

Martens *et al* were pioneers in using the concept of visual sonar for navigation and obstacle avoidance [119]. Their ARTMAP neural network combined sonar data and visual information from a single camera to obtain a more veridical perception of indoor environments. Real distance to obstacles was calculated from distances measured in pixels between obstacle edges and the bottom of the image. This distance computation was based on Horswill's idea [86]: the image was divided in eight columns, and the distance, measured in pixels from the bottom of the image to the object edge in every column, was considered to be proportional to the real world distance from the robot to the detected object.

Lenser and Veloso exposed a new visual sonar-based navigation strategy for the ROBOCUP competition and the AIBO robots [105, 56]. AIBOs are dog-shaped robots that have a single camera mounted on their heads. The system segmented color images to distinguish floor, other robots, goals, the ball and other undefined objects. Once objects had been defined, lines were radiated from the center of the image bottom, every 5°. An object was identified if there existed a continuous set of pixels in a scan line which corresponded to the same item class. Distance from object edges to the focus of the radial lines defined the real distance from the robot to the obstacle. The system builded a local grid of the environment with the robot centered on it, and avoided obstacles using contour following techniques. Since error increased with distance, anything separated more than 2 m could not be properly measured, and consequently, the algorithm only considered obstacles closer than 0.6 m.

Choi and Oh detected obstacle boundaries in images where the diagonal Mahalanobis color distance changed abruptly over points situated in radial lines that emanated from the calibrated camera to the rest of the image [36]. The system assumed that floor color and lighting conditions were constant. Odometry information was used to transform position coordinates on the image plane into world coordinates over a local occupancy grid. The cells of the grid were labeled with a probability of being occupied by an obstacle. Experimental tests have been performed on cluttered offices and the local grid was constructed to support safe navigation. The paper also introduced the idea of omni-directional observation with a standard camera.

Martin computed depth from single camera images of indoor environments using also the concept of visual sonar [120]. The novelty of this method was the use of genetic programming to automatically discover the best algorithm to detect

the ground boundaries in a training phase. These algorithms were then combined with reactive obstacle avoidance strategies, initially developed for sonar, and later adapted.

3.3.2 Mapless Navigation

This section includes a representative collection of mainly reactive visual navigation techniques. Those strategies process video frames as they gather them, and they are able to produce enough information about the unknown and just perceived environment to safely navigate through it.

Prominent mapless visual navigation techniques here included are classified in accordance with the used vision technique or clue: optical flow, feature detection and tracking, environment appearance, and extraction of qualitative information from an image.

Optical Flow-based Navigation Systems

Optical flow can be roughly defined as the patterns of apparent motion of features in a sequence of images caused by the relative motion of the camera with respect to the environment. During navigation, the robot movement is perceived as a relative motion of the field of view, and, in consequence, it gives the impression that static objects and features move respect to the robot. To extract optical flow from a video stream, the direction and magnitude of translational or rotational scene feature movement must be computed at every pair of consecutive camera frames. Optical flow between two consecutive frames is usually represented by a vector for every pixel, where its norm depends on the motion speed and its direction represents the movement of the corresponding pixel in consecutive images. In some cases, the execution time and the computation resources required can be optimized by first extracting the image prominent features, such as corners or edges [173, 82], and then computing the optical flow only for these features. Image optical flow has been used by some researchers to implement reactive mobile robot navigation strategies, either for indoor or for outdoor environments. Object boundaries appear as regions with significant optical flow, and thus as regions to be avoided. Specularities or irregularities on the floor and textured floors also appear as regions with optical flow and therefore can be wrongly considered as obstacles causing errors during navigation.

Variations in the optical flow pattern or direction are used by Santos-Victor and Sandini to detect obstacles in a reactive, fast and robust approach for plane ground environments using a single camera [166]. Objects that arise from the ground plane cause variations in its normal optical flow pattern. To analyze and determine the presence of obstacles, the image flow field must be projected inversely onto the horizontal world plane. For translational motion, the projected flow must be constant for every point on the ground plane. Obstacles alter this assumption, presenting higher magnitudes or changes in the vector direction.

Camus *et al* [27] compute on-line the optical flow divergence from sequential wide angle frames to detect and avoid obstacles. Flow divergence is used for computing time to contact to obstacles in a qualitative way. To command the robot safely, one-dimensional maps are computed, where every heading direction is labeled with a potential risk of encountering obstacles.

Talukder *et al* [189] implement a novel and robust optical flow-based solution to detect the presence of dynamic objects inside the camera field of view. It is applicable to robots with translational and/or limited rotational movement. The algorithm assumes that moving objects cause a discontinuity in optical flow orientation and changes in its magnitude, with respect to the background pixels optical flow direction and magnitude. The system is developed and first tested using a single camera, and then using a stereo camera which provides depth information.

Some authors have proved that the combination of stereo vision, to obtain accurate depth information, and optical flow analysis provides better navigation results. Talukder and Matties extended [189] combining the stereo disparity field and optical flow to, first, estimate depth, second, to model the robot egomotion and, third, to detect moving objects of the scene [190]. In [25], stereo information is combined with the optical flow from one of the stereo images, to build an occupancy grid and perform a real-time navigation strategy for ground vehicles.

A simple and preliminary qualitative visual-based navigation system was proposed in [191] by Temizer and Kaelbling, under the DARPA-Mobile Autonomous Robot Software (MARS) program. Although, to the best of the authors knowledge, this work does not represent a real progress in the field, it deserves to be included in a survey due to its simplicity and efficiency. The starting point for this strategy is the computation of image edge maps by detecting Laplacian of Gaussian (LOG) zero crossings. A patch matching procedure is subsequently applied using the edge maps of consecutive frames to compute the corresponding optical flow. Finally, the

system turns away from zones of high optical flow, since they likely correspond to obstacles.

Visual navigation techniques based on optical flow have proved to be specially useful for *Unmanned Aerial Vehicles* (UAV) because optical flow provides the scene qualitative characteristics that can not be extracted in detail from single low quality images. Within this research trend, an important effort has been devoted to imitate animal behavior as far as the use and processing of apparent motion is concerned. Particularly, insects present a high degree of precision in their navigation and guidance systems, despite the simplicity of their nervous systems and small brains. Many authors have studied the way honeybees and other insects use optical flow to avoid obstacles and/or to navigate centered in the middle of corridors or narrow long ways. Experimental results found by Srinivasan *et al* [184] proved that bees fly balancing the path in the middle of tunnels, evaluating the apparent motion of images that perceive from both sides.

Van der Zwaan and Santos-Victor [207] implemented a UAV with a camera eye equivalent to an insect compound eye. The camera eye consisted of an array of photoreceptors, each one connected to an electronic Elementary Motion Detector (EMD). This EMD was able to calculate the local optical flow at its particular position. Contrast on optical flow calculations determined the presence of obstacles, while, identifying the EMD polar coordinates that gave the changes on optical flow measures permitted to construct a local map with the location of the obstacles.

Netter and Franceschini [144] also implemented a UAV with a camera eye assembled with an array of photosensors and their corresponding EMDs. The information given by the set of EMDs was used to determine the presence of obstacles. Furthermore, when the UAV flew at a constant speed and altitude, a reference optical flow distribution was calculated from the equation that models the velocity of the artificial retina. To follow the terrain, the system varied thrust and rudders position to adjust the online computed optical flow with the optical flow reference.

Nonetheless, the use of optical flow information in terrain following applications for UAV presents limitations if the aircraft flies at low altitude, at high speed or if it is landing, even more if the camera is facing the ground. In these cases, optical flow estimation loses accuracy. Recently, Srinivasan *et al* [185] presented a new system to increase accuracy in the optical flow estimation for insect-based flying control systems. A special mirror surface is mounted in front of the camera, which is pointing ahead instead of pointing to the ground. The mirror surface decreases the speed of

motion and eliminates the distortion caused by the perspective. Theoretically, the image should present a constant and low velocity everywhere, simplifying the optical flow calculation and increasing its accuracy. Consequently, the system increases the speed range and the number of situations under which the aircraft can safely fly. Particularly interesting is the work developed by Green *et al* [76], which describes the design of an UAV prototype called Closed Quarter Aerial Robot (CQAR) that flies into buildings, takes off and lands controlled by an insect-inspired optical flow-based system. This aerial vehicle incorporates a microsensor which weighs 4.8 grams, and is able to image the environment and compute the optical flow. The minimum flying speed of CQAR is 2 m/s, the turning radius is about 2.5 m and to avoid a detected obstacle it needs to turn about 5 meters before. Later, Green *et al* emphasized again the relevance of insect-based navigation strategies in an optical flow-based navigation system, for UAVs that fly in near ground environments such as tunnels, caves, inside buildings or among trees [75]. The navigation principles applied in both [75] and [76] come from equation 3.1:

$$F = (v/d)\sin(\theta) - \omega, \tag{3.1}$$

where F is the optical flow, v is the translational velocity, d is the distance between the robot and an object, w is the angular velocity, and θ is the angle between the direction of travel and the aforementioned object. Equation 3.1 models the fact that optical flow of close obstacles has greater magnitude than the optical flow of obstacles that are at longer distances. Furthermore, optical flow magnitude is maximum for obstacles situated orthogonally to the robot motion direction.

To finish with this research line, Srinivasan *et al* presented an overview of illustrative insect-inspired navigation strategies for different situations, and the implementation of those strategies in several robots to test their feasibility [186].

Following a different research line, Cornall and Egan pointed out preliminary results corresponding to the analysis of optical flow patterns. Images were recorded during the UAV flight and transmitted to a ground station to be stored and analyzed. Optical flow example images of translation, pitch, roll to left/right and yaw motion were computed off-line and primary conclusions presented in [41].

In urban missions, UAVs have to fly usually among buildings and at low altitude, avoiding obstacles situated at both sides or at the front, and making very steep turns or even U-turns at dead ends. This increases the possibility of crashing, thus

the need of a very precise and safe navigation strategy. Hrabar *et al* present in [89] a novel navigation technique for UAVs to fly in between of urban canyons. The authors report a high degree of effectiveness of the system combining stereo forward-looking cameras for obstacle avoidance and two sideways-looking cameras for stable canyon navigation. Since the method is applied on UAVs, everything detected at the front is considered an obstacle. The system projects 3D stereo data onto a 2D map and performs a growing region process to extract obstacles. The robot stops keeping constant the altitude or simply changes direction depending on its distance to the obstacle. Besides, the robot always tries to balance the optical flow from both sides, moving to the direction of larger optical flow magnitude. The system implements a hierarchical architecture. Collisions with obstacles in the front are more probable than on the sides, therefore the stereo output is given priority over the optical flow output. The authors also expose an alternative for implementing this kind of hybrid systems, using two forward-facing fisheye cameras that have lenses with a 190° field of view. In this last case, the central part of an image can be used for stereo front obstacle avoidance, and the peripheral part can be used for computing the optical flow.

Appearance-based Navigation

Appearance-based strategies consist of two procedures. First, in a pre-training phase, images or prominent features of the environment are recorded and stored as model templates. The models are labeled with a certain localization information and/or with an associated control steering command. Second, in the navigation stage, the robot has to recognize the environment and self-localize in it by matching the current on-line image with the stored templates. The main problems of appearance-based strategies are finding an appropriate algorithm to create the environment representation and defining the on-line matching criteria.

Deviations between the route followed in the guided pre-training phase and the route autonomously navigated yield different sets of images for the sequences recorded in both cases. Different images implies differences in the perception of the environment. Main researchers have focused their contributions on improving the way how images are recorded in the training phase, as well as on the subsequent image matching processes. There are two main approaches for environment recognition without using a map [123]:

- *Model-based Approaches.* They utilize pre-defined object models to recognize features in complicated environments and self-localize in it.

- *View-based Approach.* No features are extracted from the pre-recorded images. The self-localization is performed using image matching algorithms.

Matsumoto *et al* presented in [124, 123, 125] research results focusing on indoor route construction with standard or omnidirectional images. They defined correlation equations to model the concept of distance between images, and designed a new view creation procedure using stereo divergence for outdoor environments where light conditions change most often.

Zhou *et al* [214] utilized histograms to describe the appearance of pre-recorded indoor images. Color, gradient, edge density and texture histograms were extracted from images, and stored in a multi-dimensional histogram database. The recognition of the environment during the navigation stage was reached by matching the multi-dimensional histogram of the current image with the multi-dimensional histogram of the stored templates. Working with histograms has two main advantages: i) it saves computation resources and, ii) it is simpler and quicker than entire images-based correlation processes.

Haddad *et al* in [77] were pioneers on applying the *Potential Fields* method in vision-based navigation and obstacle avoidance strategies. Remazeilles *et al* used the concept of *Potential Fields* integrated in an appearance-based navigation method [157]. This system differs from typical appearance-based navigation strategies in the way that navigation was performed. The method defined an image database, which was a set of views built off-line, representing the whole navigable environment. When a navigation mission was defined, an image sequence corresponding to what the robot camera should see during the motion was extracted from the image database. The robot motion was the result of the on-line detection and matching process between the models included in the sequence and the perceived scenes. To navigate the environment, the robot tracked recognizable previously cataloged features. To fit these scene features in its field of view it used the attractive *Potential Fields* method to approximate them.

Morita *et al* reported in [139] a novel appearance-based localization approach for outdoor navigation. They extended their *Support Vector Machine* (SVM) -based algorithm, proposed in [140], to a novel SVM-based localization architecture that used vision information from panoramic images. The SVM localization process consisted

of two main stages: first, feature or object learning, recognition and classification, and second, scene locations learning based on the previous feature classification. In this work, the authors showed how panoramic images improve considerably training, matching and localization procedures, since the scenes are less dependent on the variation of the robot heading.

Image Qualitative Characteristics Extraction and Texture Segmentation

Reactive visual techniques for robot navigation and obstacle avoidance are often devised around the extraction of image qualitative characteristics and their interpretation. There are two main types of reactive visual obstacle avoidance systems:

- *model-based* obstacle avoidance systems, which need pre-defined models of known objects, and,
- *sensor-based* obstacle avoidance systems, which process every on-line sensor information to determine what could be an obstacle or what could be free space.

These strategies can be included in what is known as qualitative navigation. Reactive navigation systems based on qualitative information avoid as much as possible using, computing or generating accurate numerical data such as distances, position coordinates, velocity, projections from image plane onto real world plane, or contact time to obstacles. In general, a coordinated behavior-based architecture is needed to manage all qualitative image information and the subsequent reactions [7].

Changes on the imaging conditions, that is for example, illumination intensity, position of light sources or glossiness of the scene materials, are of particular relevance to this sort of navigation systems due to qualitative data or image texture have a critical dependence on unprocessed sensorial data.

As a consequence, and mostly for outdoor applications, the performance of certain visual navigation systems can be seriously limited depending on time, weather conditions, season, etc.. One of the earliest solutions to these problems came from [193]. In 1997, Lorigo *et al* proposed a very low resolution vision-based obstacle avoidance system for unstructured environments [109]. The novelty of the solution was the construction of three simple modules that based the object detection

criteria on brightness gradients, RGB color and HSV (hue, saturation, value) information. The goal of this approach was to safely navigate with no destination point or pre-designed mission. The method assumed that all objects stayed on the ground plane, and that closer objects were in the bottom of the image while further objects were on the top of the image. Apart from the three modules working on brightness, RGB and HSV, a fourth one analyzed simultaneously their results to extract possible object boundaries. Afterwards, this information was used to generate motion commands.

The combination of a camera and other sensors such as laser or sonar has been applied in some reactive approaches to increase safety and the capabilities of the navigation process. CERES [32] was a behavior-based architecture that combined seven ultrasound transducers and a single grayscale camera. The vision module applied a Canny filter to extract edges from images. Edges are a clear evidence of the presence of obstacles. However, the floor carpet texture of the author test environment generated edges that could be wrongly considered as obstacles. To avoid this misbehaviour, a threshold was imposed to eliminate false edges. The system transformed distances over images to real world distances using a rough camera calibration algorithm. For this particular case, the authors knew that the first fifth portion of the image, from bottom to top, corresponded to the closest 20 cm of the scene and that the other four fifths portion corresponded to the next real world 26 cm. Consequently, all those edges found in the first fifth of the image (bottom) were considered as obstacles to be avoided while the edges on the rest of the image (top) were considered to be far enough so as to be taken into account. Sonar is used to keep distance to the walls.

Other authors prefered to use a bi-level image segmentation process to segregate floor from objects [114]. Floor detection permits determining where the free navigable space is. In the ROBOCUP competition, the detection of the opponent robot and the ball becomes a challenging task to properly play the game. Fasola and Veloso [57] proposed to use image color segmentation techniques for object detection, and gray-scale image processing for detecting the opponent robots. Cherian *et al* [35] proposed a ground plane detection method run on three stages: i) to build the 3D depth map of the scene, the YC_bC_r format of images was convolved with 17 different filters forming 34×5-dimensional feature vectors ([1]) representing

[1]each vector was composed by 34 dimensions but vectors of the 4 neighbors were include for each feature in order to include information regarding the depth at surrounding pixels

the depth information at each image point, ii) to correct irregularities in the depth information, images were divided into regions of perceptually similar textures, assuming that regions with similar textures lied on the same plane, and iii) coplanar regions were merged to form the complete ground plane.

The concept of fuzzy navigation, and particularly using visual sensors, has been used by several authors, combining the extraction of qualitative information from video frames with qualitative navigation algorithms based on fuzzy rules. One example of these techniques comes from Howard *et al* [88]. Their system is focused basically on ensuring a safe navigation through irregular terrains. It was assumed that the terrain could present rocks and variations on its slope. A region growing method based on edge detection and obstacle identification was used to detect rocks on the ground, while the terrain slope was calculated using existing techniques to retrieve 3D information and real Cartesian coordinates from a stereo pair of images. The size and number of rocks and the slope of the terrain were then classified by an algorithm that used fuzzy terms such as big, small, rocky terrain, flat, sloped, steep, etc... Since the final goal of this system was to mimic as much as possible the human criteria used to classify a terrain, the system was trained by an expert which evaluated images taken from the robot point of view and judged the ability of the robot to navigate through the terrain. The difference between the human classification and the one done by the robot is an indication of the system optimality.

Navigation Techniques Based on Feature Tracking

Many of the existing navigation approaches are based on detecting and tracking the environmental data perceived by the onboard sensors. When using range sensors, the readings themselves can be used as the tracking data. However, with visual sensors, the process of finding such trackable data usually starts by tracking the most stable and significant points in the image, so called *features* or *corners* in the early techniques. As a matter of fact, detection and tracking of visual features is a mature but still growing area in the computer vision community nowadays.

1) Feature Detectors and Descriptors. Feature descriptors are categorized using the concept of distinctiveness. Distinctiveness is related to the size of the neighborhood window captured by the feature descriptor and also to the amount and type of information processed and extracted from it. Distinctive features can be tracked without using a motion model and more accurately than non-distinctive

features. Harris and Kanade-Lucas-Tomasi algorithm (here called KLT) ([173]) methods are early and fast techniques to find and/or track little discriminative features. Harris corners ([82]) corresponded to image patches whose $C(x, y)$ matrix eigenvalues were considerable high, where $C(x, y) = \begin{bmatrix} G_x & G_{xy} \\ G_{xy} & G_y \end{bmatrix}$ with G_x, G_y and G_{xy} being the convolution of I_x^2, I_y^2 and $I_x I_y$ with a zero mean Gaussian kernel (I_x, I_y are the image I partial derivatives along the x and y directions). Harris corners are invariant to rotation and illumination transformations, and have a significant repeatability rate, but the method fails when there are scale changes between images. In order to overcome this pitfall, Mikolajczyk and Schmid used the Harris operator to find features for which the Laplacian achieved a maximum over a scale space ([133]).

Shi & Tomasi ([173]) showed that good features were those corresponding to image patches whose minimum eigenvalue of the $C(x, y)$ matrix was above a defined threshold.

Lowe developed the SIFT *(Scale Invariant Feature Transform)* method to extract high discriminative image features ([112]). A Gaussian scale space is generated convolving the original image with a set of Gaussians for different scales. This Gaussian scale space generates a set of Differences of Gaussians whose local extrema define the features location. Lowe approach combines this detector with a descriptor of 128 components, containing the gradient location and orientation of different points centered around the proposed feature. SIFT is robust to image scaling, rotation, illumination changes and camera view-point changes. It has a high degree of repeatability and it is remarkably robust to noise.

Other feature detectors such as SURF *(Speeded-Up Robust Features)* ([13]), FAST *(Features from Accelerated Segment Test)* ([163]) and CenSurE *(Center Surround Extremas)* [2] intend to increase speed and performance. SURF reduces the number of interest points, computes scale, rotation and viewpoint invariant feature descriptors with considerably good results and outperforms SIFT in terms of computation time. Features are located where the Hessian Matrix eigenvalues of each pixel have the same sign. Descriptors show how the pixel intensities are distributed within the neighborhood of each feature in different scales. SURF uses integral images instead of the original images and a 64-items descriptor to reduce computation time. CenSurE authors claim that their features have better computational characteristics than other scale-space detectors, and that they are optimum for visual

odometry since they result in longer track lengths, there are fewer frames where images fail to match, and present better motion estimates.

FAST is based on SUSAN ([183]) and it considers an image point to be a feature if a minimum of 12 pixels can be found on a circle of fixed radius around the point such that these pixels are all brighter or darker than the central point. The feature descriptor consists of a vector containing the intensities of the 16 pixels surrounding the point. FAST is more suitable for video streams at high frame rates or images taken from rapid motions, but it is vulnerable to noise and it depends on a pre-defined threshold. Mikolajczyk and Schmid [134] compared the performance of different descriptors for image local regions. Experimental results of different matching approaches used to recognize the same region in different viewing conditions showed that SIFT yields the best performance in all tests. Later, Tuytelaars and Mikolajczyk published an extensive survey of feature detectors, comparing the performance of the most important approaches ([206]).

2) Techniques for Land and Aerial Feature-Based Localization and Navigation. Visual techniques for detecting and tracking significant elements of the scene, so called features, have been extensively improved over the last years and used as the first step for localization and/or navigation purposes. Autonomous navigation and obstacle avoidance, localization or mapping systems benefit from high speed feature computation and stable and repeatable features. Of course, feature descriptors invariant to changes on rotation, scale or viewpoint are also preferable.

Many times, the control system divides a feature tracking task into two subproblems [202]:

- *motion detection*, which, given a feature to be tracked, identifies a region in the next frame where it is likely to find such a feature, and,
- *feature matching*, by which the tracked feature is matched within the identified region.

In general, feature tracking-based navigation approaches do not comprise an obstacle avoidance module, but this task has to be implemented by other means. Although video tracking and mobile robot navigation belong to separate research communities, some authors claim to bridge them to motivate the development of new navigation strategies. Some authors center their research in detecting and

tracking the ground plane across consecutive images, and steering the robot towards the free space. Mourikis *et al* based their system for planetary vehicle positioning and real time speed computation on Harris corners and the normalized correlation ([141]). Zhou and Li [215], and Dao [45] computed and used the homography matrix to detect and track the ground plane over previously tracked image corners or edges using the Harris corner detector. In a more recent work, other authors prefer to combine the concept of feature tracking with stereo 3-D environment reconstruction. In [165], stereo vision was used in a novel navigation strategy applicable to unstructured indoor/outdoor environments. This system was based on a new, faster and more accurate corner detector. Detected features were 3D positioned and tracked using normalized mean-squared differences and correlation measurements.

Localization or motion estimation are robot capabilities typically performed by combining wheel odometry (for ground robots), gyroscopes and accelerometers. Wheel odometry is inaccurate and untrustworthy for irregular terrains since wheels can slip, deflate or sink and inertial sensors are prone to drift. *Visual odometry* is a good complement for wheel odometry or inertial sensors. Feature tracking techniques are used for visual odometry estimation tasks. Image features are used as reference points, and their estimated motion in a sequence of successive images is properly employed to infer the robot motion and/or position. Nister *et al.* estimated the motion of mobile robots tracking nondistinctive Harris corners ([145]) and supposing very small displacements between consecutive images.

Stereo camera pair configurations give some advantages to single camera systems in the sake of computing visual odometry. Last approaches show significant advances in terms of speed, accuracy in the position estimation and reliability respect to early solutions presented by Matthies [126], Olson *et al* [148] or to other more recent approaches [34]. Howard *et al* proposed a new algorithm to calculate the robot motion using a stereo pair of cameras [87]:

1. A stereo pair of images were rectified and prefiltered to eliminate high frequency components.

2. A disparity image was computed by matching points in the left/righ images. The prefiltered and the disparity images were the inputs available to the visual odometry algorithm.

3. Harris or FAST features were detected in consecutive images, their world

coordinates were computed from the corresponding disparity images and descriptors, including the surrounding $m \times m - 1$ pixels.

4. The sum-of-absolute differences (SAD) between all pairwise feature descriptors formed the *score matrix* S, all minimum SAD scores determined the matches between features in consecutive stereo images.

5. Outliers were filtered out searching for inconsistent matches, were the consistency term is closely related with the discrepancy in the distance (expressed in world coordinates) between two features measured in two time-consecutive stereo images.

6. The camera motion is estimated by minimizing the image reprojection error for all matches in the set of feature correspondences.

Authors discussed the possible convenience of using scale invariant features such as SIFT or SURF in order to overcome the loss of matches in large motions. However, they noted that scale invariant features would notably increase the time of execution, reducing the real-time performance.

Special characteristics of SIFT features qualifies them to be, in general, one of the best solutions for performing visual odometry tasks. Parra *et al* proposed in [151] a robust visual odometry algorithm by means of SIFT feature tracking using a stereo pair of cameras. First, SIFT features were detected and traced in consecutive stereo pairs of frames. Second, their world coordinates were calculated using the camera parameters and the epipolarity properties. Third, the camera motion was simplified assuming translation and only a rotation in the vertical axis y. Fourth, RANSAC was used to estimate the rotation matrix and the translational vector that characterized the camera movement in two consecutive stereo pair of images. The $3D$ feature coordinates at instants t and $t + 1$ were used as input data in the RANSAC algorithm.

Johnson *et al* analyzed in [92] two stereo visual odometers, $MER - VO$ (Mars Exploration Rover Visual Odometer) and $MSL - VO$ (Mars Science Laboratory Visual Odometer). The $MSL-VO$ was an update of $MER-VO$ and overcame the shortcomings of its previous version. More specifically, $MSL - VO$ improved the run-time of the algorithm, increased the trackable number of features and eliminated the initial motion estimation dependable with the wheel odometry. This approach employed Harris corners as image main features and it tracked them in subsequent

stereo image pairs using the Pseudo Normalized Cross Correlation (PNC) and the feature $3D$ world coordinates. Finally, the Maximum Likelihood Estimation using the $3D$ point covariances was employed to estimate the camera motion.

Roberts *et al* [161] estimated the visual odometry computing the optical flow from tracked Harris scene features. Scenes were gridded in cells of 20×20 pixels. Harris corners were tracked across consecutive frames with the KLT feature tracker. A velocity model was fit for every feature trajectory in order to compute the optical flow that was finally used to estimate velocity and pose. The tracker was used in a memory-based learner that compared estimated visual odometry data with the ground truth computed with laser scan matching.

SIFT method stands out among other image feature or relevant point detection techniques, and nowadays has become a method commonly used in landmark detection applications. During the robot navigation process, detected invariant features are observed from different points of view, angles, distances and under different illumination conditions and thus they become highly appropriate landmarks to be tracked for navigation, global localization [170] and robust vision-based SLAM performance [169].

Stephen *et al* performed global simultaneous localization and mapping in mobile robots tracking distinctive visual $SIFT$ landmarks in static environments ([187]). Rodrigo *et al.* ([162]) combined the homography computed using a set of relevant SIFT features with a collection of nondistinctive features to perform a robust navigation algorithm.

Time to collision can be computed from consecutive image feature correspondences that are perfectly consistent with the epipolar geometric constraints. Cohen and Byrne sustained that time for collision was a function of the estimated motion (for example via inertial sensors), the intrinsic calibration matrix and the inlier position, if it is compliant with the epipolar geometry [40].

Pears and Liang use homographies to track ground plane corners in indoor environments, with a new navigation algorithm called *H-based Tracker* [153]. The same authors extend their work in [107] using also homographies to calculate height of tracked features or obstacles above the ground plane during the navigation process.

The accuracy of the navigation strategy must be a strategic point in aerial motion where the speed is high, the processing time must be reduced and the tracking process needs to be more accurate. In [147], Ollero *et al* propose a new image tracking strategy that computes and uses a homography matrix to compensate the

UAV motion and detect objects. This system improves their previous work [59] maintaining the tracking success despite the number of attempts is reduced. Localization and geo-location of observed elements, so called targets, from Unmanned Aerial Vehicles (UAV) has been also a field of interest and application of feature tracking procedures. The world coordinates of a certain image point corresponding to a target observed from a flying UAV can be calculated without applying the constraint of a flat terrain, knowing, i) the approximate altitude of the camera with respect to the target (given, for example, by an altimeter), ii) the vehicle absolute world position given by, for example, a GPS, iii) the rotation and translation matrices that transform the camera coordinate system into the world coordinate system, and finally, iv) the intrinsic and extrinsic camera parameters [79]. To geolocalize targets from UAVs, Han and DeSouza [79] track image features using the differential optical-flow and the KLT tracker. Feature world coordinates are then computed assuming that all required parameters are known. Afterwards, SIFT features are tracked on consecutive images and their world coordinates are computed to reffine the previously estimated UAV height. Height estimation accuracy can be improved by using a stereo camera pair. Çelik *et al* [30] proposed a new visual SLAM technique tested on indoor Micro Aerial Vehicles (*MAV*). As in many other previous visual SLAM solutions, main image features were designed to be the necessary landmarks for the motion tracking and localization process. Harris corners were initially extracted but they were not suitable for tracking agile motion. Then, it was used the Shi and Tomasi algorithm for both performances, feature detection and tracking. Features that coincided with corridor (or equally for ceiling) lines were potential good landmark candidates for the SLAM computation. Corridor o ceiling lines are parallel in scene but intersect in a point (so called the *infinity point*) in the image plain. By exploiting the geometrical properties of these lines, using the features that were detected on them and knowing the vehicle altitude, the system could calculate depth and bearing. This approach differs from optical flow methods in that depth measurement does not need a successive set of images.

Supporting vision information with GPS data in outdoor environments is another possibility of increasing reliability in position estimation. Saripalli and Sukhatme combined a feature tracking algorithm with GPS positioning to perform a navigation strategy for the autonomous helicopter AVATAR [130]. The vision process combines image segmentation and binarization to identify pre-defined features, such as house windows, and a Kalman filter-based algorithm to match and track these windows.

3) Underwater Applications. Several techniques have been developed for underwater environments. Some of them are of general application, such as image mosaicing systems, and others are more application oriented, such as the systems for pipeline or cable tracking. Sea floor mosaicing permits the robot to self-localize and thus identify its motion model. It is usually based on feature identification and tracking using texture-based operators and correlation-based procedures [62]. *Pipeline or cable tracking* is an essential issue for accurate maintenance of thousands of kilometers of telecommunication or power cables between islands, countries and continents. In particular, unburied cables can be tracked using vision techniques. The first approaches to cable tracking were based on edge detectors and Hough transform, but they were unable to perform real-time cable tracking at video rates [160, 78, 122] by that time. Grau *et al* [74] proposed a cable or pipe tracking system based on the generation of different texture groups and the segmentation of images in regions with similar textural behavior. Foresti and Gentili [61] implemented a robust neural-based system to recognize underwater objects. Balasuriya and Ura [11] increased and improved the robustness of the existing systems by solving the eventual loss of the cable with dead-reckoning positioning prediction combined with 2D models of the cable. In a recent work, Antich and Ortiz [4] present a control architecture for AUV's navigation based on a cable tracking algorithm that looks for edge alingments related with the cable sides. Finally, the same authors included a new sonar-based algorithm in the vision-based cable tracking architecture to escape from trapping zones [6].

Moving-target vision-based tracking strategies have also become a motivating research trend, specially to improve current fish shoal detection and tracking techniques. Between 2000 and 2001 some relevant solutions were presented by Silpa-Anan *et al* [176] and Fan and Balasuriya [55], respectively. Fan and Balasuriya [55] presented a process based on two parallel stages: object speed calculation represented with optical flow, and moving objects positioning with template-matching techniques. Rife and Rocks went a step forward implementing a system capable of recognizing and tracking only jellyfish [159].

Estimation of camera motion in underwater unstructured environments with no defined references, such as cables or pipes, becomes another complicated and challenging navigation problem. In this type of navigation strategies, references have to be defined, found in the image, and tracked. There are fundamentally three methods that are used for this purpose: optical flow, feature tracking or gradient methods.

Optical flow-based methods and feature tracking-based methods can cause failure in algorithms due to scattering effects, bad image quality or deficient illumination under the sea. Gradient methods use scene properties such as depth, range, shapes or color intensity, they are computationally more efficient and more accurate [44]. *Station keeping* is one of the problems that can be solved estimating the motion of the camera. Station keeping consists in holding the robot around a fixed position on the undersea floor that has a special interest at that moment. The AUV will hover around the interest point maintaining the center of the camera pointing on it. Examples of outstanding related solutions can be found in [143, 110] and [42].

IPT-based Obstacle Detection Approaches

Some authors exploited the concept of IPT to design reactive navigation and obstacle avoidance algorithms.

The IPM *(Inverse Perspective Mapping)* is a particular use of the Inverse Perspective Transformation concept. The IPM techniques allow to remove the perspective effect of lines that are parallel in the real world but converge into the vanishing point in the image. Each image pixel is re-mapped, and a new array of pixels is created where the lines in perspective are transformed into straight lines and objects are distorted. It represents a top view of the original image, like the projection of the whole scene onto a planar surface.

To detect obstacles, Mallot *et al* [115] analyzed variations on the optical flow computed over the Inverse Perspective Mapping (*IPM*) of consecutive images. Bertozzi and Broggi [16] projected two stereo images onto the ground applying the *IPM* concept. Then, they subtracted both projections to generate a non-zero pixel zone that evidenced the presence of obstacles. Batavia *et al* [12] used the *IPT* and the camera ego-motion to predict future frames and compare them with the corresponding new real frames. Differences between the predicted and real images showed the presence of obstacles. The system was designed to detect vehicles in the blind spot of the cars rear-view mirror. Shu and Tan [174] also employed the *IPM* to detect road lanes for self-guided vehicles. Ma *et al* [113] presented an automatic pedestrian detection algorithm based on *IPM* for self-guided vehicles. The system predicted new frames assuming that all image points lay on the floor. The distorted zones of the predicted image corresponded to objects. Simond combined the *IPM* with the computation of the ground plane super-homography from road lines

to discriminate obstacles from road in an autonomous guided vehicle application [181].

3.4 Conclusions

In the last decades, vision has become one of the most cheap, challenging and promising way that robots have to perceive the environment. Accordingly, the number of navigation approaches based on vision sensors have increased exponentially. Visual navigation techniques have been applied on almost all environments and in all kind of robots. The most outstanding pieces of work related with visual navigation from the early nineties until nowadays have been included in this chapter. Map-based navigation techniques have been contrasted with those systems that do not need a map for navigation in an attempt to gradually proceed from the most deliberative navigation techniques to the most pure reactive solutions.

Tables 3.3, 3.4, 3.5 and 3.6 show an overview of the most outstanding publications referenced in this chapter, from the late nineties to present. The list has been sorted by type of vehicle to facilitate analysis and comparison of the different strategies used in each of these vehicles. Concerning the different types of robots, the next general considerations can be drawn:

- Ground robots do not span the whole amount of applications revised in this chapter but cover almost all the considered strategies. Apparently, some strategies seem to be exclusive of ground robots because they are rarely found in aerial or underwater vehicles. This is the case of:
  - visual SLAM systems, because the computation of the environtment model seems to be feasible only for indoor scenarios,
  - homography-based navigation systems, because of their dependency on floor detection and tracking, and finally,
  - visual sonar systems and human pre-guided map building systems.
- The use of UAVs has significantly grown during the last decade and, as a consequence, navigation solutions for this kind of vehicles have improved in safety, accuracy and scope. The vast majority of UAVs use mapless navigation systems. We should highlight here the insect-inspired solutions for optical flow

processing as well as for feature tracking and detection. Some of these aerial robots have also gained in accuracy, operativity and robustness incorporating compound cameras or camera eyes.

- Visual navigation systems for AUVs have to cope with the special characteristics of undersea light propagation. Researchers have mostly focused on developing and/or evolving general visual navigation techniques based on feature tracking, mainly for mosaicing applications. Researchers have also focused on devising application-oriented navigation strategies, in many cases for tracking underwater cables or pipelines.

However, particularly focusing on the solutions adopted in mapless (reactive) systems, some evident shortcomings arise from the approaches analyzed in this survey, suggesting several possibilities for improvements or additive work:

- Normally, reactive solutions are proposed so as to solve either the navigation or the localization problem, but none of the analyzed pieces of work tries to solve both problems at the same time using a unique sensor and a unique visual information source.

- Solutions based on the IPT or IPM usually need to back-project the whole image onto the ground to remove the perspective effect caused by the imaging process and to infer where the obstacles are. Back-projecting all the image points can be unnecessary since maybe not all of them will be useful. Contrarily, back-projecting only a certain number of relevant image points could save execution time and focus the weight of the algorithm to those significant points of the image.

- Other solutions based on textures or appearance can fail under changes on illumination, shadows, scenarios with highly textured floors, inter-reflections or specularities unless the algorithms themselves cope explicitly with these optical phenomena. For example, some road line tracker algorithms addressed to automatic car-driving need to previously find lines in the image that converge to the vanishing point, which can be a difficult task if there is no sufficient contrast between the road and the surroundings.

- Solutions based on optical flow require to find differences on it in the different parts of the scene to infer the presence of obstacles.

- Navigation systems based on exploiting the properties of coplanar points, related in consecutive images through homographies, obligate to find the different planes present on the framed scene, segregating the ground plane from the rest.

- Focusing the navigation and the localization problems from the perspective of dealing with environmental information characterized by 3 dimensions is more difficult and more demanding than planning both problems using scene data only in the 2D ground plane.

3.5 Objectives and Contributions

Taking into consideration all the inconveniences exposed in the previous section, the challenge will be designing an architecture that can manage the tasks of obstacle detection, reactive navigation and localization, trying to accomplish the next objectives:

1. Discriminating obstacles from the ground. Obstacles in front of the robot must be detected regardless the characteristics of the environment where the robot has to move. The designed method has to be based only in the imaging geometry and frames have to be captured with no particular restrictions in the angle between the optical axis and the ground plane. The system has to overcome, to a certain extent, common scenarios with different planes, reflections, shadows, different textures or obstacle shapes and be as fast as possible to run on-line.

2. The detected obstacles have to be positioned in a local occupancy map centered in the robot pose to be used for safe navigation. The objective is not to create a perfectly accurate metric map but to find the way to represent in a qualitative manner which zones of the robot vicinity are occupied and which are free. Steering orders will point towards those zones of the map free of obstacles.

3. The final purpose of this navigation module must be to steer the robot towards several goal points avoiding all collisions. Once the obstacle detection module has proven to work properly, a navigation strategy would be needed in order to define which are the best decisions to be taken at each moment, depending

on each situation and with the final objective of reaching the programmed target/s. The strategy must consider important issues, such as: a) which are the optimum dimensions of the local map, b) where are the directions free of obstacles, c) which of these permitted directions must be taken to steer the robot towards the goal, or d) which would be the best way to infer the robot pose while it is moving through the environment.

4. No additional sensors or data sources must be used to perform the localization task. In order to reduce cost and hardware/software complexity, the robot pose must be computed from the same sensor and data gathered to detect obstacles. The localization algorithm must correct, in the best possible way, the robot position computed by dead-reckoning. minimizing drifts and inaccuracies as much as possible.

5. Since our navigation algorithm is based on the extraction and classification of image features, evaluating and comparing different feature tracking strategies becomes necessary to find the one that best combines performance and speed in our particular approach.

Consequently, the design presented in this thesis focuses on giving response to all the aforementioned objectives. The main parts of our proposal can be abstracted as:

1. An image point classifier: image main features are tracked across consecutive frames and classified as either obstacle or ground points using a new algorithm based on the IPT. Obstacle points are used for obstacle detection and avoidance while ground points are used for robocentric localization.

2. Obstacle avoidance and reactive navigation: the obstacle detection and avoidance procedure computes the edge map of the processed frames and edges comprising obstacle points are discriminated from the rest, emphasizing the obstacle boundaries; then, the points where obstacles touch the ground are incorporated into a qualitative occupancy map which represents the vicinity of the robot included in a ROI (Region of Interest); finally, the system computes the steering vector towards world areas free of obstacles. The method has been inspired on the visual sonar algorithms and on the Vector Field Histogram method [19], but here adapted for a vision sensor. These techniques

have been integrated in a reactive navigation strategy aimed to guide the robot safely in a mission way-point oriented where the robot is commanded to attain a goal point from a given start (i.e. previous) point.

3. Localization: the features classified as ground are fused in an EKF context together with the robot position obtained by dead-reckoning. Ground landmarks are added to the EKF as they are seen, or removed if they disappear from the field of view. The continuous releases of the EKF vector state stabilizes the ground landmark coordinates and gives the consecutive robot positions. The aim of this localization process is to improve, as much as possible, the robot pose data given by the proprioceptive sensors. Pose errors given by dead-reckoning information are especially significant in long routes, so localization experimental work will be focused on estimating trajectories as long as possible in the best accurate manner.

Table 3.3: Summary of the most outstanding visual navigation approaches from the late 90's to the present (1)

Authors	Type of vehicle	Category	Strategy	Type of visual sensor
[177, 178]	Ground	Map building	Visual SLAM. Landmarks localization and tracking	Single standard camera
[180, 179]	Ground	Map building	Visual SLAM. Landmarks extraction and occupancy grids	Stereo cameras
[46]	Ground	Map building	Visual SLAM. Map feature extraction	Single standard camera
[47]	Ground	Map building	Visual SLAM. 3D sparse mapping of interest points	Single wide angle camera
[116]	Ground	Map building	3D construction of an occupancy grid	Single standard camera
[200]	Ground	Map building	3D high density map and object recognition	Single standard camera
[98]	Ground	Map building	Human guided pretraining	Stereo cameras
[164]	Ground	Map building	Human guided pretraining	Single wide angle camera
[210]	Ground	Map building	Topological map	Omnidirectional camera
[66]	Ground	Map building	Topological map	Omnidirectional camera

Table 3.4: Summary of the most outstanding visual navigation approaches from the late 90's to the present (2)

Authors	Type of vehicle	Category	Strategy	Type of visual sensor
[102, 158]	Ground	Map building	Topological map	Single standard camera
[9]	Ground	Map building	Local occupancy grid	Stereo cameras
[63]	Ground	Map building	Local occupancy grid	Single standard camera
[68]	Ground	Map building	Local occupancy grid	Stereo cameras
[64]	Ground	Map building	Local occupancy grid	Single standard camera
[119, 120, 56, 36, 105]	Ground	Map building	Visual sonar	Single standard camera
[166]	Ground	Mapless	Optical flow	Single standard camera
[27]	Ground	Mapless	Optical flow	Single wide angle camera
[189, 190]	Ground	Mapless	Optical flow combined with stereo information	Stereo cameras
[191]	Ground	Mapless	Optical flow	Single standard camera
[123, 125, 157]	Ground	Mapless	Appearance-based method	Standard or omnidirectional single camera
[140]	Ground	Mapless	Appearance-based method	Panoramic camera

Table 3.5: Summary of the most outstanding visual navigation approaches from the late 90's to the present (3)

Authors	Type of vehicle	Category	Strategy	Type of visual sensor
[197]	Ground	Map building	Museum guiding robot: complete 3D map	Single standard camera
[172]	Ground	Map building	Museum guiding robot: topological map	Single standard camera
[109, 32]	Ground	Mapless	Image qualitative characteristics extraction	Single standard camera
[114]	Ground	Mapless	Image qualitative characteristics extraction	Single standard camera
[88]	Ground	Mapless	Image qualitative characteristics extraction	Stereo cameras
[107, 45, 153, 215]	Ground	Mapless	Features tracking; homography	Single standard camera
[165]	Ground	Mapless	Features tracking; homography	Stereo cameras
[111, 169]	Ground	Mapless	Features tracking: SIFT	Single standard camera
[207, 144]	UAV	Mapless	Optical flow: insect inspired (EMD)	Camera eye
[185]	UAV	Mapless	Optical flow: insect inspired (EMD)	Single standard camera

Table 3.6: Summary of the most outstanding visual navigation approaches from the late 90's to the present (4)

Authors	Type of vehicle	Category	Strategy	Type of visual sensor
[75, 76]	UAV	Mapless	Optical flow: insect inspired (EMD)	Single mini wireless camera
[89]	UAV	Mapless	Optical flow: insect inspired	Stereo cameras looking forward combined with two sideways looking cameras
[130]	UAV	Mapless	Features tracking	Single wide angle camera
[53, 67]	Amphibious	Map building	Visual SLAM. 3D total map building	Trinocular stereo cameras
[147]	AUV	Mapless	Features tracking; homography	Single standard camera
[4, 6, 160, 122, 74, 78]	AUV	Mapless	Cable tracking	Single standard camera
[110, 143]	AUV	Mapless	Station keeping	Single standard camera
[85, 118, 60, 69, 212]	AUV	Map building	Underwater floor mosaicing	Single standard camera

Part II

The Visual Navigation Approach

This part of the document concentrates the whole navigation and localization process developed in this thesis. Chapter 4 details the image feature classifier, the obstacle detection method and how the proposed algorithm builds the occupancy grids for reactive navigation. Chapter 5 presents the assessment of different feature detectors and trackers particularly applied to the obstacle detector introduced in this thesis. Some prior experiments testing the effectiveness of the obstacle avoidance algorithm running on-line in a moving platform are also presented in this chapter. Chapter 6 is dedicated to the completed navigation strategy that must guide the robot to cover missions from one starting point to one or several goal points. The obstacle detector has been integrated in the navigation architecture to provide a security module. The strategy generates the correct motion orders to avoid obstacles and to steer the robot towards the goals, overcoming different additional navigation problems, such as detecting and escaping from trapping zones or moving in relatively cluttered environments.

Chapter 7 outlines the EKF(*Extended Kalman Filter*)-based localization algorithm using the ground points. Its principal objective is to correct the pose estimates given by the robot odometers using the same camera and data sources than in the obstacle avoidance task.

CHAPTER 4

OBSTACLE DETECTION AND AVOIDANCE

This chapter begins introducing the VFH (*Vector Field Histogram*) method as the main inspiration of our algorithm for building the local occupancy map, evaluating the free and occupied zones surrounding the robot and computing the steering vector in the early navigation experiments.

Next, a brief outline of the perspective transformations from the image geometry point of view is given. These concepts are the basis of our feature classifier and obstacle detector, and are basic to introduce the whole process.

The process for building the local map runs in 3 main steps:

1. Image features are detected and tracked across consecutive pairs of frames. Features are classified as obstacle or ground using a new technique based on the IPT (*Inverse Perspective Transformation*). The classifier performance is evaluated using ROC (*Receiver Operating Characteristic*) curves.

2. Edge maps of consecutive frames are computed and edges containing obstacle features are discriminated from the rest of edges. Discriminating the obstacles edges from the rest permits: a) delimiting the obstacle boundaries, b) detecting where these obstacles contact the ground plane, and c) having a qualitative idea of the scene.

3. The world coordinates of those image points that correspond to obstacle-to-ground contact points are calculated applying the IPT. These world coordinates are placed in a semicircular local occupancy grid, centered in the robot position and with a fixed radius. This local map is continuously updated and used for a safe autonomous navigation.

This chapter also includes initial experimental results to test the obstacle detection and avoidance algorithm in autonomous missions. These first tests only included moving without a predefined goal point avoiding the obstacles encountered in the scene.

4.1 The Vector Field Histogram

The VFH [19] is a classic approach widely used in reactive mobile robotic infrastructures equipped with range sensors. From a general point of view, the VFH consists in building a one dimensional polar histogram to represent the immediate environment around the robot. The histogram comprises k angular sectors, where each sector defines a polar direction with respect to the robot center and it is labeled with a value H_k called the *polar obstacle density*:

$$H_k = \sum(m_{i,j}), \tag{4.1}$$

where $m_{i,j}$ is calculated for all the cells contained in each angular sector of k degrees, as follows:

$$m_{i,j} = c_{i,j}^2(a - bd_{i,j}), \tag{4.2}$$

where a and b are constants, $d_{i,j}$ is the distance from the cell to the center of the robot and $c_{i,j}$ is the *certainty value* of the cell. For example, using ultrasound sensors, $c_{i,j}$ is incremented in one unit each time there is a sensor reading that fails in that cell and the cell is in the acoustic axis of the sensor. The idea is that cells with repeated or continuous presence of readings will be most likely occupied by obstacles while cells that present fewer readings or readings with a random appearance would show noise data, being 0 the value of $c_{i,j}$ if no readings come from that cell.

The polar histogram represents the *polar obstacle density* at each polar direction, from 0° to 360°. Sectors with peaks on the *polar obstacle density* values are labeled

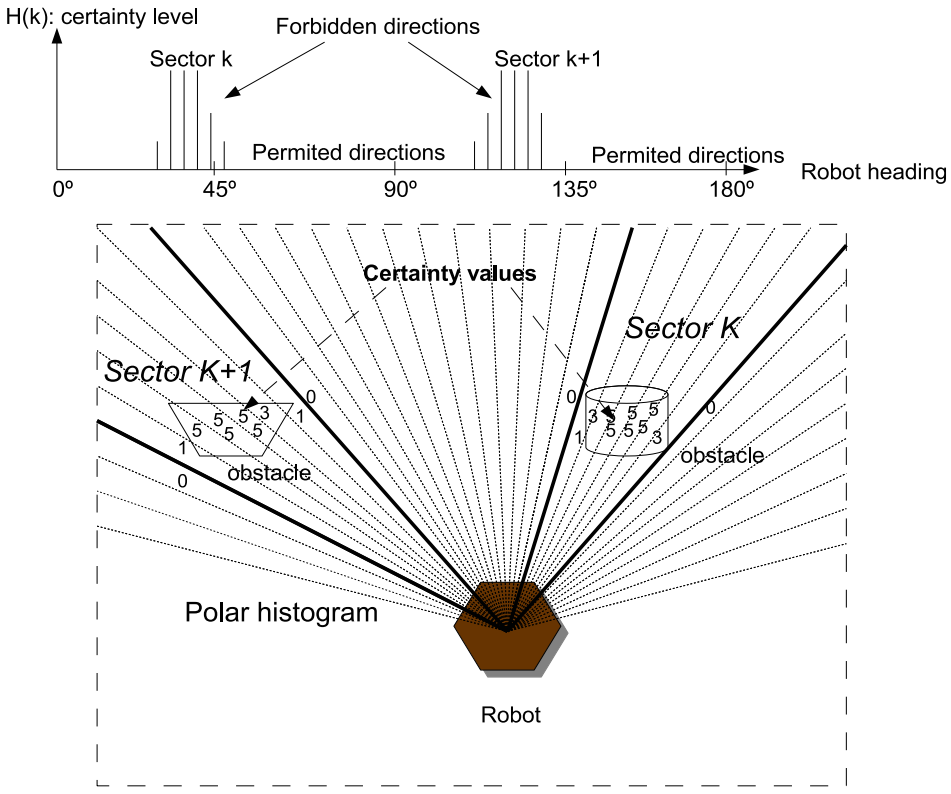


Figure 4.1: Vector Field Histogram: Sector K and Sector K+1 indicate the zones where the sensors have detected obstacles. The VFH algorithm labels each polar sector with certainty values different from 0. These values are the result of incrementing one unit each time a reading is found in that cell. Zones of the scene with high density of obstacles correspond to the peaks of the histogram.

as unsafe directions and zones of the histogram with valleys or low values of the *polar obstacle density* are labeled as free directions. Finally the control module steers the robot toward the free direction that is closest to the target direction.

A threshold defines the maximum H_k value to consider a polar direction to be a suitable candidate for the robot motion. This technique smooths the robot steering control and considers, for example, each door aperture as being a zone free of obstacles. The idea is illustrated in figure 4.1. The bottom of this figure shows how the polar sectors with obstacles have certainty values different from 0. These numbers labeling each polar direction are used to compute the H_k values which make up the histogram shown in the top of the figure. The heading directions with high density of obstacles are clearly reflected in those parts of the histogram with high density of H_k values.

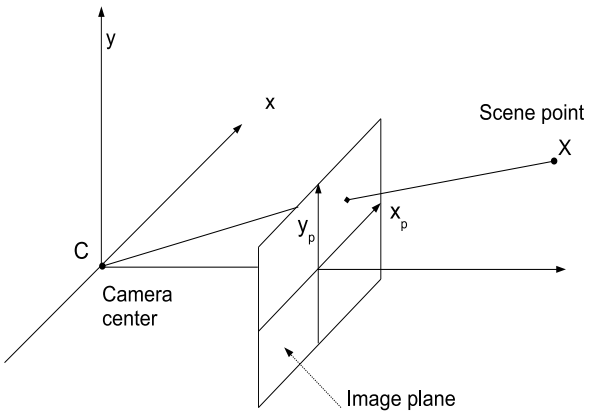


Figure 4.2: The Perspective Transformation

4.2 Perspective Transformations

This section reviews the direct and the inverse perspective transformations, since they are the basis of the obstacle detection technique used in our navigation strategy.

The Direct Perspective Transformation (DPT) maps three-dimensional points onto a plane called the plane of projection. This transformation models the process of taking a picture. The line that connects a world point with the camera lens intersects the image plane defining the corresponding and unique image point of that world point (see figure 4.2). The inverse process, that is, the projection of every image point back to the world is modeled by the Inverse Perspective Transformation (IPT). The back projected point will be somewhere in the line that connects the image point with the center of projection (camera lens).

The direct and the inverse perspective projections are usually modeled assuming a pinhole camera [52] [83]. Three coordinate systems are involved: the world, the camera and the image coordinate systems. The linear mapping between world to image points, both expressed in homogeneous coordinates, can be written as [83]:

$$\begin{bmatrix} x_p \\ y_p \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} T_w^c \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \qquad T_w^c = \begin{bmatrix} R_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}, \tag{4.3}$$

where (x_p, y_p) are the image point coordinates, f is the focal length, (x, y, z) are the corresponding scene point world coordinates and T_w^c is the 4×4 transformation matrix that relates, via a rotational (3×3) matrix R and a translational (3×1)

vector T, the world and the camera coordinate frames.

The scene point world coordinates corresponding to an image point can be calculated knowing either the distance between the camera and the point in the scene or any of the (x, y, z) world coordinates; as for example, for points lying on the floor $(z = 0)$.

The equations in closed form to perform the Direct Perspective Transformation can be exposed as follows [52]:

$$x_p = f\frac{(x - X_0)cos\theta + (y - Y_0)sin\theta}{-(x - X_0)cos\varphi sin\theta + (y - Y_0)cos\varphi cos\theta + (z - Z_0)sin\varphi} \tag{4.4}$$

$$y_p = f\frac{(x - X_0)sin\varphi sin\theta + (y - Y_0)cos\varphi cos\theta + (z - Z_0)cos\varphi}{-(x - X_0)cos\varphi sin\theta + (y - Y_0)cos\varphi cos\theta + (z - Z_0)sin\varphi} \tag{4.5}$$

while the equations in closed form to perform the Inverse Perspective Transformation [52] are:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \lambda \begin{pmatrix} x_p cos\theta - f cos\varphi sin\theta + y_p sin\varphi sin\theta \\ x_p sin\theta + f cos\varphi cos\theta - y_p sin\varphi cos\theta \\ f sin\varphi + y_p cos\varphi \end{pmatrix} \tag{4.6}$$

where:

- (X_0,Y_0,Z_0) are the lens world coordinates at the moment in which the frame was taken,
- θ is the yaw angle of the camera,
- φ is the pitch angle of the camera,
- f is the focal distance, and
- λ is a non-zero value that parametrizes the exact position of the world point on the inverse projective ray.

Since all points lying on the floor have $z = 0$, solving for λ in z and substituting back, all performed over equation 4.6, the remaining world coordinates (x^*,y^*) turn out to be:

$$x^* = X_0 - \frac{Z_0 x_p cos\theta + (y_p sin\varphi - f cos\varphi)(Z_0 sin\theta)}{y_p cos\varphi + f sin\varphi} \tag{4.7}$$

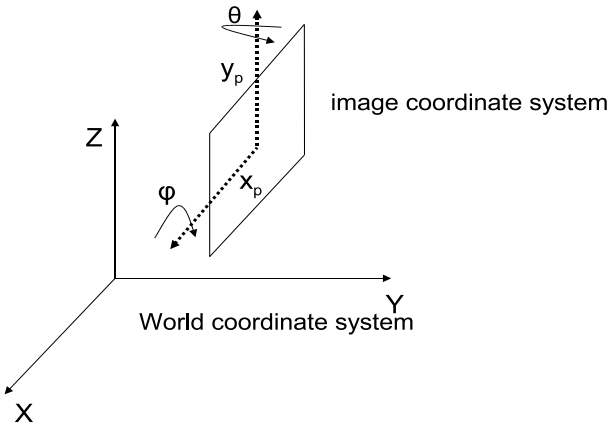


Figure 4.3: Coordinate frame conventions

$$y^* = Y_0 - \frac{Z_0 x_p sin\theta - (y_p sin\varphi - f cos\varphi)(Z_0 cos\theta)}{y_p cos\varphi + f sin\varphi}. \tag{4.8}$$

Coordinate system conventions and notations are illustrated in figure 4.3.

4.3 Obstacle Detection

4.3.1 Feature Classification

The Basic Approach

Let us consider a single camera mounted on a moving platform running on a flat ground, combining translational with rotational motion, and capturing frames at consecutive time steps. The relative motion between the camera and the scene causes the displacement of salient features in the image, changing their image coordinates (x_p, y_p), but still representing the same scene point. Given an image feature, the world coordinates of its corresponding scene point can be computed by means of equations (4.7) and (4.8) in two consecutive images assuming that the world point lies on the ground ($z = 0$).

If the assumption is correct, the two resulting pairs (x^*,y^*) will coincide. However, if the feature does not correspond to a ground point, the resulting values turn out to be different to one another and different to the real world coordinates (x, y) of the scene point. Hence, one can distinguish if an image point belongs to an obstacle or to the floor projecting it onto a previously assumed flat ground ($z = 0$) and comparing the distance between the resulting (x^*,y^*) values calculated for the

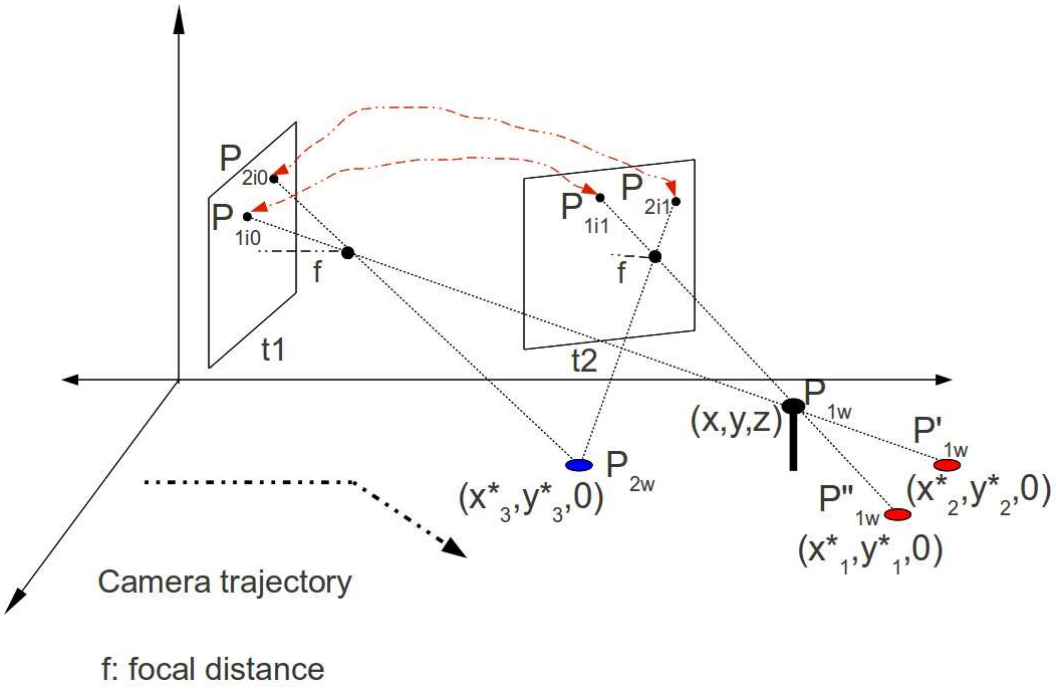


Figure 4.4: The IPT-based obstacle detection approach.

same image point tracked in two consecutive images:

$$(discrepancy) \quad D = \sqrt{(x_2^* - x_1^*)^2 + (y_2^* - y_1^*)^2} \Rightarrow \begin{cases} if D > \beta \Rightarrow \text{obstacle}, \\ if D \leq \beta \Rightarrow \text{ground} . \end{cases} \tag{4.9}$$

where (x_1^*,y_1^*) and (x_2^*,y_2^*) correspond to instants t_1 and t_2, respectively, and β is the threshold for the maximum difference admissible between (x_1^*,y_1^*) and (x_2^*,y_2^*) to classify the feature as ground point. Ideally β should be 0.

The idea is illustrated in figure 4.4. Two frames of a scene are taken at instants t_1 and t_2. Point P_{2w} is on the floor. Its projection into the image plane at instants t_1 and t_2 generates the image points P_{2i0} and P_{2i1}, respectively. The Inverse Transformation of P_{2i0} and P_{2i1} generates a unique point P_{2w}. P_{1w} is an obstacle point. Its projection into the image plane at t_1 and t_2 generates, respectively, the points P_{1i0} and P_{1i1}. However, the Inverse Transformation of P_{1i0} and P_{1i1} back to the world assuming $z = 0$ (e.g. projection onto the ground plane), generates two different points on the ground, namely, P'_{1w} and P''_{1w}.

It is important to emphasize that this algorithm does not depend on any specific

feature detector or matcher. In our particular implementation, SIFT features were used in the studies conducted to assess the classifier performance off-line and in the first on-line navigation experiments. SIFT were chosen because of their robustness to scale changes, rotation and/or translation, as well as changes in illumination and view point. However, later the KLT feature detector and sparse tracker were used in the on-line navigation experiments which required the robot to move from the departure point to some fixed goal points. KLT was chosen because: a) it is faster than other detectors, b) it generates a sufficiently large number of features with high repeatability, c) feature descriptors are not significantly affected by changes in scale since differences in consecutive frames are negligible at relative hight frame rates, d) more than 90% of the tracked features are inliers and well classified (see chapter 5), and e) they are usually found in corners, thus on potential edges, facilitating the detection of the obstacle boundaries.

Feature Detection and Tracking

The first key step of the algorithm is to detect a sufficiently large and relevant set of image features and match them across consecutive images.

Establishing good correspondences between image points, for example between P_{2i0} and P_{2i1} in figure 4.4, is of crucial importance as incorrect correspondences lead to wrong classifications. Because of this, wrong correspondences between image points in consecutive frames are filtered out using RANSAC (*Random Sample Consensus*) and imposing the epipolar constraint ($x_p^{'T} F x_p = 0$, where $x_p^{'T}$ and x_p are the point image coordinates in two consecutive frames, and F is the fundamental matrix) [83]:

The procedure can be summarized as follows:

1. Detect a set of N (N >= 8) image features and match them in two consecutive images.

2. Select M subsets formed each one by 8 randomly selected correspondences from the entire set N.

3. Compute the fundamental matrix F_j for every subset using the 8 correspondences and up to a scale factor.

4. Let us define the distance d_C corresponding to a feature matching $x_p^{'} \mapsto x_p$ as the deviation of the value $x_p^{'T} F x_p$ from 0 (the epipolar constraint).

5. For each estimate F_j compute the inliers as all those correspondences with $d_C < n$, where n is a threshold previously set. Usual values of n are between 1 and 3 pixels and the value chosen in our practical implementation was 1 pixel. The rest of matchings are considered to be outliers.

6. Choose the fundamental matrix F_j that gives the maximum number of inliers as the resulting F.

7. Refine the fundamental matrix estimation only from the final set of inliers.

The number of subsets M is usually given by:

$$M = \frac{log(1-P)}{log[1-(1-\epsilon)^q]} \tag{4.10}$$

where P is the probability that the estimate matrix is correct, q is the size of the subset, in this case 8 points, and ϵ is the assumed initial percentage of outliers.

However, in our practical implementation M was set to 500 since it experimentally demonstrated to give an optimum performance.

Direct Identification of Some Obstacle Points

The set of scene points that map to a given image point can be written as [52]:

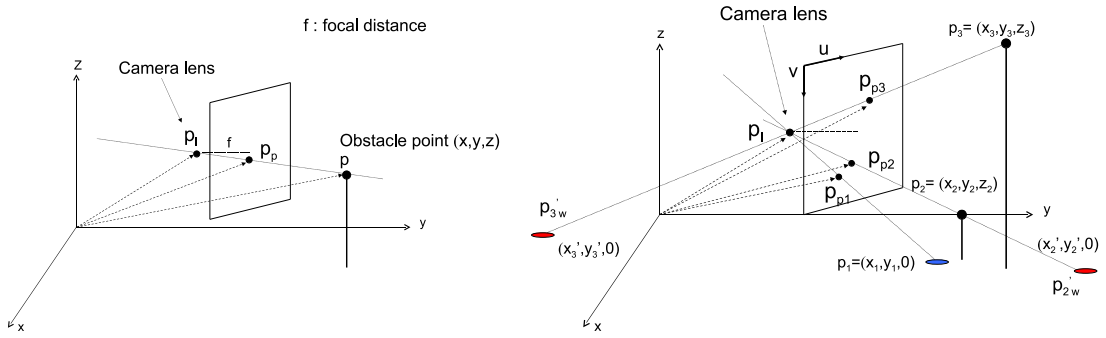
$$p = p_l + \lambda(p_p - p_l) \tag{4.11}$$

where p is a point of the scene with coordinates (x, y, z), p_l is the camera center (X_0, Y_0, Z_0) and p_p is the image point corresponding to the scene point p. This expression is equivalent to equation 4.6. The idea is illustrated in figure 4.5-(a).

All those image features (p_p) that correspond to scene points (p) located below the plane parallel to the flat ground and that contains the lens center (p_l), require a positive λ to be back-projected onto the ground, while λ is negative for all image features corresponding to scene points located above the mentioned plane.

The idea is illustrated in figure 4.5-(b). p_1 is a point lying on the floor, and its corresponding image point is p_{p1}. In equation 4.11 p_1 is obtained from p_{p1} with $\lambda > 0$. Likewise, p_2 and p'_{2w} result from p_{p2} for $\lambda > 0$. However, p_{p3} leads to a point on the ground p'_{3w} for which λ is < 0

Clearly, image points with λ negative necessarily correspond to scene points above the ground, while image points with λ positive can correspond either to



(a) (b)

Figure 4.5: (a) The IPT: $p = p_l + \lambda(p_p - p_l)$. (b) $\lambda's$ positives and negatives.

elevated points or to points lying on the floor. Consequently, the process of inverse projection and discrepancy computation can be omitted for all these image features that need a $\lambda < 0$ to be projected onto the ground, as they can be directly classified as obstacle points.

From equation 4.6, the z world coordinate of a scene point can be calculated as:

$$z = Z_0 + \lambda(f sin\varphi + y_p cos\varphi) \tag{4.12}$$

where λ determines the exact position of the scene point in the inverse perspective projecting ray.

If $z = 0$, λ can be easily solved as:

$$\lambda = \frac{-Z_0}{f sin\varphi + y_p cos\varphi}. \tag{4.13}$$

$\lambda < 0$ means that $(f sin\varphi + y_p cos\varphi) > 0$. Solving for y_p:

$$y_p > -f \tan\varphi. \tag{4.14}$$

Expressing the y_p image coordinate in pixels, and translating the origin of the image coordinate system from the image center to the upper left corner (from now on, the image coordinates with respect to the origin located at the upper left corner will be denoted as (u, v)), we obtain that:

$$v < v_0 + k_v f \tan\varphi \tag{4.15}$$

where v_0 depends on the image resolution and represents the constant factor used to convert vertical image coordinates with respect to the center of the image into

vertical coordinates with respect to the upper-left corner of the image, and the k_v factor is the vertical relation $[pixels/length]$ in the images.

All image points with a vertical coordinate v lower that $v_0 + k_v f \tan\varphi$ pixels correspond to obstacle points of the scene. Therefore, these image points can be directly classified as obstacles without being evaluated using the expression 4.9, and thus reducing the total execution time of the classifier.

Overall Performance of the Classifier: Experimental Assessment

A total of 16 different image sequences, with more than 150 frames for each sequence, were recorded by a calibrated camera mounted on a moving Pioneer 3DX robot. The camera height was 430mm, the camera pitch and yaw angles were $-9°$ and $0°$, respectively, and the focal length was 3.720mm. Test recordings comprised several types of scenarios: with regular and irregular shaped obstacles, without obstacles, with textured and untextured floor, with specularities and other with low illumination conditions. Frames were recorded during rectilinear motion in the forward direction and all were processed off-line to test the point classifier performance.

Just before running the feature tracking procedure, all images were, a) down-sampled to a resolution of 256×192 pixels, in order to reduce the feature tracking computation time, and b) undistorted to correct the error in the image feature position due to the distortion introduced by the lens, and thus, to increase the accuracy in the calculation of (x^*, y^*). For each scene, the classifier algorithm was run over undistorted pairs of 0.5-second separation consecutive frames (one frame every $2cm$) so that the effect of the IPT was noticeable. This separation between frames was empirically determined. Increasing the frame rate decreases the IPT effect over the obstacle points, and decreasing the frame rate delays the execution of the algorithm.

Although SIFT features are not necessarily located on corners or on edges, they were initially used to test the algorithm with real images and to assess it because of their robustness, repeatability and invariance in scale changes or rotations. Lately, once the important parameters that affected the classification process and the system on-line applicability (classification time, number of features, their stability, number of inliers, miss-classification rate, etc...) were determined, it was necessary to search for that feature tracker that could optimize all these variables and give

the best results to the global navigation system. The tracking algorithm was implemented according to the methods and approaches described in [112]. In frames captured in our daily scenarios, SIFT features classified as obstacle points were all near edges facilitating the obstacle detection task.

The camera world coordinates were calculated for each frame by dead reckoning, taking into account the relative camera position with respect to the robot center.

First of all, the classifier performance was formally determined using *ROC* (*Receiver Operating Characteristic*) curves [24]. A *ROC* curve can be seen as a graphical plot of the binary classifier sensitivity, evaluated under different environmental conditions. These different conditions are usually modeled by a threshold that determines the boundary used to decide whether an element belongs to a class or to another. The *ROC* curve space can be defined by the true positives rate (*recall*) in the y axis and the false positives rate (*fall-out*) in the x axis:

$$recall = \frac{TP}{TP + FN} \qquad fall - out = \frac{FP}{FP + TN}, \tag{4.16}$$

where TP is the number of true positives (obstacle points correctly classified), FN is the number of false negatives (obstacle points classified as ground), FP is the number of false positives (ground points classified as obstacle) and TN is the number of true negatives (ground points correctly classified).

Both axes range between 0 and 1. A *ROC* curve close to the straight diagonal line that connects the points $(0, 0)$ and $(1, 1)$ defines a completely aleatory classifier, while a *ROC* curve close to the $y = 1$ line defines a classifier with a sensitivity close to 100%.

In our case, these curves were computed for every pair of consecutive images. They plotted the *recall* of classified points in the y axis vs the *fall-out* in the x axis, for different values of the threshold β defined in equation (4.9). Every different value of β leads to different values of TP, TN, FP and FN. The cost function $f(\beta) = FP(\beta) + \delta FN(\beta)$ was calculated for every pair of consecutive frames, varying the β value. The optimum β was chosen so as to minimize $f(\beta)$. During the experiments, δ was set to 0.5 to prioritize the minimization of false positives over false negatives. For all tested scenes in all different scenarios, $f(\beta)$ minimized for a common β value around 21mm.

The AUC *(Area Under the ROC Curve)* is commonly used as a summary measure, providing significant insights of the system performance in a very simple way: the closer it is to 1 the better, with 1 indicating a perfect classifier and 0.5 indicating

a completely aleatory performance [81]. The AUC is equivalent to the probability that a randomly chosen member of one class (class A) have a smaller estimated probability of belonging to the other opposite class (class B) than a randomly chosen member of class B [80]. Although some authors of some other disciplines claim that the AUC permits distinguishing between bad and good classifiers but not between good classifiers [121], as a matter of fact, we have considered that classifiers with higher AUCs present better classification results [171]. The AUC was calculated for every *ROC* curve as a measure of success classification rate.

The classification of those image features that present a discrepancy D (see equation (4.9)) close to the threshold β can be altered very easily with slight oscillations in β or in D.

In order to decrease the sensitivity of the classifier with regard to β, all these points are left unclassified. Besides, in a previous training phase conducted before the autonomous navigation, histograms of D values for misclassified points were built from images taken in all different testing scenarios. D ranges with higher values in these histograms were stored in a database. During the autonomous navigation phase, all points with a D included in any stored D interval were neither classified. Then, nearly all ground points classified as obstacles were eliminated, reducing the risk of detecting false obstacles. Although some true obstacle points were also removed, the remaining ones were sufficient to permit the detection of those obstacles. The elimination of miss-classified features increases the *recall* and decreases the *fall-out*, improving the success rate of the classifier.

Figures 4.6 and 4.7 show some examples of the classifier algorithm. Pictures (a)-(b) and (g)-(h) of figure 4.6, and (a)-(b) of figure 4.7 show two consecutive frames corresponding to scenes 1, 2 and 3 respectively. Pictures (b) and (h) of figure 4.6 and picture (b) of figure 4.7 show obstacle points in red and ground points in blue. Although some ground points were wrongly classified as obstacles, the AUC of the ROC curves for scenes 1 to 3 (plots (c) and (i) of figure 4.6 and plot (c) of figure 4.7) show values of 0.98, 0.94 and 0.92, respectively, suggesting considerably high rates in the proportion of well-classified points. Plots (d) and (j) of figure 4.6 and plot (d) of figure 4.7 show the histograms of D values for true (blue) and false positives (red). D values for true positives predominantly range from 21mm to 300mm, and false positives from 21mm to 50mm. All positives with D values from 21mm to 50mm were filtered out, as shown in pictures (e) and (k) of figure 4.6 and in picture (e) of figure 4.7, where no false positives appear. The risk of detecting false obstacles

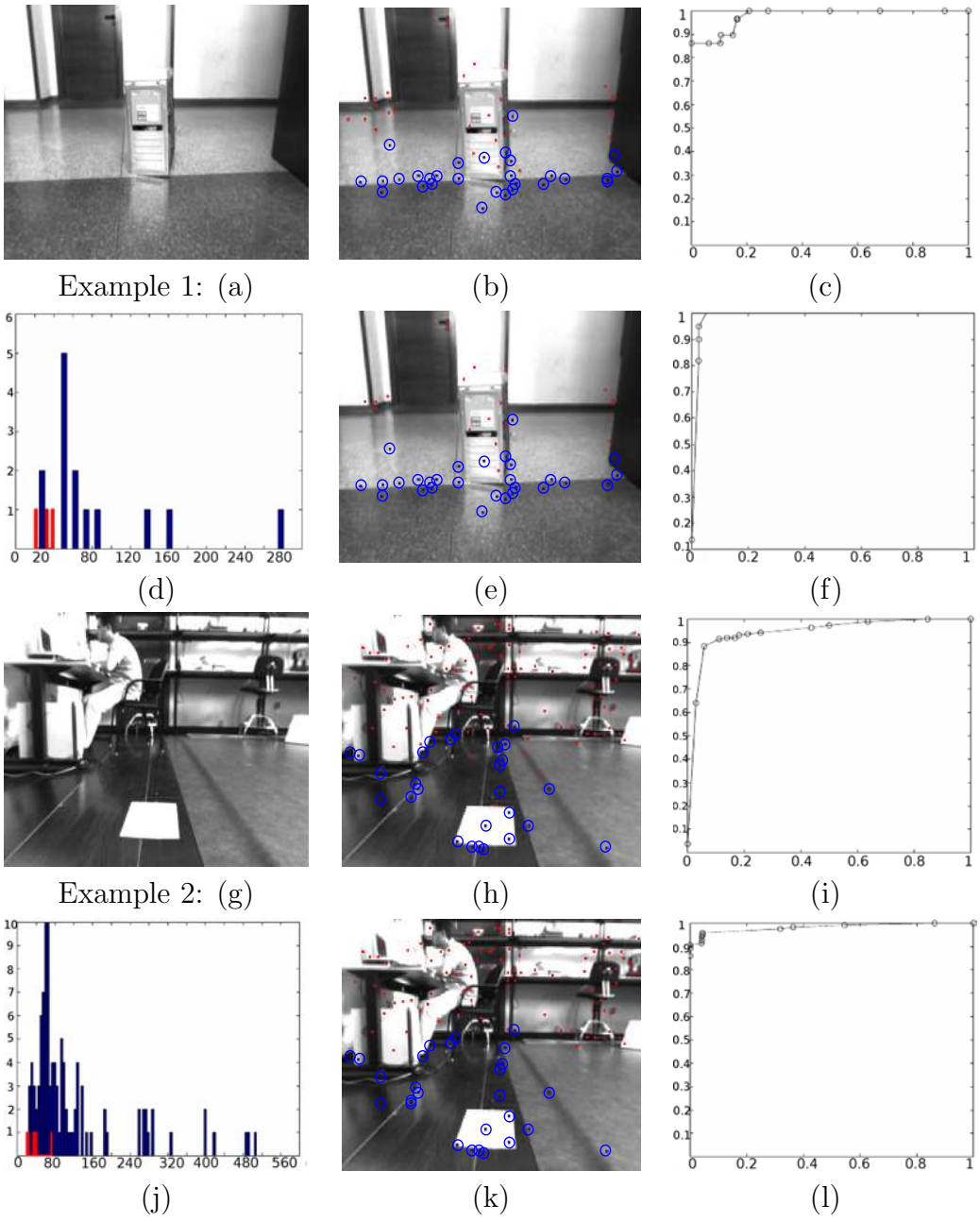


Figure 4.6: Classifier Assessment. (a), (g) Undistorted first frame of examples 1 and 2, respectively. (b), (h) Undistorted second frames. (c), (i) ROC curves of scenes 1 and 2, respectively (AUC=0'9791, AUC=0'9438). (d), (j) Histogram of D values for TP (blue) and FP (red). (e), (k) Second frame with filtered SIFT points. (f), (l) ROC curves after the filter (AUC=0'9843, AUC=0'9821).

Table 4.1: Rates of false positives and AUC for some of the analyzed scenes

Scene	FP/(Total N_{br} of features)	AUC after the filter
scene 1	0.0129	0.9482
scene 2	0.0078	0.9412
scene 3	0.0847	0.9434
scene 4	0.0000	0.9554
scene 5	0.0069	0.9834
scene 6	0.0000	0.9376
scene 7	0.0069	0.9827
scene 8	0.0166	0.9900

is reduced but a sufficient number of true positives is maintained to detect the real obstacles. Plots (f) and (l) of figure 4.6 and plot (f) of figure 4.7 show the final ROC curves and AUCs increasing to 0.98, 0.98 and 0.94, respectively. Notice that all the scenes present inter-reflections and specularities, although they do not affect the classifier performance.

Table 4.1 presents some relevant data concerning various tested scenes. 21 frames were analyzed for every scene. The total amount of features ranged between 105 and 260. The percentage of false positives with respect to the total number of features suggests failure ratios lower than 8%. Finally, the AUC values after the filtering process have values greater than 0.93 in all cases, suggesting a high degree of performance. Although the classifier performs accurately, obstacle points near the floor, which may have low D values, have more probability of being miss-classified than others with a larger height.

4.3.2 Obstacle Profiles

Image features or corners are usually detected at regions of high gradient, thus they are likely to be near or belong to an edge. Besides, features classified as obstacle are most likely to be contained or near a vertical edge belonging to an obstacle. Hence, the next step of the algorithm is the computation of edge maps and the association of such edges with features classified as obstacle. This permits isolating the obstacle boundaries from the rest of edges and getting a qualitative perception of the environment.

In order to combine a high degree of performance in the edge map computation with a relatively low processing time, the edge detection procedure implements a

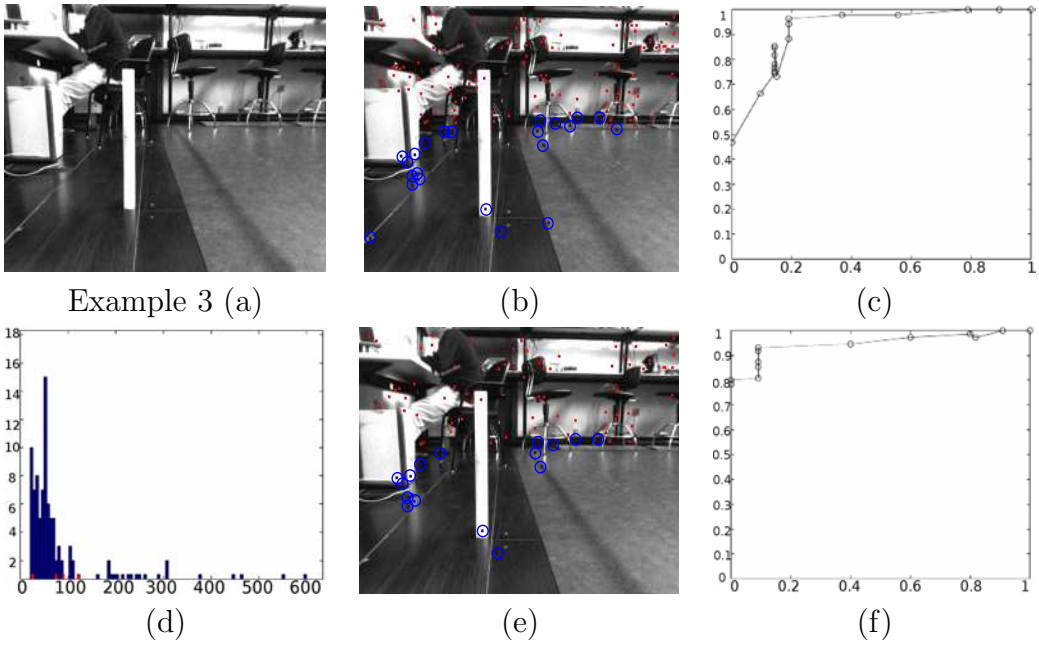


Figure 4.7: Classifier Assessment. (a) Undistorted first frame. (b) Undistorted second frame. (c) ROC curve (AUC=0'9236). (d) Histogram of D values for TP (blue) and FP (red). (e) Second frame with filtered SIFT points. (f) ROC curve after the filter (AUC=0'9494).

reduced version of the Canny edge detector [28] running only these two basic steps:

1. First, the original image is convolved with a 1D Gaussian derivative horizontal kernel. This permits detecting, with a single convolution, zones with high vertical gradient from areas with smooth intensity values.

2. Next, a process of hysteresis thresholding is applied. Two thresholds are defined. A pixel with a gradient above the highest threshold is classified as edge pixel. A pixel with a gradient above the lowest threshold is classified as edge if it has in its vicinity a pixel with a gray value higher than the highest threshold. In this way, edge pixels with low gradient are not filtered if the threshold is defined too high, and noise is not considered as an edge if the threshold is defined too low.

The algorithm locates in the image all features classified as obstacle. Then, for each one, it founds all edge pixels which are inside a window centered on the feature image coordinates. Every edge is tracked down starting from the obstacle point position until the last edge pixel or a ground point is found. This will be

considered as to be the point or points where the object rests on the floor. This process permits isolating the relevant obstacle boundaries from the rest of edges and getting a qualitative perception of the environment. In our implementation, the window used to find edges near obstacle points is longer in the vertical direction to overcome possible discontinuities in the obstacle vertical border.

4.4 Obstacle Avoidance and the Navigation Task

4.4.1 Building a Local Occupancy Map for Reactive Navigation

Knowing the camera position (X_0, Y_0, Z_0), and the world coordinates of a point on the floor $(x, y, 0)$, the distance *dcp* between the camera and the floor point can be calculates as:

$$dcp = \sqrt{(x - X_0)^2 + (y - Y_0)^2} \tag{4.17}$$

and the angle ϕ defined by the direction of motion and the relative orientation of this floor point with respect to the camera optical axis can be defined as:

$$\phi = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right) \tag{4.18}$$

where $\vec{a}$ is a vector with the same direction as the vector from the world coordinate system origin to the point $(X_0, Y_0, 0)$, $\vec{b}$ is the vector from point $(X_0, Y_0, 0)$ to the point $(x, y, 0)$, and $\vec{a} \cdot \vec{b}$ is the dot product between both vectors. The idea is illustrated in figure 4.8.

As it has been previously stated, it is possible to calculate the world coordinates of an image point if it corresponds to a ground point of the scene. Those points where the obstacle touches the ground are ground points, hence it is easy to compute their world coordinates from the image point coordinates.

The orientation and distance of obstacles with respect to the robot can be qualitatively estimated taking into account the relative position of the camera with respect to the robot center and computing the distance and orientation with respect to the camera of those obstacle points that are in contact with the floor (using, for example, equations 4.17 and 4.18).

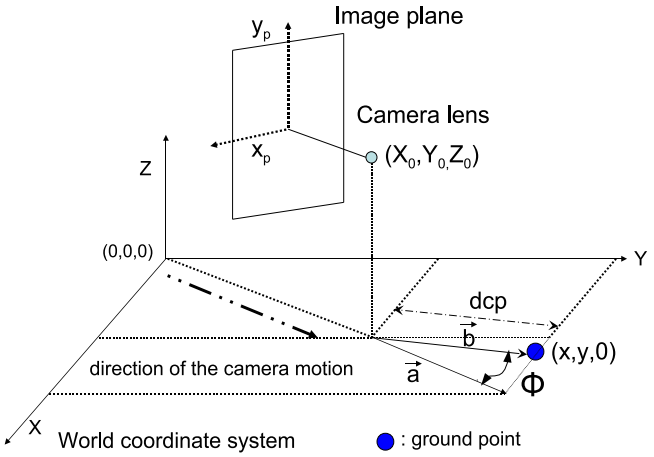


Figure 4.8: Distance and orientation of a ground point with respect to the camera

A semicircular area of a fixed radius, centered at the robot position and virtually located on the ground plane, is considered to be the ROI (*Region of Interest*) used for the obstacle avoidance. Only obstacles detected inside this ROI are considered to be avoided.

The ROI is in turn divided in angular regions. Histograms of obstacle-to-ground contact points at each polar direction of the ROI are computed. Those polar directions corresponding to angular regions occupied by at least one obstacle-to-ground contact point are labeled, in principle, as forbidden and those free of obstacle-to-ground contact points are included in the set of next possible movement directions. This process results in a polar occupancy map of free and occupied zones, that qualitatively represents the vicinity of the robot. Obstacle-free polar regions which are narrower than a certain threshold (determined empirically and depending on the robot size) are excluded from the possible motion directions. If all free angular regions are narrower than the defined threshold, the algorithm concludes that all the space ahead is occupied by obstacles and returns the proper order to stop the robot.

This method has been inspired on the VFH approach but here adapted to vision-based systems.

4.5 Experimental Results: Autonomous Navigation in an exploration task

After the classifier demonstrated an appropriate performance, it was convenient to test it online, integrated in the global obstacle detection and avoidance process. In a first set of experiments the mission to be accomplished by the vehicle was merely moving erratically through the free space, detecting the obstacles present in the scenarios described in section 4.3.1. Hence, there was no target point. For all these experiments, the navigation module implements and applies the VFH method, but here adapted to visual systems. The objective of these experiments was simply evaluating: a) how the algorithm discriminated the obstacle boundaries from the ground, b) the accuracy in the world coordinates of the obstacle-to-ground contact points, and, c) if the occupancy grids, as defined, were appropriate for the obstacle avoidance task.

The direction of motion was given as a vector pointing to the center of the widest polar obstacle-free zone. Positive angles resulted in turns to the right and negative angles in turns to the left. The system used the same operating conditions described in section 4.3.1, in terms of: camera, robot speed, focal distance, lens height, frame rate, image resolution and intrinsec camera parameters for image undistortion. The camera world coordinates were obtained composing the robot world coordinates obtained from the dead reckoning data with the camera position with respect to the robot.

Figure 4.9 shows four examples of the obstacle contour discrimination algorithm applied over images of a sequence recorded during these navigation experiments. Pictures (a), (c), (e) and (g) are the second frame of four different pairs. Pictures (b), (d), (f) and (h) show the corresponding edge maps with the obstacle boundaries highlighted in orange. Although picture (e) shows a very high inter-reflection on the ground and a very granulated texture on the floor tiles, only real obstacle boundaries survived.

Figure 4.10 shows in plots (a), (b), (c) and (d) the trajectories followed by the robot, according to the odometry data, during the navigation through the scenarios of figures 4.11, 4.12, 4.13 and 4.14. All space coordinates shown in these plots are expressed in milimeters. The blue circle denotes the starting point and the red circle denotes the end point. Figures 4.11, 4.12, 4.13 and 4.14 show data corresponding to some navigation experiments. Pictures (a), (b), (c) and (d) in all four

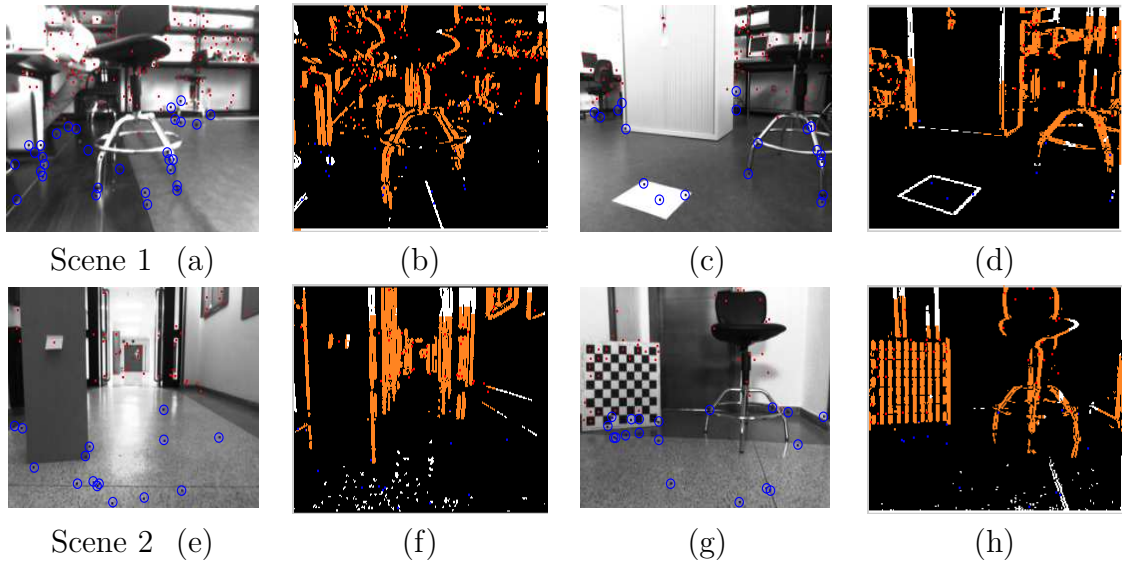
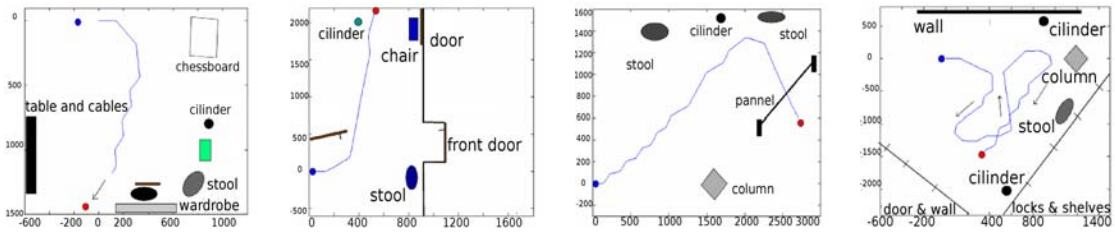


Figure 4.9: Examples of obstacle contour discrimination. (a), (c), (e), (g): Undistorted second frame of different pairs of consecutive images of Scenes 1, and 2. (b), (d), (f), (h): Obstacle contours.

figures show the second frame of some pairs recorded and processed in scenarios 1, 2 and 3. Every image was taken before the robot had to turn to avoid the frontal obstacles; obstacle points are shown in red and ground points in blue. Figure 4.11 (scenario 1) shows a room full of obstacles with regular and irregular shapes. This scene presents shadows and inter-reflections. Figure 4.12 (scenario 2) corresponds to a corridor with a very high textured floor, columns, walls, inter-reflections and some specularities. Figures 4.13 and 4.14 (scenario 3) present bad illumination conditions, important inter-reflections, specularities on the floor, and some image regions (white walls, shelves and lockers) with homogeneous intensities and/or textures. Regions with homogeneous textures result in few distinctive features and poorly edged obstacles, which can difficult their detection. Pictures (e), (f), (g) and (h) in all four figures show the vertical contours (in orange) comprising obstacle points. As shown, obstacle contours were differentiated from the rest of the edges. Obstacle-to-ground contact points inside the ROI have been highlighted in pink.

Histograms (i), (j), (k) and (l) in figures 4.11, 4.12, 4.13 and 4.14 account for the number of obstacle-to-ground contact points detected in each polar direction. Plots (m), (n), (o) and (p) show the semicircular floor portion in front of the robot with all the obstacle-to-ground contact points, representing the local and qualitative occupancy maps used for reactive navigation.



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

Figure 4.10: (a), (b), (c) and (d), robot trajectories for tests of figures 4.11, 4.12, 4.13 and 4.14, respectively. Length information referred in mm.

In all these tests the steering direction is shown in green. The experiments performed show a remarkable robustness against textured floors, bad illumination conditions, shadows or inter-reflections, and deals with scenes that include different planes. In all scenes, features were well classified with success rates greater than 90%, obstacle profiles were correctly detected and the robot navigated through the free space avoiding all obstacles.

4.5.1 Conclusions

Constructing local maps is a suitable way to represent the vicinity of the robot for reactive navigation. Many of the reactive visual-based navigation solutions that build or use local occupancy maps are sensitive to floor and obstacle textures, homogeneity in the color intensity distribution, to the illumination conditions or to the relative position of the camera with respect to the ground. Determining or identifying exact obstacle shapes, exact positions, dimensions, colors or textures is not essential to qualitatively decide the next direction of motion.

In this chapter, a new qualitative obstacle detection and avoidance algorithm has been presented.

The complete strategy starts with a novel image feature classifier that distinguishes between obstacle features from features lying on the ground. The classifier was experimentally assessed using ROC curves and their AUC. The ROC curves were plotted from data fetched from sequences recorded by a camera mounted on a moving robot platform. AUCs were higher that 0.9 in all curves, suggesting considerably high success rates in the classification results. The detection of points that belonged to obstacles permitted: a) discriminating the obstacle boundaries from the rest of edges, and b) detecting those points where obstacles contacted the ground,

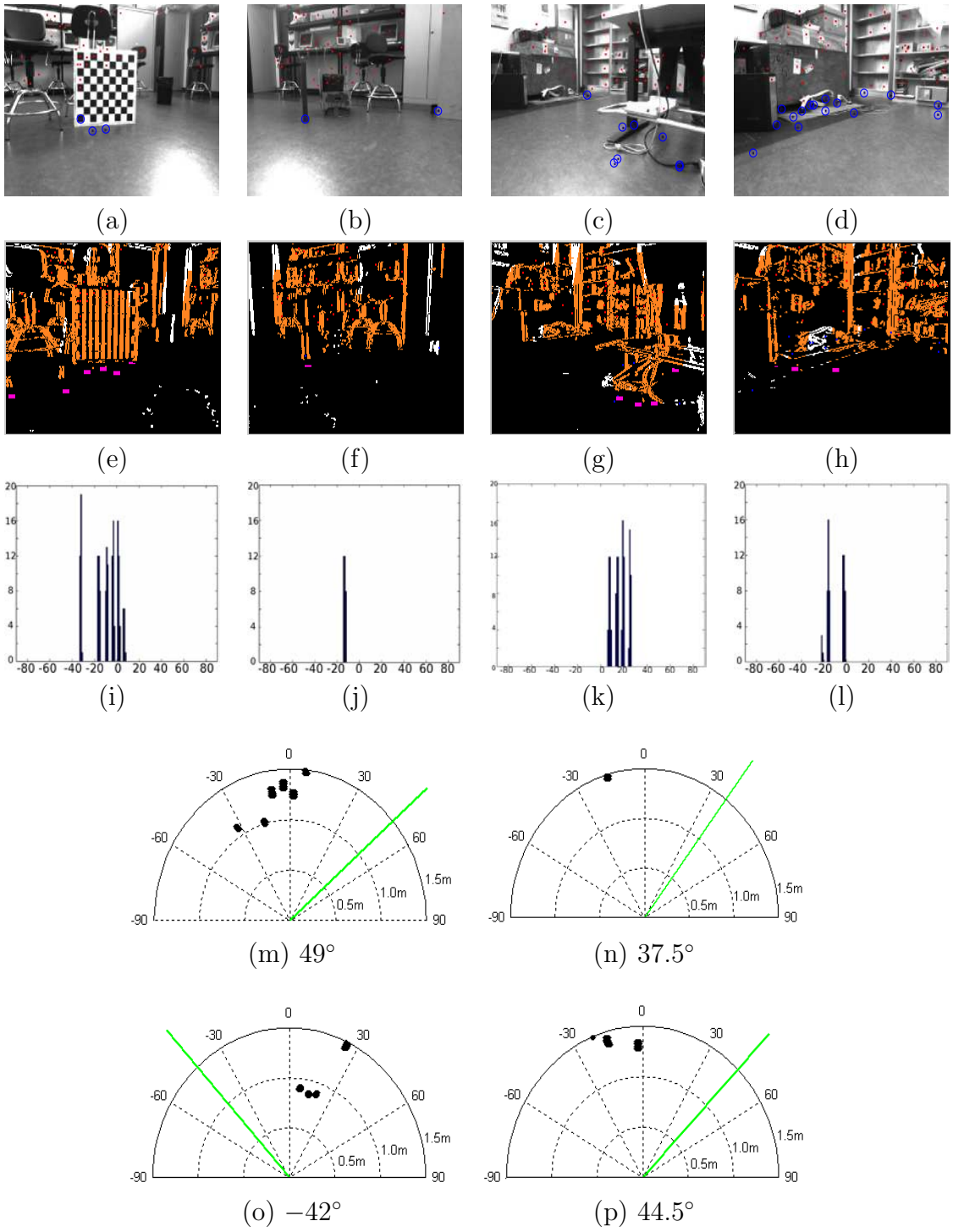


Figure 4.11: Scenario 1. Experiment 1: (a), (b), (c) and (d), undistorted frames; (e), (f), (g) and (h), corresponding edge maps with obstacle borders highlighted in orange. (i), (j), (k), (l), histograms of obstacle-to-ground contact points for each polar direction between −90° and 90°. (m), (n), (o) and (p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and (d) respectively.

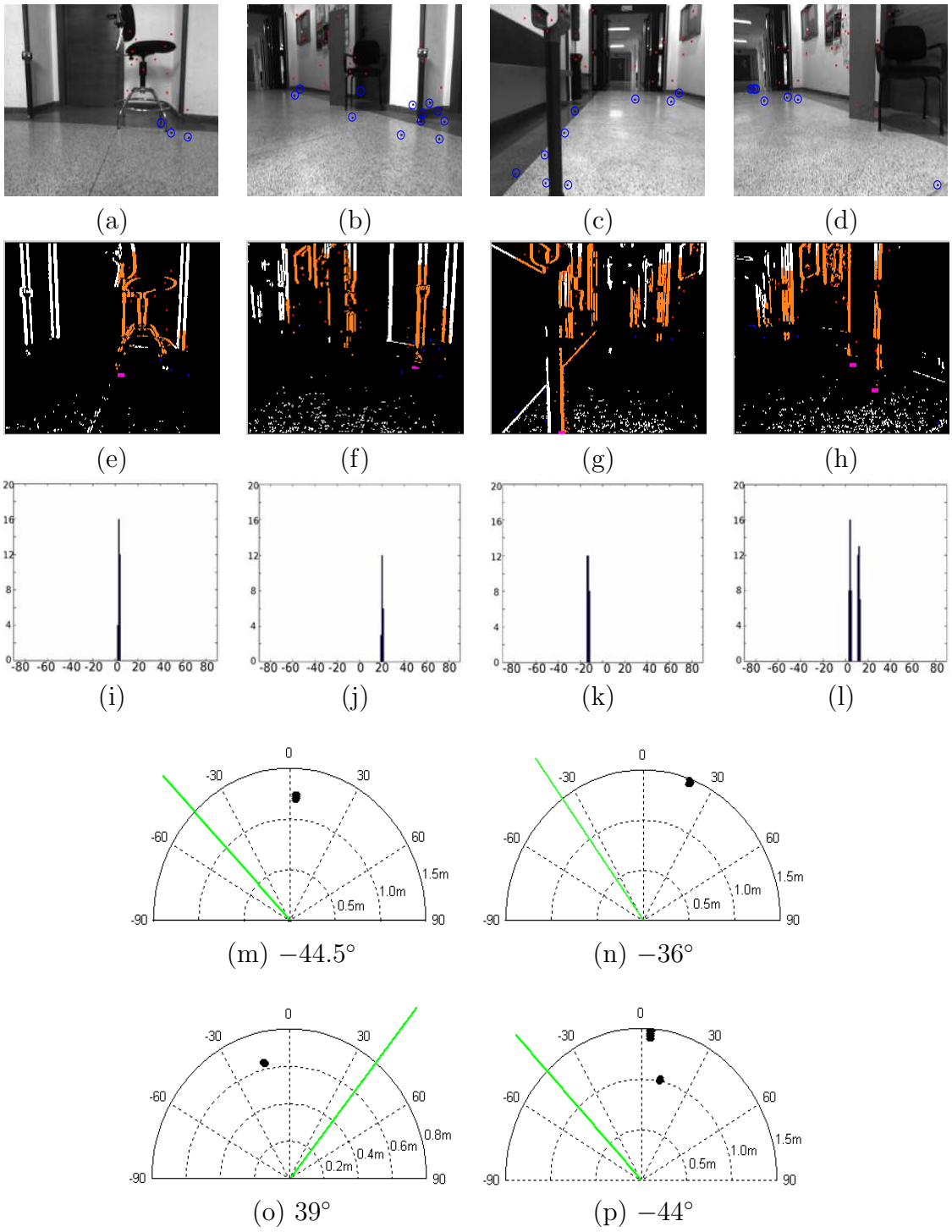


Figure 4.12: Scenario 2. Experiment 2: (a), (b), (c), (d), undistorted frames; (e), (f), (g) and (h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k), (l), histograms of obstacle-to-ground contact points for each polar direction between −90° and 90°; (m), (n), (o) and (p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and (d), respectively.

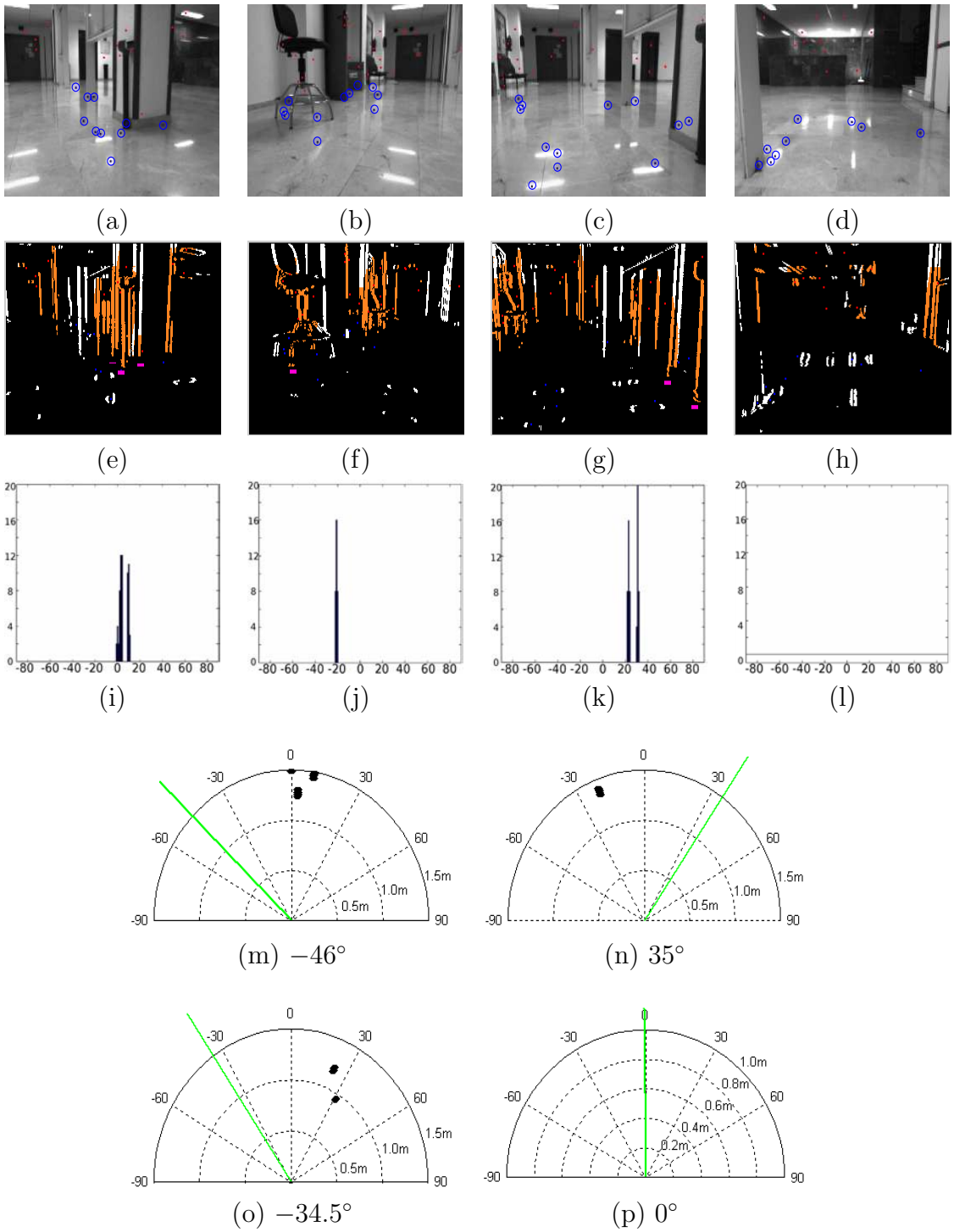


Figure 4.13: Scenario 3. Experiment 3: (a), (b), (c) and (d), undistorted frames; (e), (f), (g) and (h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k) and (l) histograms of obstacle-to-ground contact points for each polar direction between −90° and 90°; (m), (n), (o) and (p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and (d) respectively.

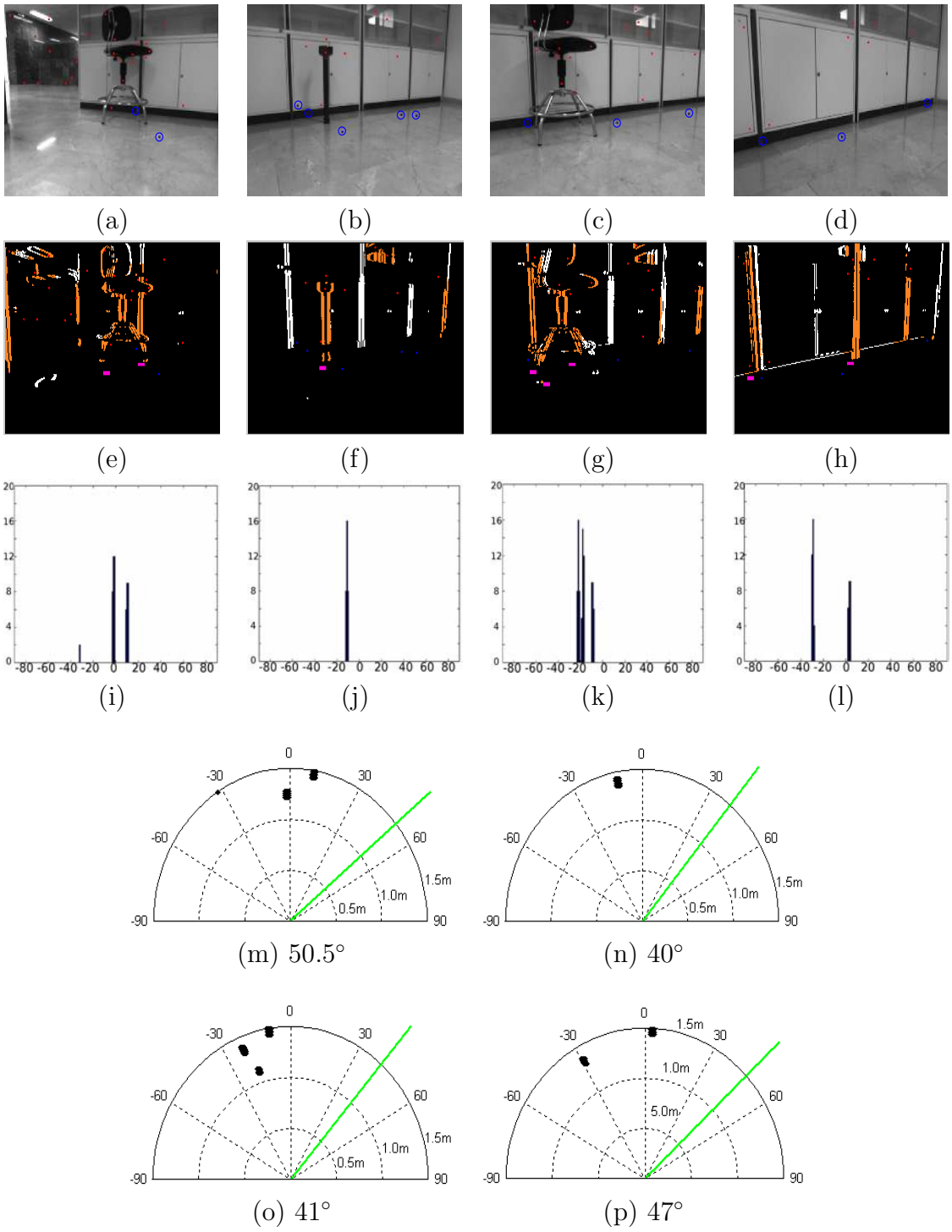


Figure 4.14: Scenario 3. Experiment 4: (a), (b), (c), (d), undistorted frames; (e), (f), (g) and (h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k), (l), histograms of obstacle-to-ground contact points for each polar direction between $-90°$ and $90°$. (m), (n), (o) and (p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and (d) respectively.

and calculating their world coordinates.

The system built a radial qualitative model of the robot vicinity where only the world coordinates of the obstacle-to-ground contact points detected in the image were included. The goal was not to build maps as accurately as possible, but to infer a qualitative idea of which parts of the environment were occupied and which ones were free. These maps evidenced the presence of *something* that had to be avoided, in a determined direction and at a specific distance. Zones free of obstacle points were the candidates for the next direction of motion.

The algorithm shows a certain robustness to the presence of shadows, inter-reflections, specularities or textured floors, overcomes scenes with multiple planes and uses only a certain number of image points reducing the execution time respect to those that process the whole image.

The obstacle detection algorithm was tested in an autonomous robot with the unique mission of roaming inside the environment avoiding all obstacles. The direction of motion was always pointing to the center of zones free of obstacle points. Steering vectors were computed just to escape from areas potentially occupied. These preliminar on-line tests were necessary to evidence the suitability of the algorithm for real on-line navigation applications and to see which environmental variables could affect the system. The experimental setup consisted of distinct scenarios with different characteristics, several kinds of obstacles, different illumination conditions and various floor textures. In all cases the mobile robot was able to navigate through the free space avoiding all obstacles, walls and columns.

One of the major problems encountered in the navigation tests was the execution time. SIFT is slow compared to other commonly used feature detectors, and it delays the whole process of obstacle detection and avoidance. If the feature tracker is slow, then the calculation of the steering vector can be delayed too much. This delay can lead the robot to crash unless it stops until the moving decision has been taken. There are also other parameters that affected the obstacle detection algorithm: for example, the image coordinates where the features were detected, how many features were detected and how many features could be correctly tracked. Due to the nature of our obstacle detector, features located over corners and edges would be preferred than others. In order to detect all obstacle boundaries, the algorithm needs features in all obstacle edges and the more features it gets, the best it performs. And finally, under inconvenient illumination or environmental conditions, the number of outliers can increase. It is important to have a high

degree of inliers in the matching process since outliers are rejected and, thus, some parts of the environment could be undetected.

For all these reasons, testing different feature detectors and trackers in our classifier algorithm was necessary in order to find out which was the one that gave the best results and performance for our particular navigation system. This specific point is studied in the next chapter.

CHAPTER 5

ASSESSMENT OF DIFFERENT FEATURE TRACKERS

5.1 Introduction

The initial version of the navigation approach involved the use of SIFT features because of their properties described in section 3.3.2. As it has been exposed in the previous section, early experimental results of the complete navigation strategy demonstrated appropriate performance as for the feature classification process and as for the automatic robot operation. However, the complete strategy is not restricted to a certain feature detector. Obviously, changing the feature detector, descriptor and tracking algorithm leads to different number of features, different feature locations and different execution times, and consequently, to different tracking results. All these parameters can influence the obstacle detection and thus the navigation process. Therefore, it makes sense to select the best configuration of feature detector/descriptor and tracking approach to get the best performance from the navigation task.

5.2 Evaluating Several Feature Trackers

A set of image sequences recorded with the same calibrated camera mounted on the same Pioneer 3DX were processed off-line to test different feature detectors

and matching approaches. Frames were recorded during the motion in the same scenarios used for the experimental evaluation presented in the previous chapter. However, all sequences utilized in the current assessment were exclusively recorded for this purpose. The camera world coordinates were calculated for each frame also by dead reckoning and taking into account the relative camera position with respect to the robot center. The illumination conditions did not change abruptly during the robot navigation. As it was done in the previous experiments, images were down-sampled to a resolution of 256×192 pixels in order to reduce the computation time. All frames were also rectified to correct the distortion introduced by the lens and thus to reduce errors in the feature image coordinates. For each scene, the classifier algorithm was run over more than 40 undistorted pairs of frames. The time separation between consecutive frames was reduced (with respect to the experiments presented in previous sections) to 0.4 seconds. Three additional feature detectors were tested to compare them with the early results obtained with SIFT: SURF-64 (i.e. descriptor length of 64), FAST, and KLT as a classic representative of the autocorrelation or SSD (*Sum of Squared Differences*) based methods ([10]).

FAST features were labeled with the 16 position vector descriptor defined in [163] and matched minimizing the SSD of the feature descriptor in consecutive images. SURF detection and matching was implemented following [13]. Finally, the KLT feature detection algorithm was combined with the feature tracking Lucas and Kanade's iterative method in image pyramids [23].

In all cases, the threshold β was obtained so as to minimize the cost function $f(\beta) = FP(\beta) + \lambda FN(\beta)$ for a representative set of images (FP are *false positives* and FN are the *false negatives*). During the experiments, λ was set again to 0.5 to prioritize the minimization of false positives over false negatives.

Figures 5.1, 5.2, 5.3, 5.4 show some important results obtained from the experiments carried out to test the aforementioned detectors and descriptors. Plots (a), (b) and (c) of each one of these four figures refer to, respectively, scenes 1, 2 and 3, shown in figure 5.5. Plots (d) of all four figures show, as additional examples, the results of the tests of the different feature trackers for 18 different image pairs corresponding to other 18 different video sequences recorded in different scenarios.

Consecutive frames were paired and used off-line as input to our classification algorithm to evaluate:

1. time of execution of the feature matching process,

2. the number of features,

3. the number of inliers, and

4. the percentage of miss-classified features with respect to the number of inliers.

In these experiments, the number of KLT features was set to 250. This was experimentally demonstrated to be an appropriate number for detecting all obstacle boundaries, and not too large to considerably increase the time of execution. Because of this, the KLT tracker does not appear in figure 5.2.

Plots (a), (b) and (c) of figure 5.1 show the execution time (in miliseconds) using different feature detectors and tracking approaches. Every plot shows data recovered from the execution of the classifier over 10 consecutive image pairs. Time magnitudes include the time for detecting and tracking features in a pair of consecutive images, plus the time for filtering outliers. As shown in the three plots, SIFT is the approach that consumes the greatest amount of time while KLT and FAST are the approaches that run faster. For our real time application, it can be concluded that FAST and KLT seem to be the most appropriate to ensure an acceptable frame processing rate of the obstacle detection algorithm.

Plots (a), (b) and (c) of figure 5.2 show the number of features detected using SIFT, FAST and SURF in each one of these 10 image pairs from the 3 different scenes. On the one hand, SIFT and FAST report more than 100 features, a suitable number for obstacle detection purposes. A larger number of detected features increase the probabilities of detecting all the relevant obstacle edges. On the other hand, SURF returns insufficient features to detect the whole obstacle boundaries: if some obstacle edges are not found, an obstacle or part of an obstacle will not be shown in the resulting qualitative occupancy map.

Plots (a), (b) and (c) of figure 5.3 show the percentage of inliers from the total amount of features matched in two consecutive images. These inliers are the features that will be finally classified as obstacle or ground points and that will be used to discriminate the obstacle boundaries from the rest of edges. If the number of discarded features increases, some obstacle edges can be missed. SIFT and FAST maintain a medium number of inliers while SURF and KLT show a high percentage of inliers, in some cases equal to the 100% of the matched features.

The results of the different evaluated parameters maintain a similar trend during the consecutive frames over an image sequence, suggesting stability and repeatability in the feature matching and obstacle detection process. Concerning plots (d)

of figures 5.1, 5.2, 5.3, again, all these data suggest that: i) SIFT is the most time consuming, ii) although SURF presents a great percentage of inliers, it generates few features, and iii) SIFT and KLT generate more well-classified points than FAST or SURF, which makes these options more suitable to be used within the navigation algorithm. Furthermore, in some of these scenes, not only the percentage of inliers in FAST is too low, but the percentage of miss-classified features is too high to properly detect all the obstacle edges. This makes this method too irregular to be properly used for the on-line application.

In conclusion, SIFT is the method that shows the best performance in different scenarios, it has an average proportion of inliers but, conversely, it is the most time consuming. SURF outperforms SIFT in time of execution (6 times on average); further, although the percentage of inliers is high, the number of detected points is considerably low so as to get an acceptable performance from the obstacle detection step. FAST is quick but, in some cases, the lack of regularity makes it unreliable. Sometimes, using FAST, the number of inliers can be considerably scarce with respect to the initial number of detected features, depending on the image conditions. The KLT approach is also fast, it is stable in a lot of different scenes with different lighting conditions, it detects an appropriate number of features and the number of inliers is close to 100%. Although the method is not invariant to scale changes, ensuring small magnitude displacements, e.g keeping a sufficient frame rate, reduces the relevancy of this shortcoming.

Concerning the point classifier and obstacle detection algorithm performance, plots of figure 5.4 show the number of miss-classified points with respect to the total amount of inliers, using the 4 different feature detectors. Again, plots (a), (b) and (c) refer to the aforementioned scenes 1, 2 and 3, and plot (d) covers the data corresponding to the other 18 different scenes. The features wrongly classified induce errors in the detection of obstacle edges and thus in the local qualitative occupancy map construction. The classifier algorithm using SIFT or KLT generates a lower percentage of miss-classified points than using SURF or FAST. As expected, in most cases it is lower than 10%. FAST shows a greater percentage of miss-classified features, which makes this detector non-appropriate to our purposes. These differences on the percentage of miss-classified features is due to the discrepancies in the matched feature position calculated by every different approach. Different feature image locations lead to differences in the world point coordinates.

With SURF or FAST, the number of images where some obstacle edges are not

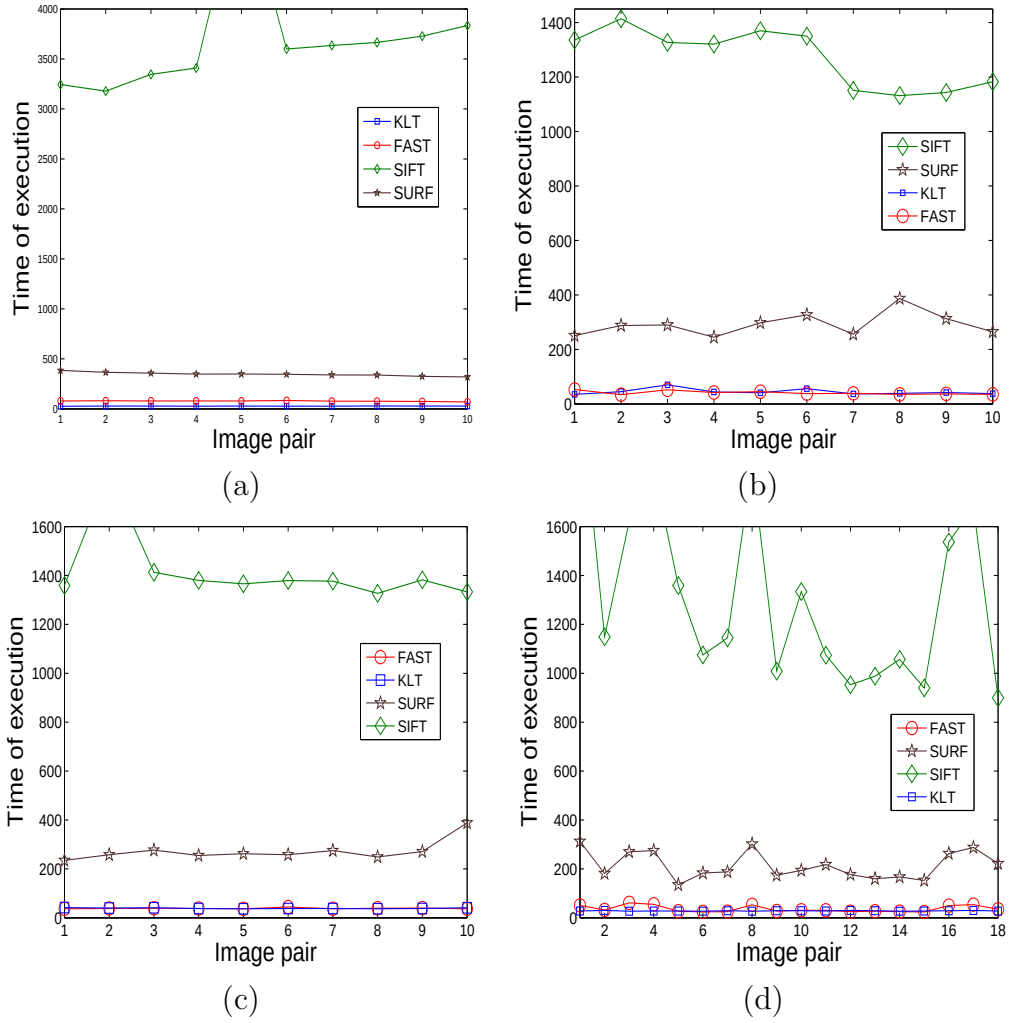


Figure 5.1: Assessment of feature trackers - execution time: (a) scene 1; (b) scene 2; (c) scene 3; (d) mixed scenes;

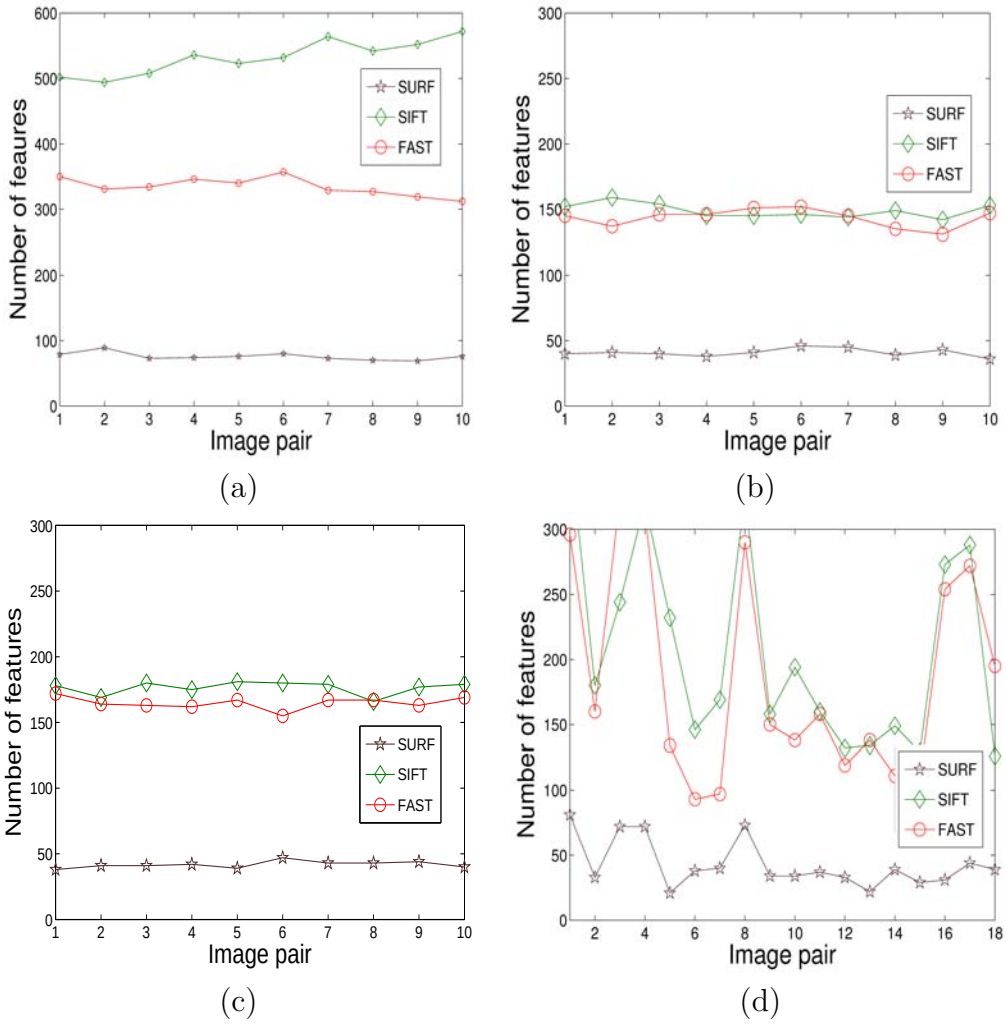


Figure 5.2: Assessment of feature trackers - number of detected features: (a) scene 1; (b) scene 2; (c) scene 3; (d) mixed scenes;

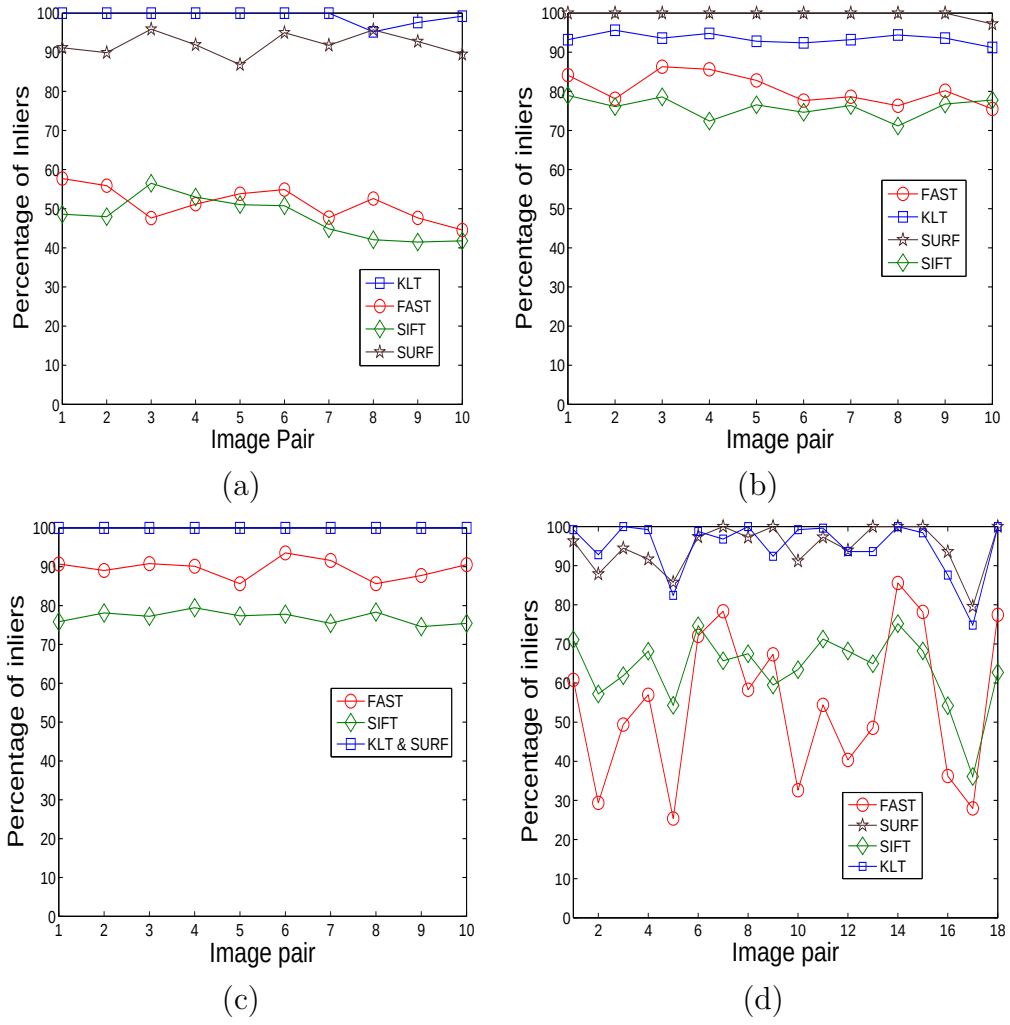


Figure 5.3: Assessment of feature trackers - percentage of inliers: (a) scene 1; (b) scene 2; (c) scene 3; (d) mixed scenes;

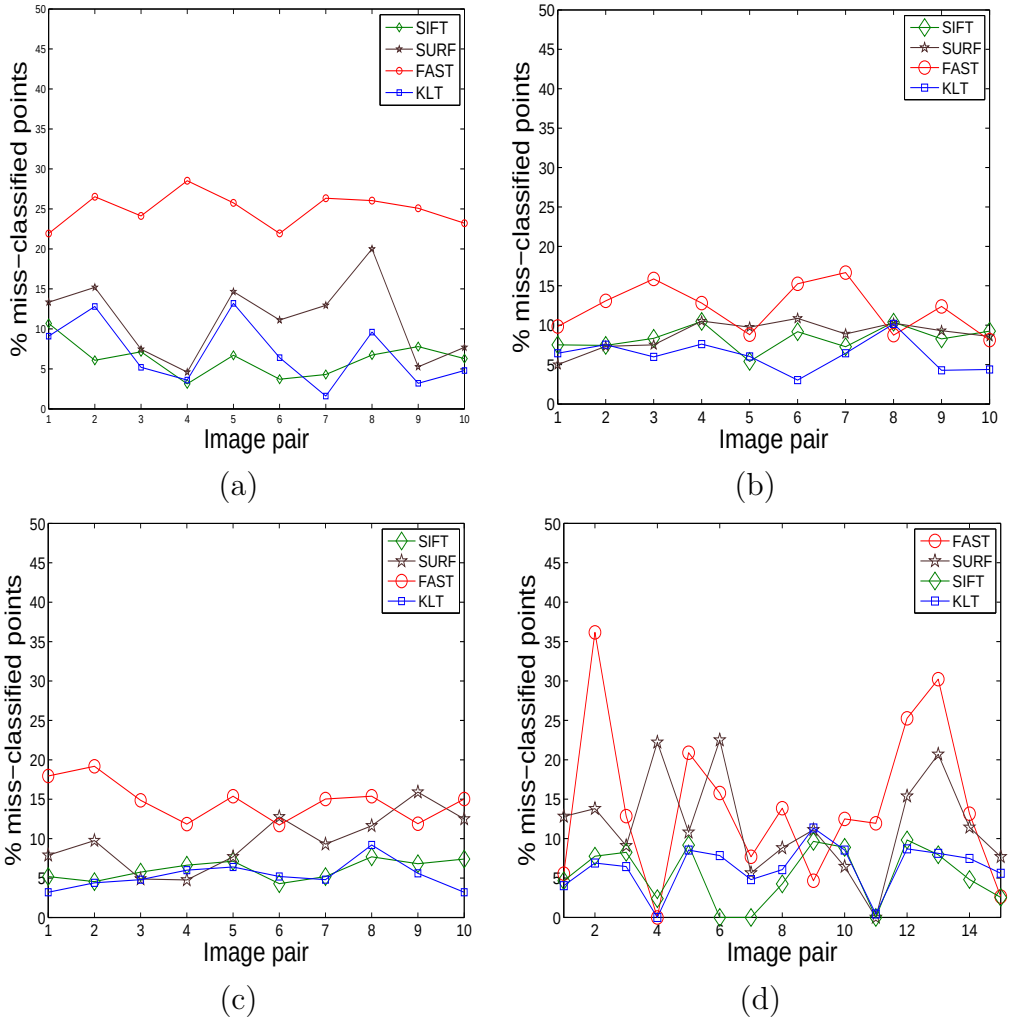


Figure 5.4: Assessment of feature trackers - percentage of miss-classified points: (a) scene 1; (b) scene 2; (c) scene 3; (d) mixed scenes;

detected increases due to the lack of features they suffer from. In consequence, from the point of view of performance, SIFT and KLT are the methods that best contribute to our classifier algorithm, but again, SIFT is too time consuming.

Figure 5.5 shows different results of the obstacle boundary detection process using the different feature detector approaches. Pictures (a) to (d), (i) to (l) and (q) to (t) show the second frame of 3 different pairs of consecutive images taken from the sequences evaluated earlier. Features classified as obstacle are shown in red and those classified as ground are shown in blue. Pictures (e) to (h), (m) to (p) and (u) to (x) of figure 5.5 show the corresponding edge maps with the obstacle contours highlighted in orange. Furthermore, obstacle-to-ground contact points that are inside a ROI with a radius of 1'5m are indicated in pink.

As can be observed, SURF either finds few features on the floor or it does not find any. This complicates the discrimination between obstacle edges and edges caused by, for example, ground textures, lines, inter-reflections or specularities. It is obvious that different approaches generate different amounts of features with different locations and this circumstance is specially relevant in the process described in this thesis. In all cases where SIFT or KLT approaches are used, all obstacle boundaries are well distinguished because both detectors find a considerable number of features and it is more likely that all obstacle edges include one. However, as it can be seen, for example, in figures 5.5-(n),(o),(v),(w), SURF or FAST generate few inliers. Consequently, all obstacle edges which have no features on them can not be well discriminated. Both, SIFT or KLT feature detectors ensure a good performance of the obstacle boundary discrimination algorithm, but KLT is much faster that SIFT.

5.2.1 Conclusions

Different feature detectors and matching techniques have been tested with the image feature classifier and obstacle detector to compare reliability, execution time, accuracy and performance. Tests conducted with SIFT, SURF-64, FAST and KLT approaches over a number of different image pairs reveal that KLT is the approach that provides the best performance to our purposes, since it is the fastest, it generates a sufficiently high number of features, near 100% of the detected features are inliers and more than 90% of the detected features are well classified. Improving all these parameters increases the obstacle boundary detection accuracy and the

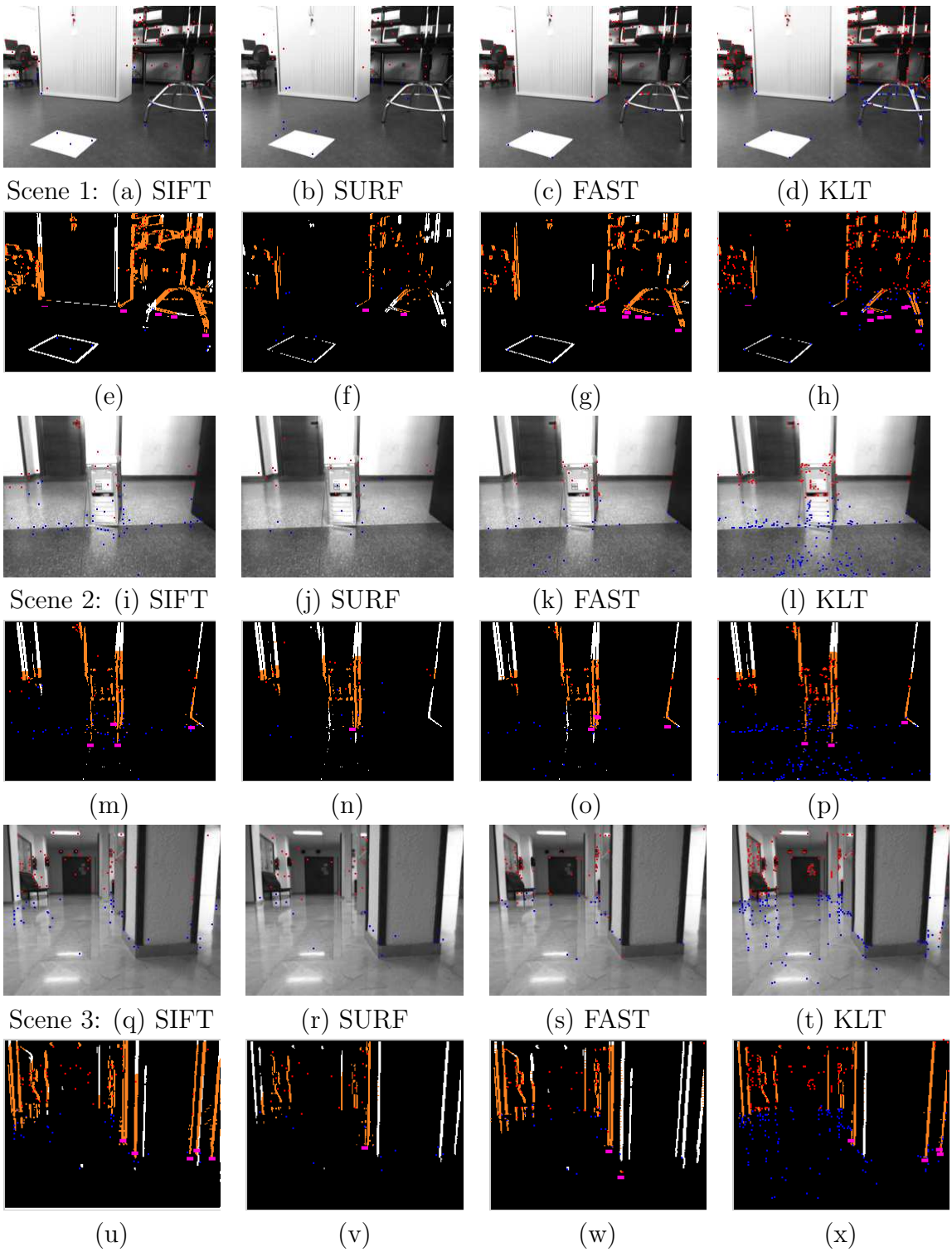


Figure 5.5: The feature classifier using different feature trackers. (a), (b), (c), (d): frame captured in scene 1. (i), (j), (k), (l): frame captured in scene 2, (q), (r), (s), (t): frame captured in scene 3. (e), (f), (g), (h), (m),(n), (o), (p), (u), (v), (w), (x): Obstacle contours and obstacle-to-ground contact points.

quality of the obstacle-to-ground contact point detection. As a consequence, the corresponding qualitative occupancy grid that has to be build for the safe navigation attains a larger level of reliability.

CHAPTER 6

ROBOT NAVIGATION STRATEGIES

6.1 Introduction

As it has been defined in section 2.1 the navigation problem deals with the ability of the mobile robots to move between several target points, avoiding all collisions and trying to maximize efficiency in terms of time and covered path length.

Current applications involving mobile autonomous agents entrust the motion and behavior control to navigation strategies. Optimum and robust skills in mobility and navigation, either for reactive, deliberative or hybrid systems have a great influence in the system global performance. The navigation abilities are crucial in the accomplishment of routine tasks such as exploration, transportation, guiding or moving in daily dynamic or scenarios densely occupied by obstacles. For example, an automatic wheel chair moving inside a university or administrative building or a transformation robot working in a warehouse.

Particularly, reactive autonomous agents do not assume any knowledge of the environment before they begin the navigation task, so it is crucial for them to be endowed with obstacle detection and navigation algorithms, robust enough to assure the achievement of all the assigned tasks without collisions as well as an accurate localization approach in case different goal points have been previously programmed. It is not enough being able to detect and avoid obstacles and perfectly localize the robot while it moves, but also to maximize efficiency, basically in terms of time and covered path length. For just erratic collision avoidance, the system only needs a

suitable obstacle detection algorithm and a minimum criteria to direct the robot to zones of free space. However, in reactive missions programmed to cover routes from a starting point to a set of subsequents goal points, maximizing efficiency implies taking into consideration that obstacles have to be avoided but the robot must deviate as less as possible from the goal direction.

The goal of the mission described in this chapter is not only to detect and avoid the obstacles, but to move from a starting to one or several goal points. Consequently, from now on, the idea of steering the robot towards the center of the widest obstacle-free zone adopted in the experiments of section 4.5 might be no longer valid, if this is not the option that pursues guiding the robot to the objective, trying to maximize the efficiency in the entire followed route.

The literature is profuse in reactive navigation strategies, (see chapter 3). Among them, Bug-T^2 *(Traversability and Tenacity)* [5] and ND *(Nearness Diagram)* [136] are the techniques that provide the most important insights for our particular system. At the time of detecting obstacles and constructing the occupancy maps, our system used all algorithms described in section 4.4. But the strategy for guiding the robot from a starting to a goal point was newly designed and inspired on the most relevant aspects of Bug-T^2 and ND. Although these two algorithms were implemented, experimentally tested and proven to be effective using range sensors, an important effort has been done to adapt their main benefits to vision-based navigation systems. It is obvious that the special characteristics of the sensors equipping a robot are critical issues on final robot navigation capabilities. For this reason, some modifications with respect the original solutions were needed to reach similar performance when using cameras with regard to using range sensors.

6.2 Comparing Bug-T^2 and ND: advantages and inconveniences

6.2.1 Some considerations about Bug-T^2

A local trapping zone is one of the most limiting situations to get success in a reactive navigation task using the Potential Fields method. Usually, the robot moves in the direction of the goal point until it finds obstacles. The influence of the attractive vector caused by the goal is reduced and/or canceled by the repulsive

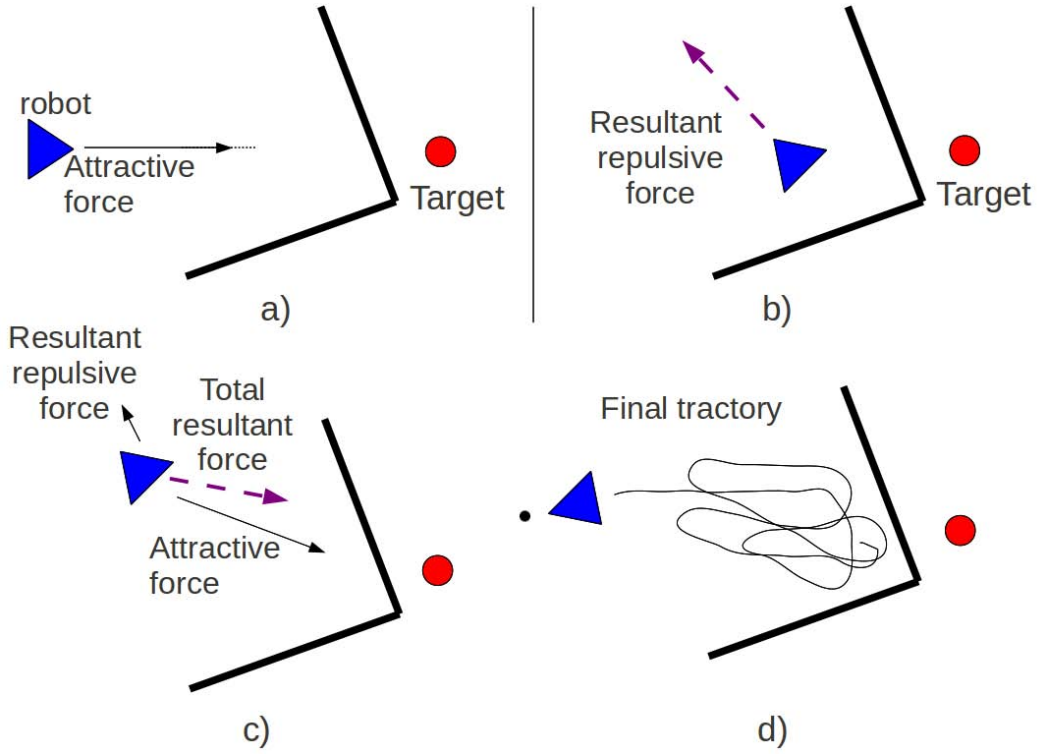


Figure 6.1: (a): The robot moves towards the target under the influence of its attractive force. (b) When it is close to the obstacle concavity, the resultant of the repulsive forces has a significant magnitude and moves the robot away from the target direction. (c): The robot is again far away from the obstacle, the magnitude of the repulsive force attenuates, reducing the influence of the obstacle on the robot. The robot feels attracted again by the goal point and it moves once again to the obstacle concavity. d) The trajectory is cyclic, approximating to and separating from the obstacle.

force generated by the obstacles. If one of the obstacles is a wall with an L or U shape and it is located in between the robot and the goal point, the robot gets the L or U concavity and then, rejected by its repulsive force, it moves backwards. Once the robot has reached a considerable distance from the obstacle, the repulsive force attenuates while the goal attractive force actuates again. The effect is that the robot goes towards the target entering again in the obstacle influence and starting a cyclic and infinite process of attraction and rejection (see figure 6.1). One of the ways to escape from this situation is to plan a strategy to move the robot away from the target direction, and that is exactly what successfully proposes Bug-T^2.

Antich *et al* [5] proposed a new reactive navigation method based on two main principles: *traversability* and *tenacity*. The *traversability* principle is supported on

the so called *navigation filter*. This *navigation filter* is a virtual circular structure with a pre-defined radius that represents the environment surrounding the robot up to a certain distance. The circle is divided in equal sectors and is filled with the sparse data supplied by the range sensors (either sonar or infrared). The sensor readings indicate the presence of obstacles. Each sector is labeled as *permitted* or *forbidden*, depending on the presence or not of nearby obstacles. Figure 6.2 shows the circular space of directions surrounding the robot. Those regions where sensor readings are detected (in brown) are labeled as occupied areas (in black). No considerations concerning the distances between obstacles and the robot are done. Simply, if there are obstacles inside one section of the navigation filter, then this section is banned. Therefore, the navigation filter shows which regions of the environment surrounding the robot can be *traversed* and which are occupied by obstacles.

If the way to the target is clear of obstacles, the robot heads directly towards the goal by means of its attractive force. When an obstacle is detected, the robot follows its contour either to the left or to the right until the obstacle has been circumnavigated. This behavior is the result of combining the repulsive force caused by the obstacle and the principles of *traversability* and *tenacity*. When surrounding the obstacle, the algorithm searches for the first traversable region, either clockwise or counterclockwise (left or right), according to the *tenacity* principle. Roughly speaking, the *tenacity* principle consists in making the same decision (left or right) as the decision made in the previous execution of the navigation algorithm. In this way, the robot keeps following the obstacle contour in the same direction until the goal direction is free again. At that moment, the filter is reset and the robot faces again towards the goal. However, the first time a certain obstacle is found there is no previous decision to apply the *tenacity* principle. In this case, three different criteria can be applied: minimum turn, fixed-beforehand and randomly [5]. Recent readings are temporarily stored in the navigation structure until the current obstacle has been completely overcome.

A further criterion that is additionally applied is as follows: if the goal point appears to be in the same sector of an obstacle, and that sector was previously banned, the local environmental information is updated and the robot is redirected towards the target.

Both principles are general concepts that can complement other classical navigation strategies such as the Potential Fields method, the Dynamic Window Approach

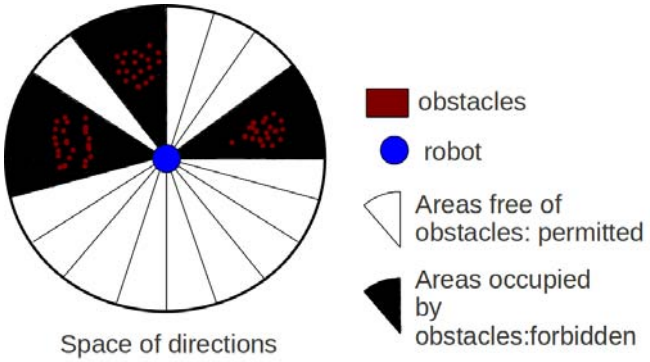


Figure 6.2: The navigation filter.

or other classical path-planning schemes such as Bug2.

To guarantee convergence, Bug-T^2 establishes some additional rules that tune the application of the *tenacity* principle and the criteria to leave or to continue the following of the obstacle contour. More specifically, in cases where, for example, the goal point is placed inside or behind a G-shaped obstacle, if *tenacity* is always applied a cyclical situation can emerge, forcing the vehicle to play always the same trajectory without reaching the goal point. To guarantee convergence even in these uncommon circumstances, Bug-T^2 must adopt the next additional criteria: a) leaving the obstacle contour either if the obstacle direction is free and it is the first time the robot passes by this current position, or, b) if the robot trajectory cuts the *Main-Line* [1] in a point closer to the goal than previous intersections between the Main-Line and the robot trajectory (see [5] for further details).

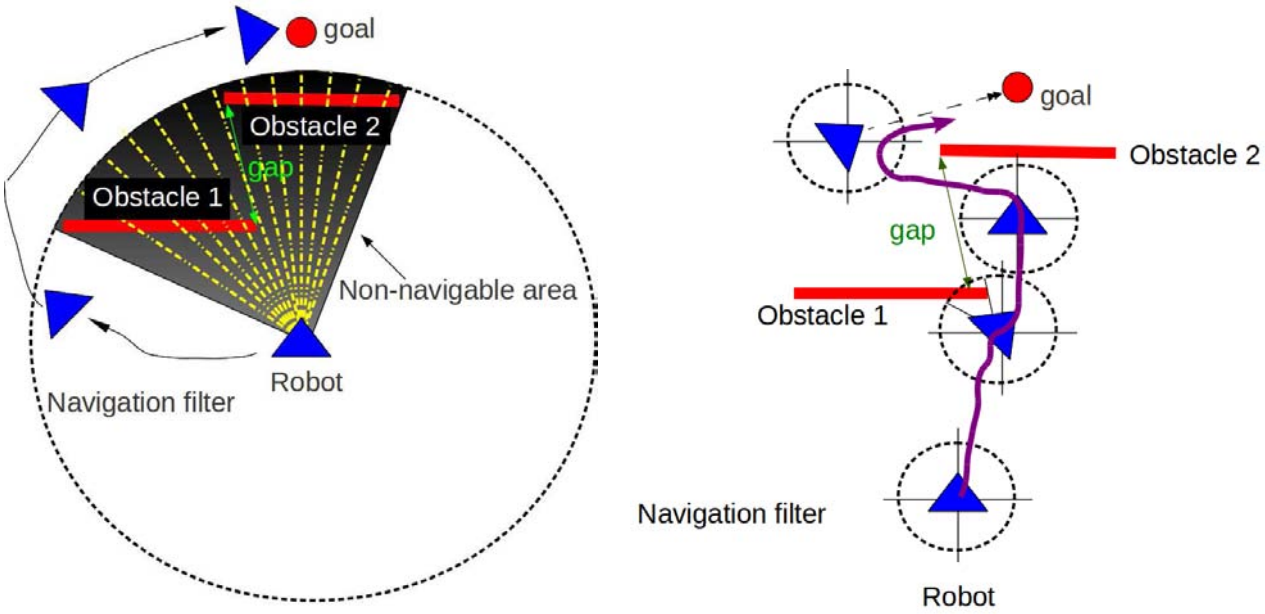
Bug-T^2 is a particularly robust strategy because: (a) with a relatively sparse set of low-cost sensor readings, it builds an occupancy map sufficiently adequate to define the free and the occupied zones in the vicinity of the robot, (b) it uses all benefits from the *Bug*2 strategy and from the Potential Fields method, all combined with the original principles of *traversability* and *tenacity* to guarantee convergence to the goal point, c) it guarantees the robot escaping from trapping zones, regardless their shape and dimensions.

One of the important issues of the T^2 algorithms is to determine the radius of the navigation filter. Extending the area of this region of interest would permit the system to perceive wider parts of the environment; however, in this case, the algorithm can fail when it tries to detect free apertures between obstacles, if they

[1] Main Line: the straight segment joining the starting and the target points

do not occupy, at least, one entire sector of the filter (see figure 6.3). From now on, discontinuities or apertures between obstacles will be referred to as gaps. As it can be observed in figure 6.3-(a), the robot can perfectly pass through the indicated aperture between both obstacles. But, as the sectors of the filter which contain the gap are all forbidden since they also contain part of the obstacles, the robot has to turn around to continue its route. Contrarily, figure 6.3-(b) shows the same environmental configuration but with a smaller navigation filter. As the radius of the filter is shorter, the robot does not notice the presence of the obstacles until they are close. Furthermore, the robot is only able to see one of the two obstacles inside the navigation filter. Figure 6.3-(b) shows in magenta the most likely route followed by the robot. The robot starts going towards the goal point since there are no obstacles inside the region of interest. When it detects the *obstacle 1* it begins to round it in the orientation of minimum turn. Immediately, the goal direction is free since nothing is found inside the filter. The filter is reset and the goal direction is taken again. When the *obstacle 2* is detected, the robot starts its surrounding until the path to the target is free and the robot can proceed towards it. The robot has passed through the gap, in a path towards the goal much more efficient than the one shown in picture (a).

The length of the navigation filter diameter can be adapted to the characteristics of the environment and to the robot structure and dimensions. For navigation purposes, it would be necessary to detect only those gaps that the robot can pass through. Navigation filters with shorter radius permit the robot navigating closer to the obstacles and thus detecting more quantity of traversable gaps, however as the navigation filter radius decreases and the robot navigates closer to the obstacles, two implications progressively arise: 1) the collision risk increases, and 2) the detection of obstacles present in the environment is delayed, preventing the system to anticipate the upcoming situations. Figure 6.4 illustrates the idea. The blue robot has a navigation filter with a radius much shorter than the green robot. Due to the shortness of its filter radius, the blue robot is unable to detect the obstacle and the gap until it is very close to them. This permits the robot always finding the gap but, on the other side, traveling so close to the obstacles increases the collision risk and it prevents the robot from anticipating that the gap is not traversable. The green robot will minimize the risk of collision and it will anticipate the obstacle, but it will be able to detect the gap only if it is long enough to occupy, at least, one entire portion of the navigation filter. In summary, the length of the filter radius must



(a) (b)

Figure 6.3: Relevance of the filter radius in the navigation performance. (a) Gaps are not detected because the radius of the navigation filter is too long; (b) Gaps are detected because the radius of the filter is smaller than in case (a).

be always adjusted to find a compromise between following the obstacle contours, minimizing the risk of collision and being able to find as many traversable gaps as possible. Practical implementations of the algorithm used navigation filters with a radius between 1 and 2 meters, giving optimal results in the tested environments.

Besides, the algorithm is able to define a sort of occupancy map of the surrounding scenario only with the disperse readings provided by the range sensors. By decreasing the range of the navigation filter, the amount of obstacles influencing the robot is reduced and they are easier to control. Under these circumstances, the robot is unlikely to anticipate potential undesired situations but it is prepared to escape from them.

6.2.2 Some considerations about ND

The ND [136], ND$^+$ [137] and other strategies derived from them (for example [54]) tackle the problem of autonomous navigation in troublesome environments, which include either complex and cluttered scenarios. To this end, the method applies the paradigm of *perception-action* which consists on defining a set of cases and designing

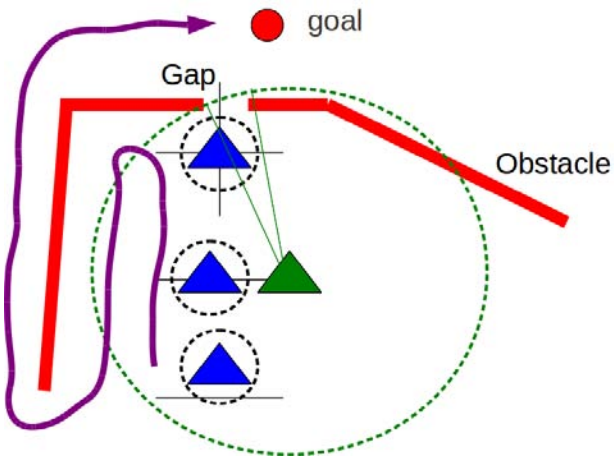


Figure 6.4: Relevance of the filter radius in the navigation performance. The green robot, with a larger navigation filter, will detect the obstacle earlier than the blue robot. If the gap occupied more than one sector of the navigation filter, the green robot could detect it and anticipate its movements.

an action to be executed in each case. Authors claim that the method anticipates and avoids trapping situations. However, ND is not able to rule escaping from local minima, once the robot is deeply inside one of them. Using this technique, the motion is free of oscillations in narrow apertures. ND and ND$^+$ assume a holonomic robot and the sensor readings (obstacle information) represented as points.

The strategy searches for discontinuities between obstacles (gaps) wider than the robot diameter. The search of gaps is done analyzing diagrams that represent the nearness to the robot of obstacles located between it and the maximum sensor range. Then, regions are built between two contiguous discontinuities. The algorithm selects the navigable region closest to the goal location. This region is called the *free walking area*. This formulation avoids declaring a U-shaped obstacle as a navigable region. A security zone is defined as a region surrounding the robot, with a pre-set radius and where the risk of collision is considered very high. The objective is to find a set of situations, being each one capable of uniquely characterizing a defined configuration of the group formed by obstacles, robot and goal location. The different situations are organized in a binary decision-tree:

- If there are obstacles inside the security zone the situation is defined as LS (*Low Safety*), and if the security zone is free of obstacles the situation is defined as HS (*High safety*).

- If the system is in the HS case, two different sub-situations can arise:

    1. if the goal is inside the *free walking area*, then the situation is labeled as HSGR (*High Safety Goal in Region*), or
    2. if the goal is outside of the *free walking area*, the possible situations will be, either HSWR (*High Safety Wide Region*) or HSNR (*High Safety Narrow Region*).

- If the system is in the LS case: if there are obstacles in the security zone and the goal location is within the *free walking area*, the situation is labeled as LSGR (*Low Safety Goal in Region*). If not, situations can be either LS1 (*Low Safety 1 side*) where there are obstacles within the security zone, but only on one side of the discontinuity of the *free walking area*, or LS2 (*Low Safety 2 sides*), when there are obstacles on both sides of the discontinuity.

The actions for every situation can be summarized as follows:

- HSGR: the direction of motion, θ_{mot}, is the goal direction, θ_{goal}.
- HSWR: $\theta_{mot} = \theta_{wa} + \alpha$, where θ_{wa} is the direction of the discontinuity of the walking area and α is an additional angle to prevent the obstacle invading the security zone.
- HSNR: $\theta_{mot} = 0.5(\theta_{wa1} + \theta_{wa2})$, where θ_{wa1} and θ_{wa2} are the directions of both sides of the walking area.
- LSGR: $\theta_{mot} = \theta_{goal} + \beta$, where β is a additional angle which depends on the distance to the obstacle inside the security region.
- LS1: $\theta_{mot} = \theta_{wa} + \alpha + \beta$, where $\alpha + \beta$ depend on the direction and distance of the discontinuity and the distance of the closest obstacle.
- LS2: analogously, $\theta_{mot} = 0.5(\theta_{wa1} + \theta_{wa2}) + 0.5(\gamma_1 + \gamma_2)$, where γ_1 and γ_2 depend on both discontinuities and the distance to the closest obstacle.

All experimental implementations presented in [136], [137] and [54] used laser range finders as the main sensor to perceive the environment, obtaining optimal results.

6.2.3 Discussion

Navigating in unknown, dynamic or cluttered environments is one of the most challenging tasks in reactive systems. On the one hand, Bug-T^2 is designed to represent the navigability of the area surrounding the robot from a set of disperse sensor readings. Sectors of the local map with obstacle points are banned and regions without obstacle points are permitted. The robot follows the obstacle boundaries until the path to the goal in free again. The dimensions of the local maps, so called *navigation filters*, must be conveniently determined to circumnavigate all obstacles at a cautious distance to avoid collisions but trying to detect as many gaps as possible. The strategy prioritizes arriving to the goals than rising maximum efficiency in terms of time and path length. Besides, the application of the *Traversability and Tenacity* principles guarantees escaping from trapping zones. However, the strategy only considers the presence of obstacles, but not their distance to the robot or the existence of discontinuities among them.

On the other hand, ND and other solutions derived from it are techniques designed to navigate in cluttered environments, finding all gaps or apertures between obstacles detected from the robot pose to distances equal to the maximum sensor range. ND-based techniques take into consideration, not only the presence of obstacles, but also the distances between them and the robot, trying to search each time for the closest path to the goal. Though, they do not guarantee neither convergence towards the target point nor escaping from deep global trapping situations once the robot is immerse into one of them.

Considering all these arguments, it appears reasonable to search for a solution that tries to take advantage from some of the benefits of both techniques and refuse their drawbacks. Besides, cameras can drastically reduce the cost of the robot equipment with respect to laser-based approaches, but still offering data with considerable spatial and temporal resolutions.

From Bug-T^2 the new visual navigation strategy adopted the principles of *Traversability* and *Tenacity* and the concept of the navigation filter as a way to characterize the local environment with a sparse set of sensor readings. From ND, it was adapted the strategy of searching for discontinuities between obstacles, taking into consideration their relative distances and their distance to the robot, and including all environmental data present between the robot and the goal.

Concerning the obstacle avoidance technique described in this thesis, increasing the size of the navigation filter would increase the range of the gathered data (obstacle and ground points) to deal with in the navigation module. The number of obstacle-to-ground contact points included in the qualitative occupancy map would increase, covering wider areas, facilitating the anticipation of undesired circumstances, detecting all visible gaps and choosing more efficient paths towards the goal. However, the system could not guarantee escaping from trapping zones without applying the principles of *Traversability* and *Tenacity*.

6.3 The Navigation Strategy

6.3.1 Overview

One of the important differences between the sensorial equipment used in the experiments presented in [136], [137] and [54] and the visual sensor used in the experiments of this thesis is the way the environment is scanned. Laser rangefinders are usually mounted on rotational platforms that permit to scan the environment in considerably wide horizontal ranges. For example, the Hokuyo URG-04LX horizontal measuring area covers 240°. For covering such horizontal ranges with a standard lens it is necessary to capture several frames rotating the camera around its vertical z axis. This point limits, in some way, the adaptation of the ND algorithm from laser to vision, since, unless a fisheye lens or an omnidirectional system is used, a camera will need several shots to cover an horizontal area that can be covered in a single laser scan. Besides, being FOV_H the lens horizontal field of view expressed in degrees, the robot will have a blind zone of $(180°\text{-}\text{FOV}_H)/2$ at both sides, left and right, increasing the need of storing recently captured obstacle information and imposing additional navigation rules to carefully explore both sides of the robot.

Another important difference with respect to the ND-family systems is the quantity of captured points. As a matter of fact, laser surely retrieves much more readings than our system generates obstacle-to-ground contact points. Laser scanned environments can generate dense maps, favoring the detection of obstacles and discontinuities with a considerable accuracy. However, the number of visual points detected by our algorithm is limited to those points where obstacles touch the ground. As stated in previous sections, the aim of the system is not to represent a map of

the environment as accurate as possible, but to deal with a qualitative representation. The purpose will be to represent the environment with the obstacle-to-ground points, in the most reliable way to be able to determine which parts of the scene are occupied and which are free. Later, it would be necessary to infer if a space sufficiently wide to be traversed by the robot can be found between zones occupied by obstacle evidences. As our environmental representation tries to emulate a visual sonar, the sparse set of retrieved obstacle-to-ground contact points will be placed in the navigation filter, analogously to sonar readings are managed in Bug-T^2, but taking into account three important features: a) sections of the filter containing at least one obstacle point will be labeled as occupied, but, b) occupied zones will not be automatically declared as forbidden directions; in some cases, it will be possible to pass through one or two occupied contiguous filter sections, if the distance between obstacle points located in those neighbor sections is much wider than the robot dimensions, and c) the strategy must guarantee escaping from trapping zones, and, to a certain extent, the ability of anticipating hazardous situations.

The principal points of the new visual navigation strategy are:

1. The circular occupancy map (or so called the navigation filter) size will be dynamic, centered at the robot pose and will cover from the current robot coordinates up to the goal position. Obviously, the closer the robot is to the goal, the shorter will be the radius of this ROI. This ROI, or navigation filter, will be divided in polar sectors of γ° each one. The aim is to cover an area as wide as possible at the beginning, and as the robot advances, to consider only those obstacles of the environment that can obstruct the space between the robot and the target.

2. Obstacle information is inferred from detected obstacle-ground contact points. Analogously to Bug-T^2, the world coordinates of all those visual obstacle-ground points that range between the robot and the goal are placed and stored in the accumulative and qualitative navigation filter. This information will be used to determine the *traversability* of the different areas of the environment. As the robot displaces and perceives the different parts of the environment, points are continuously stored in the map until the immediate obstacle is overcome. Then the filter is reset and the process is re-started again. According to Bug-T^2, the coordinates of the obstacle points included

in the navigation filter are always measured with respect to a world-fixed coordinate frame. However, the center of the navigation filter moves with the robot. This means that the orientation of a certain obstacle point with respect to the robot position will change as the robot displaces, thus, changing the sector currently occupied by this obstacle point. The 0° will be in the heading direction, being positive angles to the left and negatives to the right.

3. Each execution of the algorithm involves a pair of images and outputs a steering vector.

4. When a sector of the navigation filter is occupied by several points, the one closest to the robot will be the most restrictive in terms of estimating the level of collision risk.

5. To guarantee escaping from trapping situations, and of course, to overcome ordinary obstacles, the principle of *tenacity* will be applied until the goal direction is free again or the obstacle has been completely surpassed.

   Now, the scope of application of the *tenacity* principle has been re-defined. Gap-based navigation makes unnecessary on many occasions, and difficult on others, considering the same situations for applying the tenacity principle as described in Bug-T^2. Now, the robot can go through sectors occupied by obstacle points, if the spacing conditions described below are true. *Tenacity* will be applied as long as the goal direction is still blocked and obstacles that are currently been avoided are still in the portion of the navigation filter which comprises angles γ such as: $-90° < \gamma < 90°$, always with respect to the robot direction of motion. This is similar to checking if the immediate obstacle is or is not behind the robot. So, either if the immediate obstacles are behind the robot or the path to the goal is free, the navigation filter is reset and the *tenacity* concept is no longer applied. Continuity in obstacle evidences along neighbor sectors means dealing with the same immediate obstacle. Reseting the navigation filter implies immediately rebuilding it as the robot moves towards the goal and perceives again the environment. Although the path to the goal can still be occupied by more obstacles, the navigation algorithm will search again for gaps to go through and *tenacity* will be applied again to overcome the upcoming obstacles.

   As now the navigation filter covers the whole area in between the robot and

the goal point, it would be unnecessary considering the additional criteria detailed in section 6.2.1, concerning situations where the goal is found in a previously banned sector, being the goal in front of the obstacle that occupies that section [5].

6. Analogously to [137], maps are continuously analyzed searching for gaps under a certain criterion. To define this criterion, a set of different situations is modeled. Each situation involves a certain configuration of robot, obstacles and goal, and has associated a set of predefined actions or behaviors. These actions will be conscientiously designed to efficiently proceed in front of three main groups of characterized situations: 1) Low Risk Obstacle Avoidance, which includes managing paths through obstacle discontinuities, 2) Go to Goal, and 3) High Risk Obstacle Avoidance, which means immediately avoiding any obstacle which is inside the security area. Next section details all the situations defined with their associated actions.

7. A safety distance S_{dist} will be defined around the robot as the maximum distance between the robot and an obstacle point to be considered as a robot-obstacle configuration of high collision risk. This distance defines and delimits the robot security zone. S_{dist} might be valued as the radius of the minimum circumference containing the whole robot (in case the real robot is not completely circular) plus an additional safety distance for covering blind zones close to the robot. The presence of an obstacle inside the security area will immediately activate the situation of High Risk Obstacle Avoidance. In this situation, the robot will take the direction computed by the purely reactive module, which can be, for instance, the classic Potential Fields method.

6.3.2 The Navigation Strategy: Situations and Actions

The objective of this section is to describe a set of situations, and a set of behaviors related to each different identified situation. Each situation uniquely describes a certain configuration involving the robot, the nearby obstacles and the goal point.

Previously to describe all possible scene configurations and their corresponding decisions, some assumptions must be done:

- The navigation architecture involves a cooperative coordinator which manages a set of basic behaviors, namely, Go to Goal and Avoid Obstacle, in low or

high collision risk mode.

- Scenarios will contain commonly used and shaped obstacles which generate clusters or aggregations of obstacle-to-ground contact points. Points corresponding to the same obstacle will occupy several contiguous sectors of the navigation filter (a sector will be sized between 10° and 15°), and they will be at a nearly constant distance to the robot. In fact, it is assumed that the relative position of an obstacle point with respect to the robot has a very small change between two subsequent frames.

The *Go To Goal* module is the first of being executed, giving the direction towards the target. The robot takes this direction as the starting point of the process for deciding the next movement. Then, the algorithm analyzes which of the next possible situations can come out:

(A) - GF (Goal Free): The sector of the navigation filter containing the goal direction is free of obstacles. If the sector is relatively narrow, for example, between 10° and 15°, evaluating if the sector of the goal direction is or is not free is approximately equivalent to evaluate if the goal direction is free of obstacles. If the sectors were wider or too wide (for example 40° or 65°), the fact that the one containing the goal direction is occupied with obstacles would not significantly mean that the goal direction is not free.

Some verifications have to be done before the navigation module decides whether the robot can take the goal direction or can not. Starting at the sector corresponding to the goal direction, the algorithm searches to the left for the first sector of the navigation filter occupied by an obstacle point, and analogously, it searches for the first occupied sector at the right of the goal direction. Let us denote as:

- P_L: the closest obstacle point at the left of the goal direction,
- P_R: the closest obstacle point at the right of the goal direction.
- θ_{goal}: the goal direction with respect to the robot pose,
- θ_{PL} and θ_{PR} the polar directions of P_L and P_R, respectively, with respect to the robot pose,
- d_1 the euclidean distance between the robot and P_L, and d_2 the distance between the robot and P_R, and,

- d_{RL} the euclidean distance between P_R and P_L, defining the length of the gap or discontinuity.

If $d_{RL} < 2S_{dist}$, then the gap will be labeled as *forbidden*, otherwise the gap will be permitted and the robot will be directed, in principle, to pass through it following the direction θ_{goal}.

However, following the goal direction can lead the robot to collide with one of the obstacles that delimit the gap. To avoid the collision the robot will have to deviate from the direction θ_{goal} to pass, at least, tangential to the closest obstacle point. Let us denote ϵ as the angle of minimum turn to pass tangential to the closest obstacle point, defined as:

$$\epsilon = arcsin(R/d_i), \tag{6.1}$$

being R the radius of the minimum circumference containing the robot and d_i either d_1 or d_2 (see figure 6.5).

Slightly bigger turns away from this tangential motion will guarantee no collision. Let us redefine ϵ as:

$$\epsilon = arcsin(S_{dist}/d_i), \tag{6.2}$$

being S_{dist} the security distance and d_i either d_1 or d_2 (the closest to the robot).

If θ_{goal} fulfills the requirement

$$|\theta_{goal} - \theta_{PX}| > \epsilon \tag{6.3}$$

being θ_{PX} the direction of either P_L or P_R (take the closest to the robot) with respect to the robot, then the direction of motion will be $\theta_{mot} = \theta_{goal}$. If requirement 6.3 is not accomplished, then $\theta_{mot} = \theta_{PX} \pm \arcsin(S_{dist}/d_i)$ (see figure 6.6).

In this case θ_{mot} points to the direction closest to the goal that satisfies equation 6.3. The sum or the subtraction will be applied depending on the relative orientation of the robot with respect to the goal (see figure 6.6).

If P_R and P_L are one behind the robot and the other in front of it, the algorithm will control the directive 6.3 only for the point which is in front of the robot because this will be the only one limiting the robot motion towards the target.

(B) - GO (Goal Occupied): The sector of the navigation filter containing the

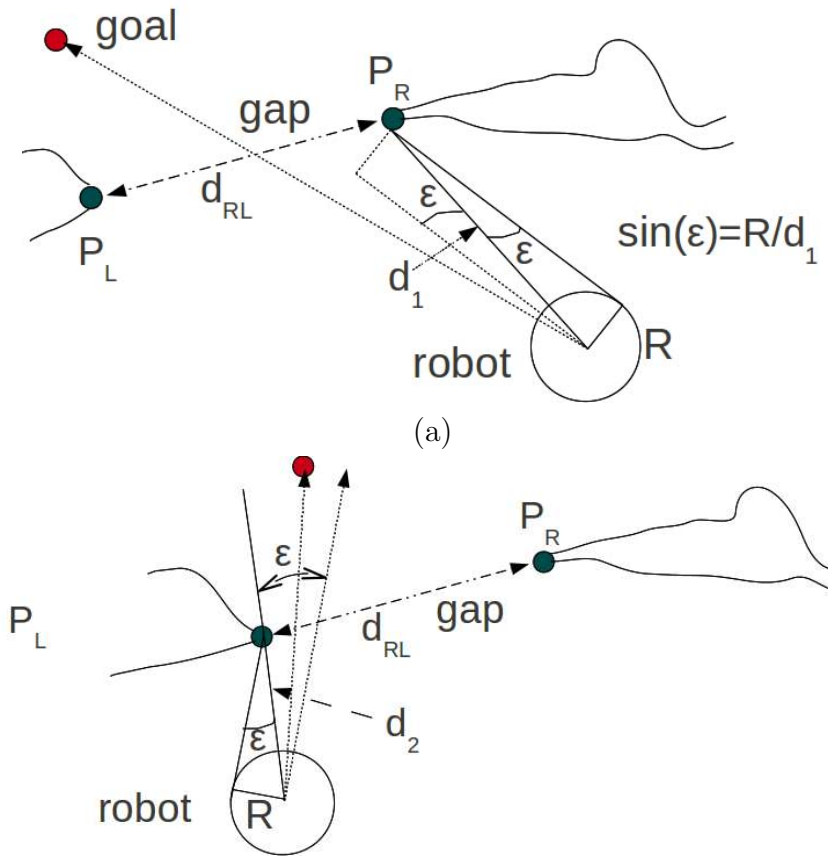


Figure 6.5: The goal direction is free of obstacles and the gap is wide enough for being traversed by the robot. (a) Example 1: the robot faces the goal and the distance between each point over the goal direction and the obstacle point P_R is bigger than the radius of the robot. Equation 6.1 is always fulfilled. The robot can keep the goal direction. (b) Example 2: the robot faces the goal but, at a certain moment, equation 6.1 is not fulfilled. The robot must deviate its trajectory to the right in order to avoid a certain collision. A minimum turn of ϵ with respect to P_L guarantees the hitting avoidance.

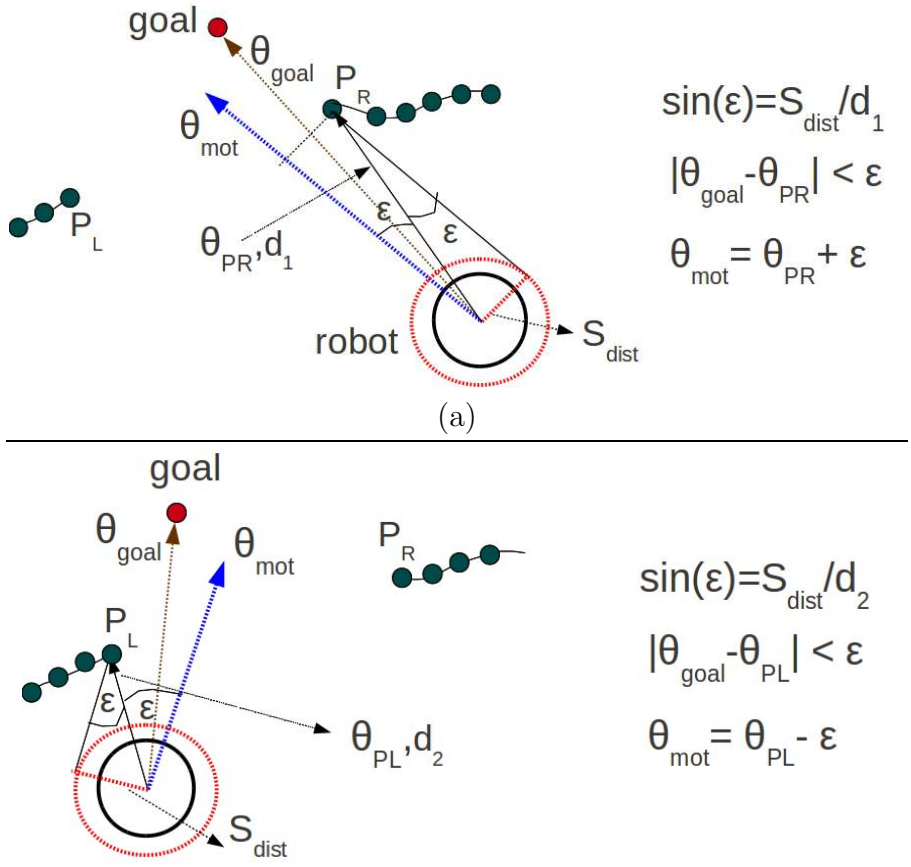


Figure 6.6: The goal direction is free of obstacles and the gap is wide enough for being traversed by the robot, but the robot can not face directly the target. In red, the distance of security and the security area surrounding the robot (black double-thickness circle). In brown the goal direction and in blue the final motion. In both cases the robot will not hit the closest obstacle point since the security area will pass tangential to it. (a) case 1: the robot faces the goal from the right. ϵ is added to θ_{PR}, since positive angles are at the left and negatives are at the right. (b) case 2: accordingly, the robot faces the goal from the left and ϵ is subtracted from θ_{PL}

goal direction is occupied by obstacle points. Again, if the sector width is between 10° and 15°, it can be assumed that the goal direction is blocked by an obstacle and the robot can not move directly towards the target.

If the obstacle is inside the security area, the robot must urgently avoid the closest obstacle. In that case, the direction of motion will be the result of applying the Potential Fields method.

If obstacles are out of the security area, the system will propose two possible solutions for the next steering vector, similarly to Bug-T^2.

The idea is to find the gaps located as close as possible to the goal direction, at both sides of it. Gaps will be found either in sectors free of obstacle points delimited at both sides by occupied sectors, or in n contiguous occupied sectors (where $n \in \mathbb{N}$, with $n > 0$).

For the first solution, the algorithm will search, from the goal direction, sector by sector, clockwise, if one of the next cases is fulfilled:

- Case (1) - n contiguous sectors of the navigation filter are occupied by obstacle points.

  Find two obstacle-to-ground contact points, P_L and P_R, both in front of the robot (the front of the robot will be the portion of the navigation filter comprising from $-90°$ to $+90°$ with respect to the robot orientation), with no other points in between, satisfying two conditions:

  1. $\mid P_L - P_R \mid > 2S_{dist}$,
  2. $d_1 > S_{dist}$,

  or $d_2 > S_{dist}$, depending on which point, P_L or P_R, is closer to the robot. If two points fulfilling these requirements are found, it will be assumed that they form a permitted gap which the vehicle can pass through. If several permitted gaps are found in the set of occupied contiguous sectors, take the one closer to the goal direction as the first candidate to be traversed by the robot.

  The direction of motion will be the direction corresponding either to P_L or P_R, depending on which one is closer to the robot, plus or minus ϵ [2], in such a way that the resulting vector moves towards the furthest obstacle point (see figure 6.7).

[2] being ϵ the minimum turn to fulfill the equation 6.3

If the sector corresponding to the computed steering vector is occupied by an obstacle point which, after the robot motion, can fall inside the security distance, the gap is no longer valid, the motion is not performed and the algorithm searches for another gap.

If no gaps are found, the sector/s are banned and the algorithm keeps searching clockwise for another set of occupied sectors or another set of sectors which fulfills case (2).

- Case (2) - n contiguous sectors free of obstacles bordered by one occupied sector at each side. Each occupied sector at both sides of the free zone delimit the discontinuity, which has to be located in front of the robot. Let us denote again the points P_L and P_R as the extremes of both obstacles that define the aperture. In this case, P_L and P_R can be determined, for example, as those points at both sides of the empty zone that are closer to the polar direction corresponding to the center of any of the free sectors.

  The motion rules to control the robot towards the free space will be the same as exposed in the previous case:

  if $\mid P_L - P_R \mid > 2S_{dist}$ and $d_i > S_{dist}$ (i=1 or i=2), then the gap is labeled as permitted, else the gap is labeled as forbidden and the corresponding sector/s are banned.

  The direction of motion will be, again, $\theta_{mot} = \theta_{PX} \pm arcsin(S_{dist}/d_X)$, being θ_{PX} the direction with respect to the camera of either P_R or P_L (the closest to the robot), and d_X the distance between P_X and the robot, always moving the steering vector from the closest obstacle point to the furthest obstacle point (see figure 6.8).

  Equally to the previous case, if the resulting vector steers the robot towards a sector occupied by an obstacle point which shall fall inside the security distance, the gap is discarded.

- Case (3) - One of the points, either P_L or P_R, of an existing gap is in front of the robot and the other one is behind it. This case represents the situation in which one of the obstacles delimiting the gap is behind the robot and the other is still in front of it.

  Let us denote θ_{PX} the orientation with respect to the robot of P_X, being P_X either P_L or P_R, the one which is in front of the robot. The direction of

motion θ_{mot} will be $\theta_{mot} = \theta_{PX} \pm \epsilon$, with $\epsilon = arcsin(S_{dist}/d_X)$, being d_X the distance between the robot and P_X. The summation or subtraction will be chosen in order to displace θ_{mot} in the opposite polar direction of the obstacle point which is in front of the robot (see figure 6.9). As it can be seen in figure 6.9-(a), in the front of the robot there is only one compact set of obstacle points. Obviously, even being P_L closer to the robot than P_R, P_R is the only gap extreme that plays an influence in the motion control since it is the only one blocking the path towards the goal. The situation is analogous in figure 6.9-(b), where only P_L will be taken under consideration for computing the steering vector.

In practical implementations, prohibiting or permitting sectors relatively narrow, lets say, for example, between 10° and 15°, will be more representative of the navigation filter occupancy state than managing wider sectors (for example between 45° and 60°), since the later could be partly occupied by obstacles and partly free, but they would be identified with a unique label.

For the second motion alternative, starting on the goal direction, the algorithm searches again one of the aforementioned situations, but now counterclockwise.

At this point, and in case the goal direction is not free, the algorithm has computed two possible steering vectors, one to the right and another one to the left of the goal direction, both pointing to a sector of the navigation filter that fulfills one of the cases described above. The next step would be to choose one of the two proposed solutions. The decision will be made according to the next criterion:

1. if the current obstacle is still not left behind, the tenacity principle is applied and the direction of turn (clockwise or counterclockwise) will be the same as the previous decision,

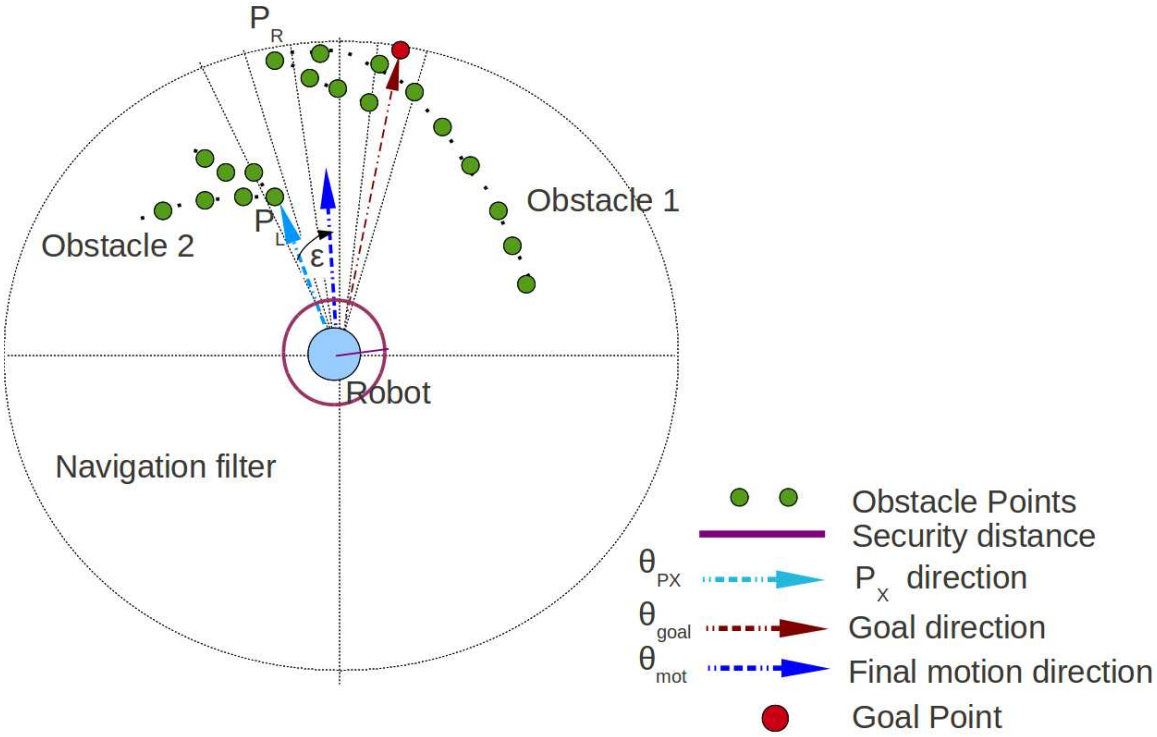
2. contrarily, if there is no previous decision because the navigation filter was recently reset, the direction of motion closer to the goal direction will be chosen.

The idea is illustrated in figure 6.10. Nearly all sectors between the robot position and the goal point are occupied by obstacle evidences. In plot (a) two possible directions of motion are proposed: V1 (in blue) and V2 (in red). V1 is chosen because it is closer to the goal direction. In (b), again, two possibilities are generated, V1 and V2. Applying the tenacity concept the robot chooses again V1. In (c)

the direction of motion is still occupied. The immediate obstacle has been already overcome. V1 is chosen again since it is the option closest to the obstacle direction. The gaps at the right of the robot are not permitted because they are narrower than the security distance. The first available gap at the right is indicated as V2 in red. In (d) the robot still chooses V1 applying the tenacity concept, since the upcoming obstacle has still not been overcome. In (e), the goal point is perfectly visible, its direction is completely free and the current obstacle has been overcome. Immediately, the navigation filter and the tenacity boolean condition are both reset. The robot faces directly to the goal. Notice how the navigation filter moves with the robot and reduces its dimensions as it approximates to the target, checking each time only the piece of the environment between the robot and the goal point.

The algorithm is valid to avoid a cyclic situation in some intricate scenarios, for example if the goal point is inside some G-shaped obstacles (see an example in figure 6.11). However, although the strategy permits the robot to reach the target in numerous theoretical and practical situations, this version of the algorithm can not formally guarantee convergence in all cases, since it is not applying the Bug-T^2 own rules that formally and practically assure convergence.

Finally, figure 6.12-(a) shows a simulation of escaping from a U-shaped obstacle much bigger than the camera field of view. The robot is represented by a small blue circle. The navy blue vector is the motion vector chosen at every position and the light blue vector represents the rejected possibility. The black dotted line represents the big U. The triangle over the blue circle represents the horizontal camera field of view. The U is so big compared with the robot and the FOV_H dimensions that the camera is unable to perceive the frontal wall until this is inside the FOV_H. As seen, the portion of the wall perceived by the camera corresponds to a very small part of the U. The first motion vector (from point (A) to (B)) corresponds to the closest to the goal direction (in red). As the robot moves to the left applying the principle of *tenacity*, it perceives more parts of the wall, storing all points in the navigation filter. Between two motions, the perceived part of the environment will overlap since the FOV_H is sufficiently wide to see in one frame part of the environment seen in the previous frame. Due to this continuity in the obstacle points, the algorithm will consider that the obstacle is still the same and that it has not been overcome. At each position, the motion vector will point to the most recently perceived part of the obstacle, always to the left of the FOV_H. At point (C) the robot will have perceived all the U from point (A) to point (C) and will have stored all these data



(a)

Figure 6.7: The Navigation Strategy: the goal direction is occupied. There are two contiguous sectors occupied by two obstacle points with a gap in between. $\theta_{mot} = \theta_{PL} - \epsilon$

into the navigation filter. The robot has not perceived the right part of the U, but it does not need it to reach the goal. Following the *tenacity* principle the robot will move out of the U towards the goal as represented in the trajectory plot in green.

The main points of the navigation algorithm detailed above are presented in figure 6.13. The schema represents the principal steps on the decision tree and the corresponding actions performed in each case.

6.4 Experimental Results

In this stage, the platform mission was to move from a starting point to a goal, or to a set of subsequent goal points. Except for the fact that SIFT features were substituted by KLT features (without causing any significant variation in the classifier performance), the system used the same operating conditions described in section 4.3.1, in terms of: camera, robot speed, focal distance, lens height, frame

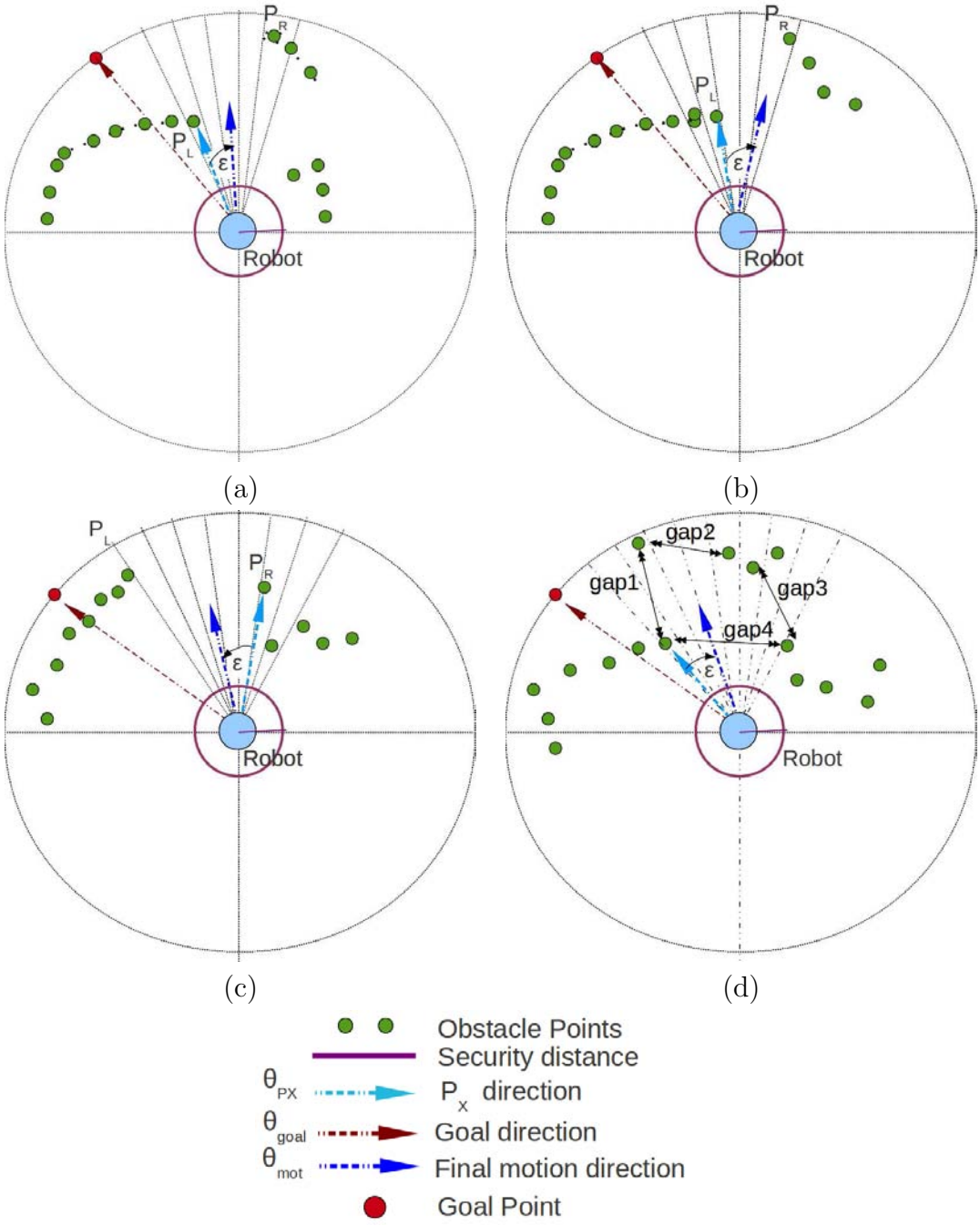
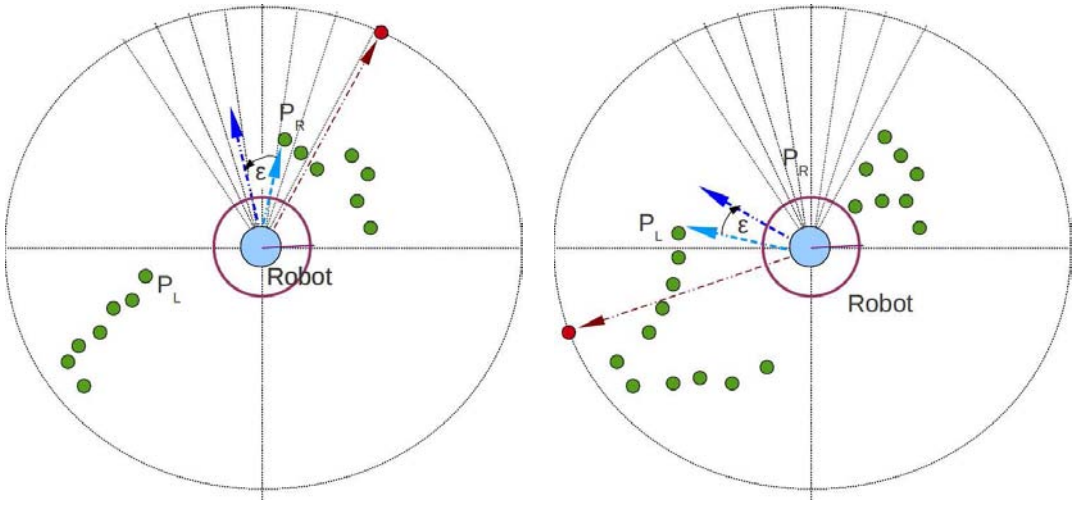


Figure 6.8: The Navigation Strategy: the goal direction is occupied. (a), (b) and (c): there is a discontinuity longer that the robot security distance. (d) Gaps 1 and 4 are permitted while gaps 2 and 3 are forbidden since they do not exceed the robot security distance.



(a) (b)

Figure 6.9: The Navigation Strategy: the goal direction is occupied. One of the extremes of the gap is behind the robot. (a): The direction of motion $\theta_{mot} = \theta_{PR} + \epsilon$, where $\epsilon = arcsin(S_{dist}/d_2)$. (b): $\theta_{mot} = \theta_{PL} - \epsilon$, where $\epsilon = arcsin(S_{dist}/d_1)$.

rate, image resolution and intrinsec camera parameters for image undistortion. The approximate diameter of the robot used (a Pioneer 3DX) is 50 cm and according to the camera settings (height, focal length, vertical and horizontal FOV (*Field of View*), and its position on the robot) there were nearly 60 cm of blindness between the Pioneer front and the bottom of the camera vertical field of view. Considering that the robot can rotate around its z axis, this blind area extents to a semi-circular portion centered on the robot (see figure 6.14). With a focal length of 3.7mm, a resolution of 1024×768 squared pixels and a pixel side of 4.65μm, the horizontal FOV (FOV_H) is 65.5° and the vertical FOV (FOV_V) is 51.5°. The camera height was imposed by the platform and the gimbal that append the camera to the robot, but the tilt angle was chosen to obtain a reasonably short blind area in front of the robot combined with a considerable vertical visual range. All these variables directly conditioned the criteria for determining the size of the S_{dist}, and the value of S_{dist} in turn is determinant to define which gaps are traversable and which are not. The robot had very short time to react in front of obstacles or gaps closer than 85 cm (which includes the blind area plus the approximate robot radius). At shorter distances, gaps, can be either not properly perceived or too close for the robot to correctly maneuver through them. Consequently, the S_{dist} was fixed on 95 cm in order to guarantee that the robot had enough time and space to perceive and

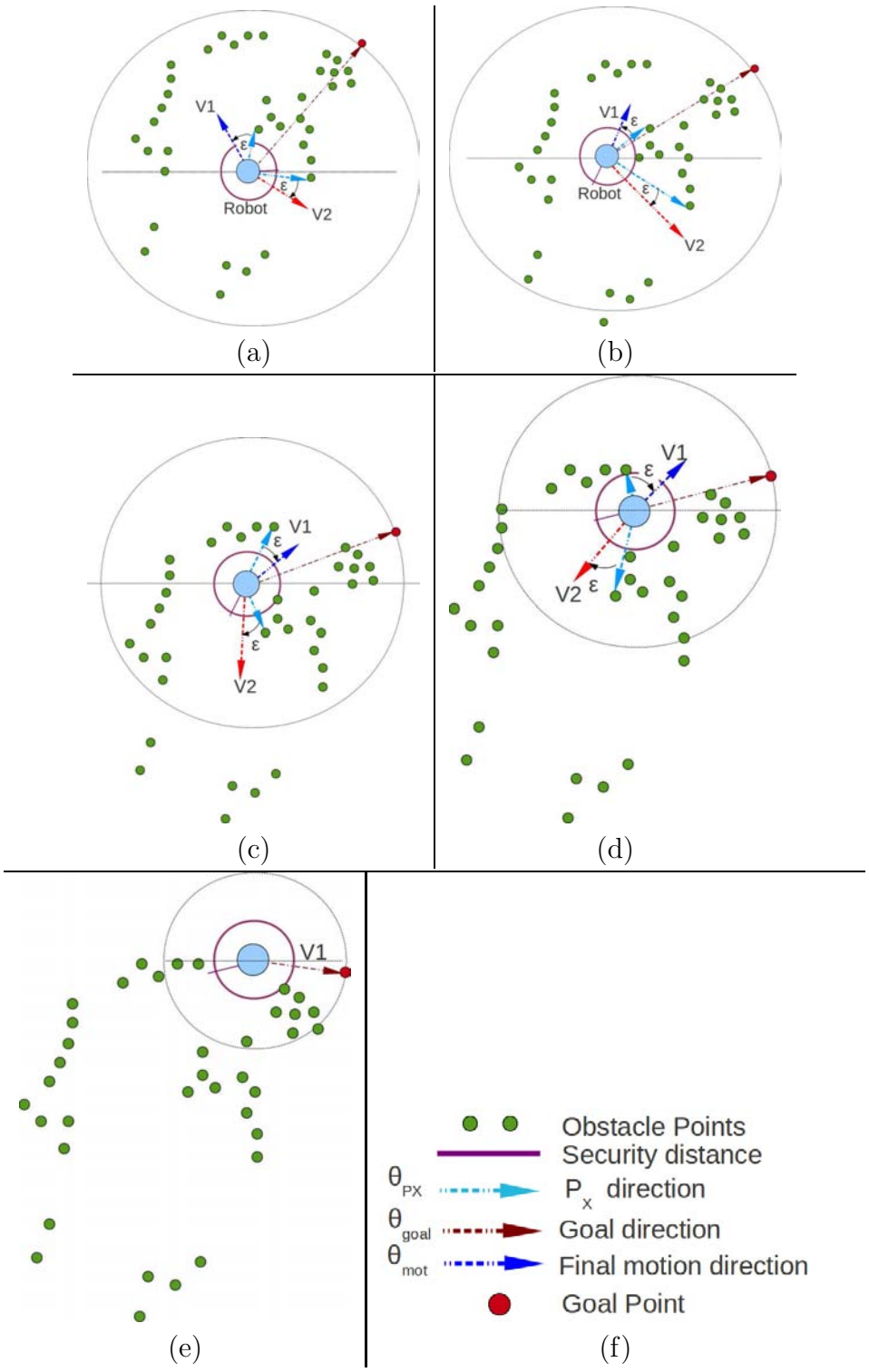


Figure 6.10: From a starting to a goal point passing through all available gaps. (a), (b), (c) and (d): The robot goes through the permitted gaps, applying the criteria of minimum turn with respect to the goal direction and Tenacity. (e) The robot faces directly to the goal.

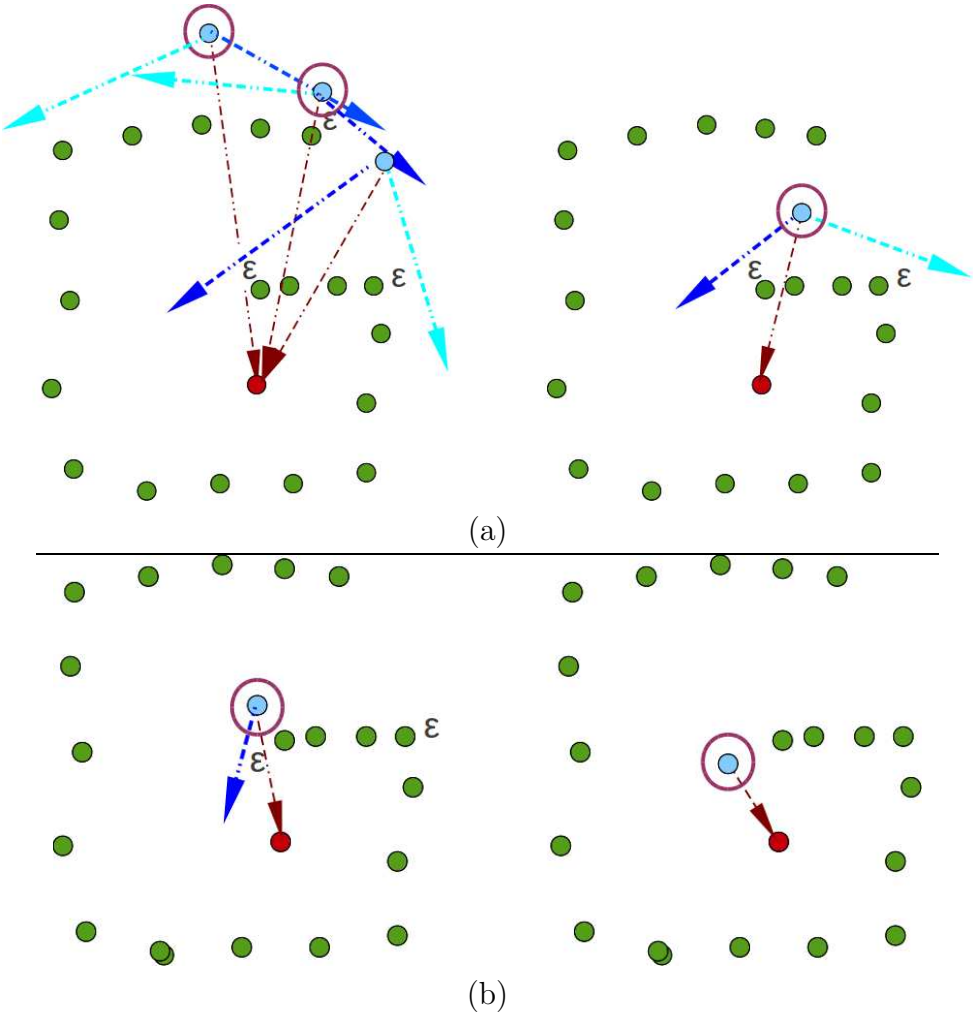
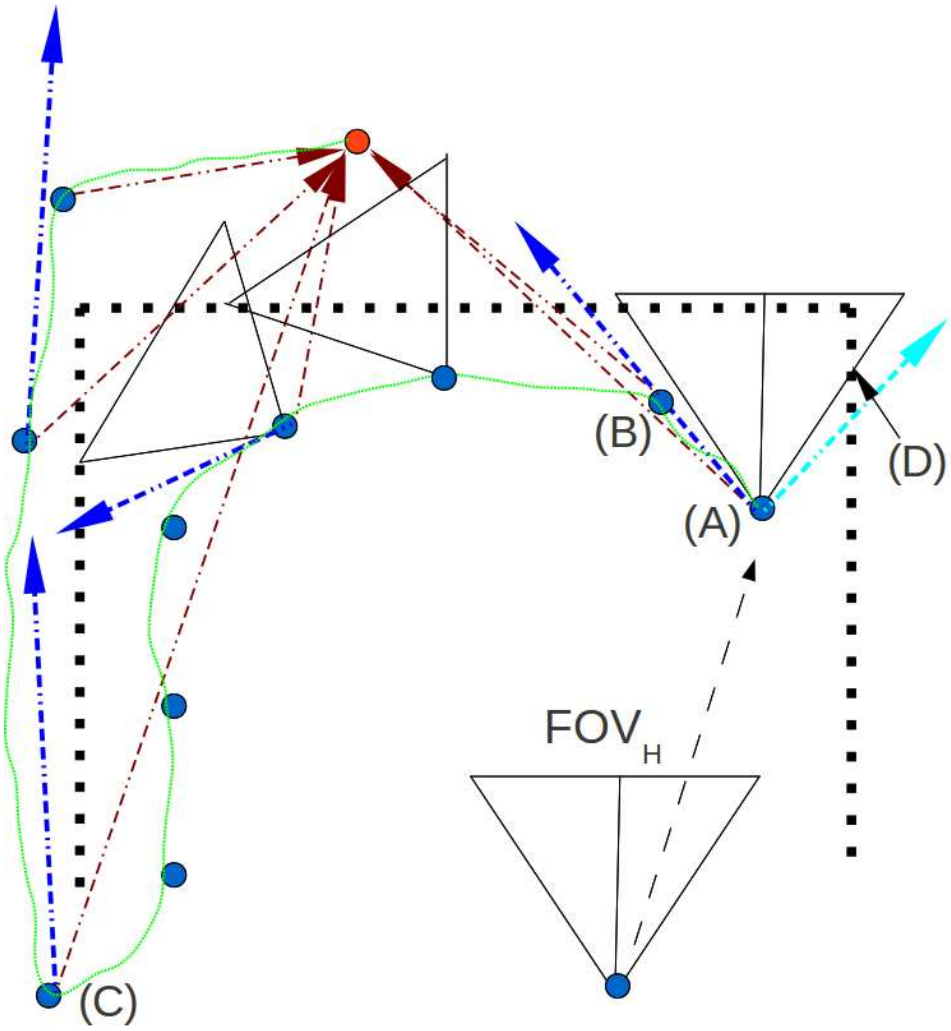


Figure 6.11: The goal is inside a G. The blue point is the robot and the red point is the goal. The red vector indicates the goal direction. For each robot position, the navy blue vector indicates the chosen direction of motion, and the light blue vector indicates the rejected option. Take into consideration that the navigation filter only considers obstacle evidences from the robot pose up to the goal point. (a) In the two first movements, the tenacity principle is applied, until the immediate obstacle is behind the robot. In the third position, it applies the criterion of minimum distance to the goal direction. In the plot of the right, tenacity is applied until the goal is directly visible. (b) The goal point direction is free, but the robot has to slightly deviate from it to guarantee that there will no be collision on the obstacle extreme. Obstacles behind the goal are not considered.



(a)

Figure 6.12: Escaping from a huge U-shaped obstacle applying *traversability* and *tenacity*.

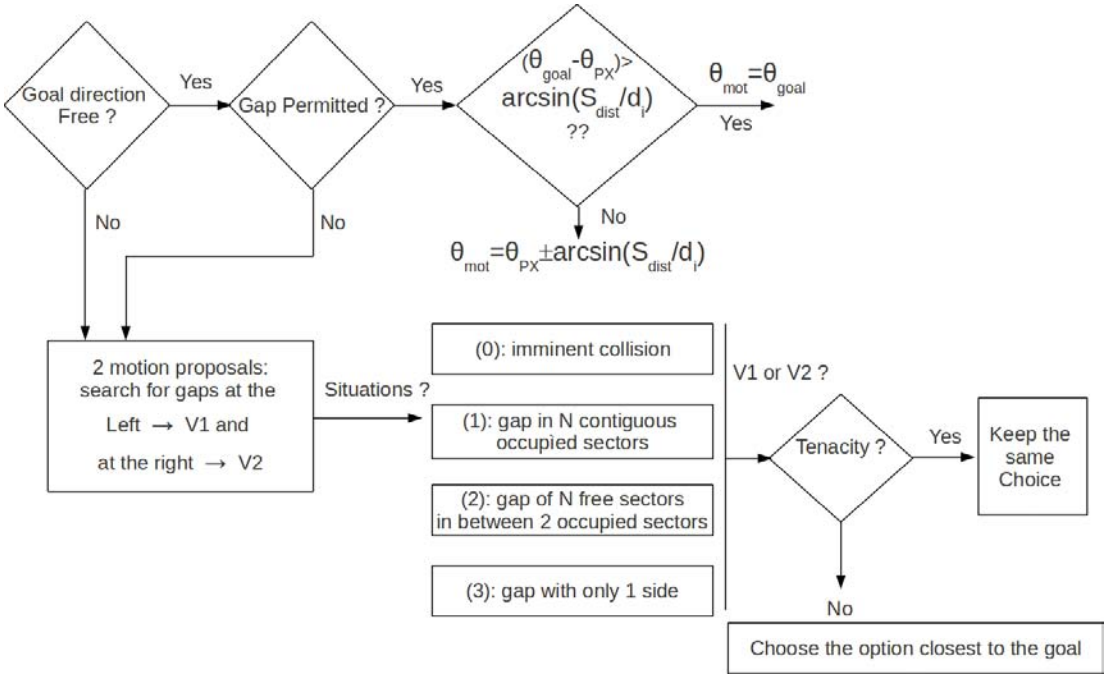


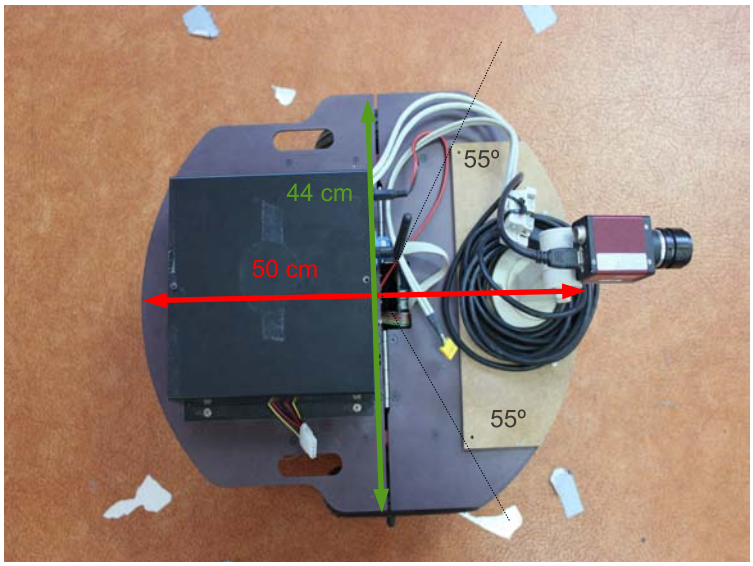
Figure 6.13: A simplified overview of the complete navigation algorithm in a schematic structure.

react. Besides, gaps shorter than 95 cm were labeled as forbidden. The width of one navigation filter sector was 12°, which means a total of 30 sectors. This value experimentally demonstrated to be adequate for the navigation performance run by our particular robot.

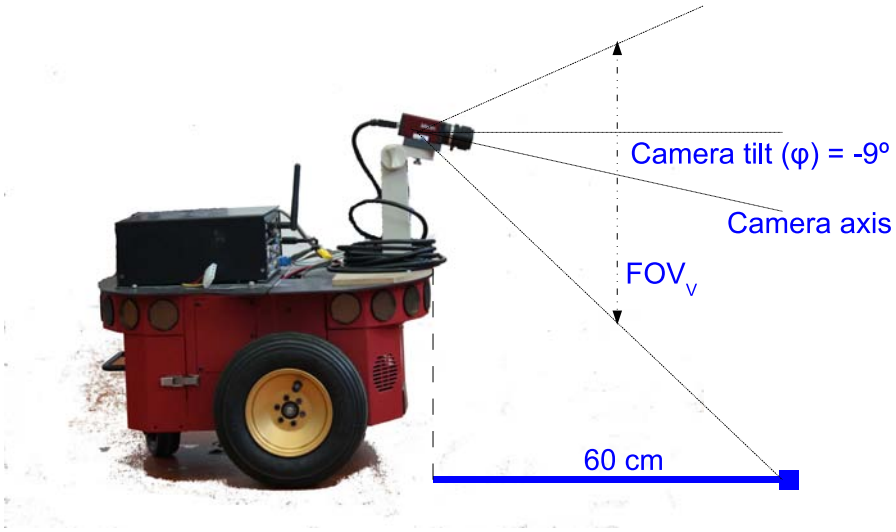
The limited camera FOV reinforced the idea of storing in memory the environmental information during a certain period of time. It is fundamental in order to plan the robot motion using data from already visited portions of the environments but currently not visible.

In all the experiments included below, the robot detected the obstacles and set the occupancy of the navigation filters according to the procedures described in section 4.3. However, the motion orders were generated applying the strategy reported in the previous section of this chapter. In all plots, the starting point is colored blue, the goal points are colored in red, the trajectory points are pictured as tiny empty red circles and all length units are expressed in *mm*.

At the beginning, first navigation experiments were planned inside the laboratory already known as Scenario 1 in section 4.5. There, scenarios were particularly prepared to test certain robot behaviors that were considered to be fundamental in



(a)



(b)

Figure 6.14: Robot platform (a): Approximate dimensions of The Pioneer 3DX. (b): The robot with the camera. Blind zone and FOV_V

the system, such as, for example, escaping from potential trapping areas or navigating through environments densely occupied by obstacles. In most of the cases, the robot had only one possible free path to reach the goal.

Figure 6.15 shows some experiments conducted in the laboratory. Plot (a) shows a route where the goal point is just behind a wall, and there is only one free path to reach it. Plot (b) shows an example of going and returning to the same departure point. Both objectives are always behind a set of obstacles. Plots (c) and (d) show two examples of avoiding local minimas. As it has been seen, U shaped obstacles are areas where autonomous agents typically get continuously stuck under the influence of attractive-repulsive dual-action forces.

In both cases, local minimas were foreseen and the vehicle circumnavigated them reaching the goal point without any incidence. If the robot had fallen into a deep trapping canyon, due to, for example, the impossibility of detecting or anticipating it in a reasonably number of frames, the application of the *tenacity* concept would help to scape from it.

Figure 6.16 shows some pictures taken during these experiments conducted in the lab. Obstacle points are colored red, ground points circled in blue, and the computed obstacle-to-ground contact points are shown in pink.

The following experiments intended to simulate an ordinary situation in which the robot had to execute a mission over two defined points in a relatively crowded daily used environment inside a building. All tests tried to demonstrate the suitability of our navigation strategy in common scenarios such as, for example, offices, corridors, halls, sidewalks, warehouses, where it is unlike to find intricate obstacles such as a maze or a G.

Figures 6.17 and 6.18 plot 4 different trajectories conducted by the robot in a considerable busy hall, located inside a university building (the scenario known as Scenario 3, where part of the experiments shown in section 4.5 were conducted).

Obstacles have been clearly labeled in all pictures.

Notice how the robot is able to pass through all available apertures in order to advance along the free space towards the goal. Figure 6.17-(b) shows 3 goal points, enumerated according to the order in which they had to be reached. Figure 6.19 shows some images recorded during experiments of figures 6.17 and 6.18 with the obstacle and ground points and with the corresponding edge maps highlighting the obstacle contours. In all cases every obstacle-to-ground contact points detected between the current robot pose and the goal point was included in the navigation

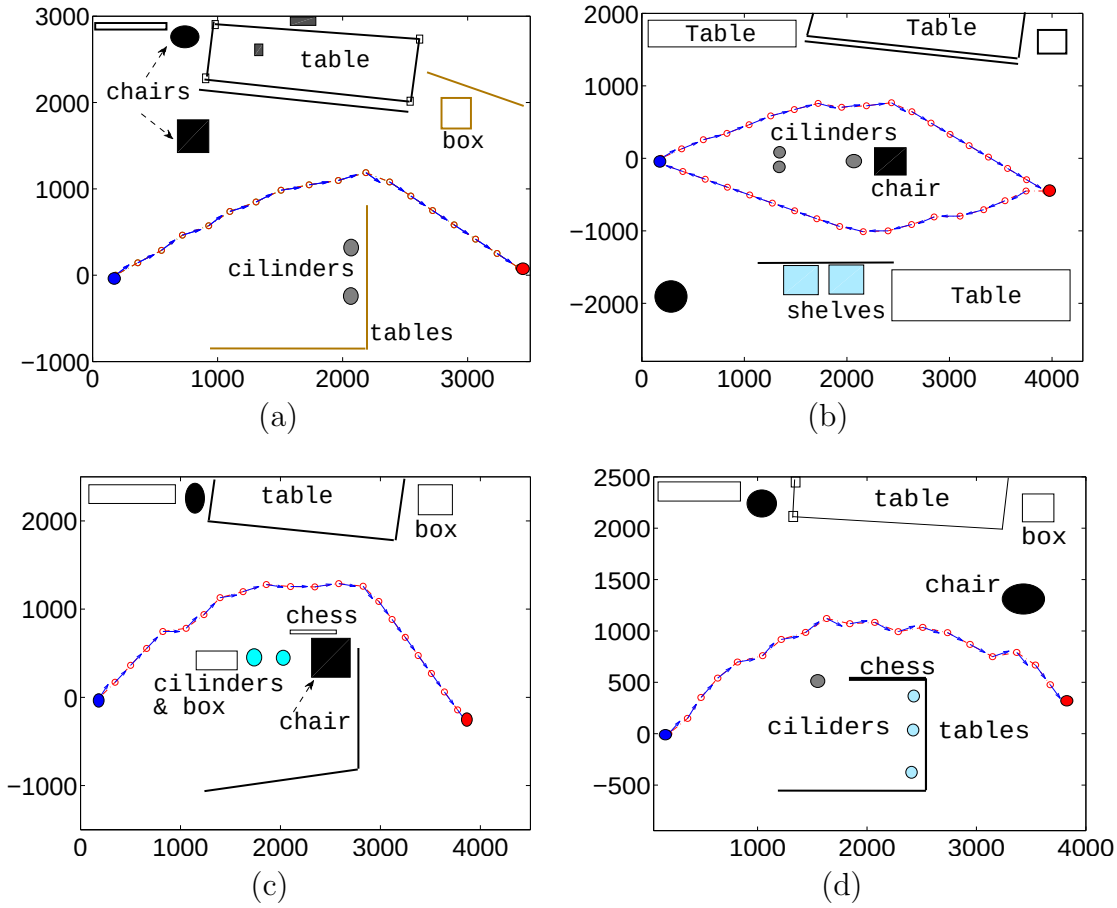


Figure 6.15: Scenario 1: (a): from starting to the goal point in a relatively dense environment. (b): Going and return. (c) and (d): avoiding trapping zones. All length units expressed in mm.

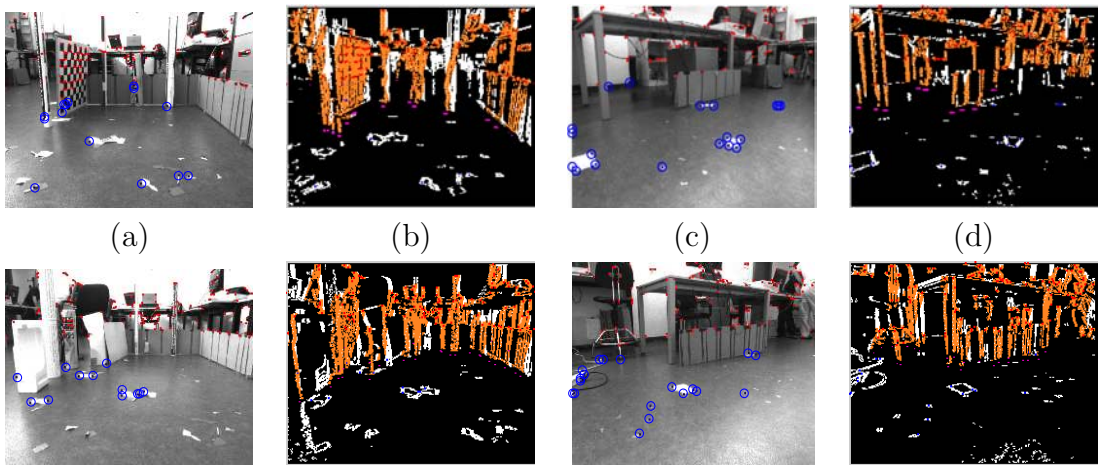


Figure 6.16: Avoiding a trapping zone. (a), (c), (e)nd (g): frames with obstacle and ground points. (b), (d), (f)and (h): edge maps with obstacle boundaries in orange and the obstacle-to-ground contact points in pink.

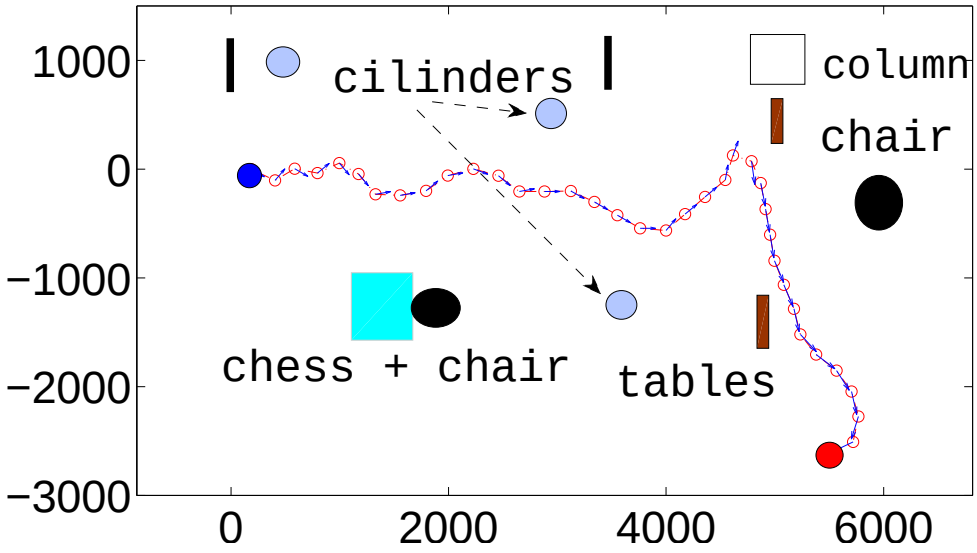
filter.

These tests evidence the capacity of the algorithm to provide a robust obstacle detection module suitable for being integrated in reactive navigation architectures.

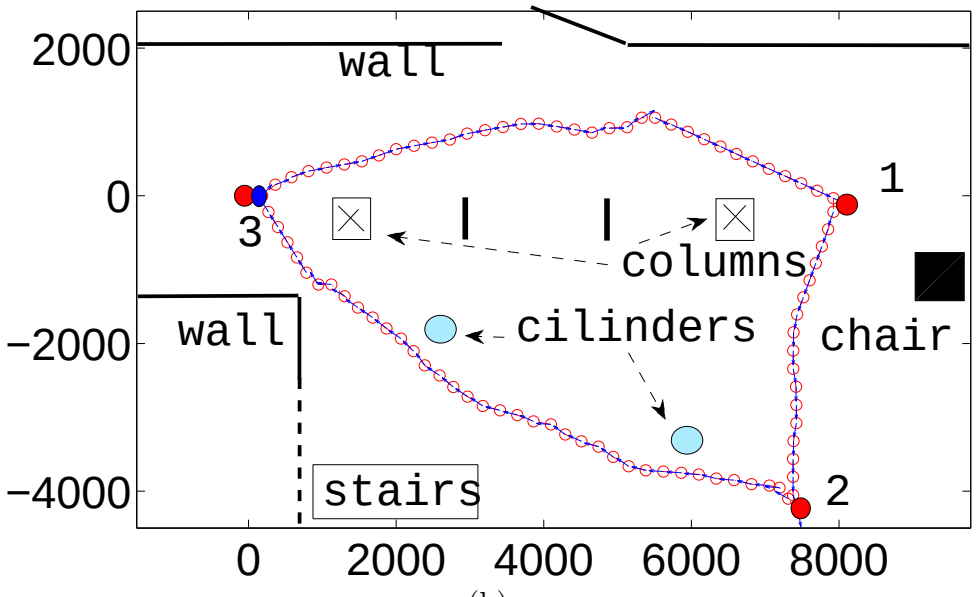
Navigating through long and relatively narrow spaces with a considerably number of randomly placed obstacles could be a challenge situation for a robot with the dimensions of the Pioneer 3DX. In the experiment of figure 6.20, the starting and goal points were separated 25 meters in the x axis of a long corridor. This corridor corresponds to Scenario 2, also described in section 4.5 and was filled with obstacles distributed along the whole path. Again, the robot was able to cover the whole distance avoiding all the incoming obstacles using the so far described obstacle avoidance and navigation algorithms. Walls of the corridor are clearly indicated as well as obstacles colored in blue, soft blue and brown.

In some of these plots, there is a slight difference, caused by the odometry errors, between the red circle, which represents the programmed goal point, and the point where the robot really ends its trajectory.

All the position data used to plot robot trajectories and to calculate the world coordinates of the ground points was obtained, up to now, from the robot wheel odometers. But, as it has been previously mentioned in the Introduction of this dissertation, the pose estimation obtained from proprioceptive sensors is unreliable at relatively long distances since it is prone to drift. Visual localization is a good



(a)



(b)

Figure 6.17: Experiments in Scenario 3: (a) One fixed goal point. (b) Three consecutive goal points forming a closed loop. Length units expressed in mm.

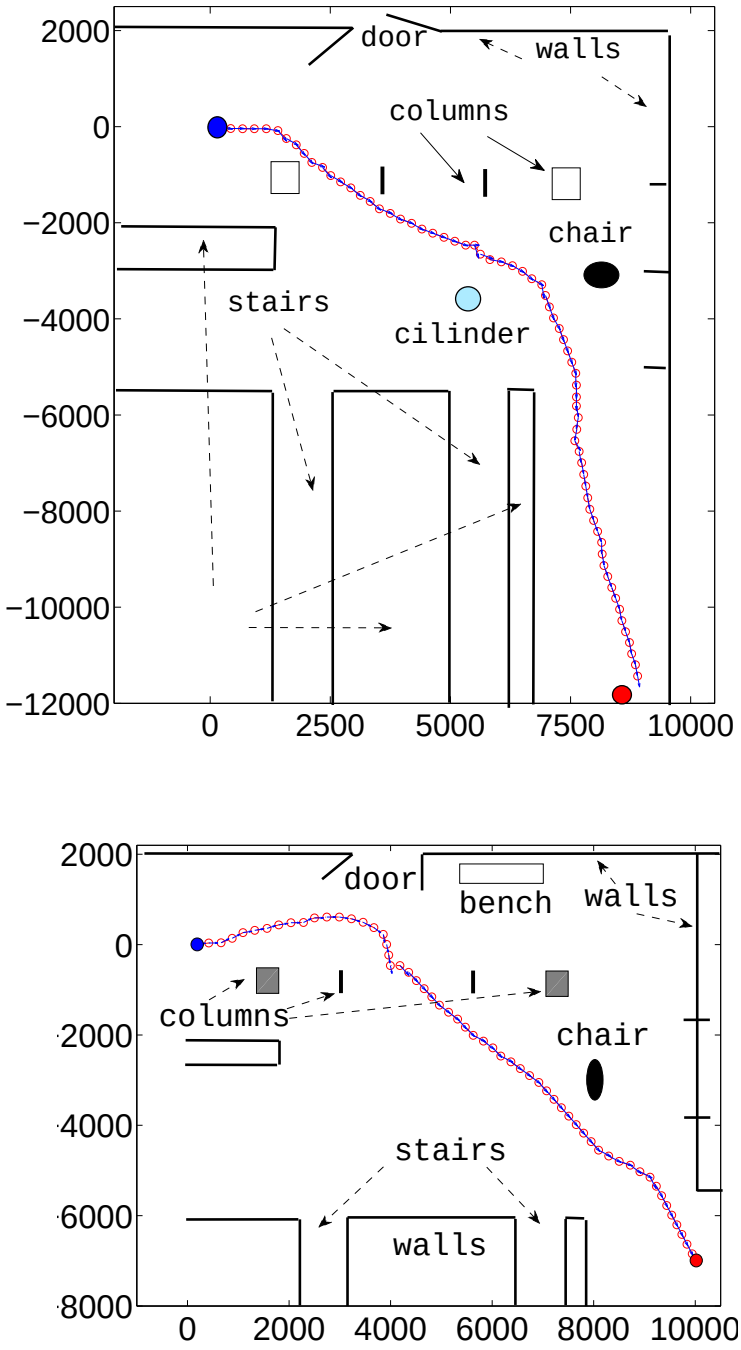


Figure 6.18: Scenario 3: one fixed goal point, longer trajectories. Length units expressed in mm.

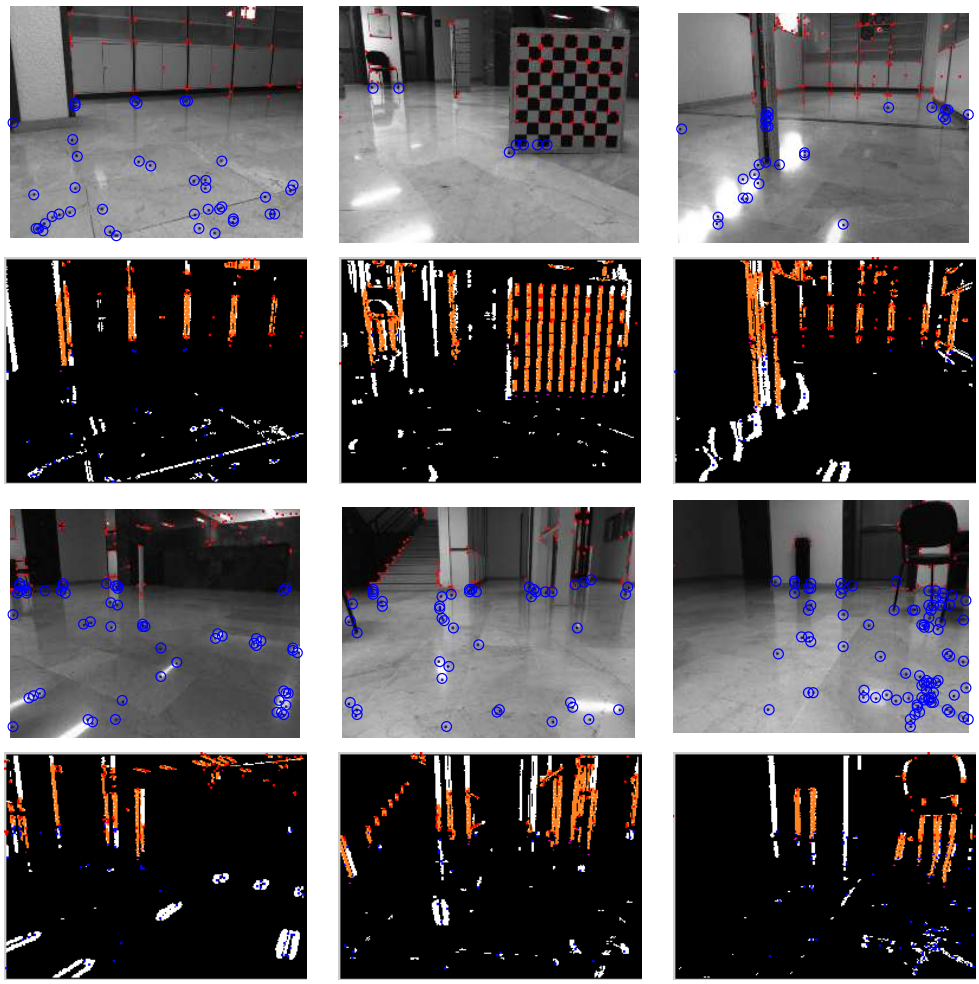


Figure 6.19: Experiments on Scenario 3: Images recorded online during the route.

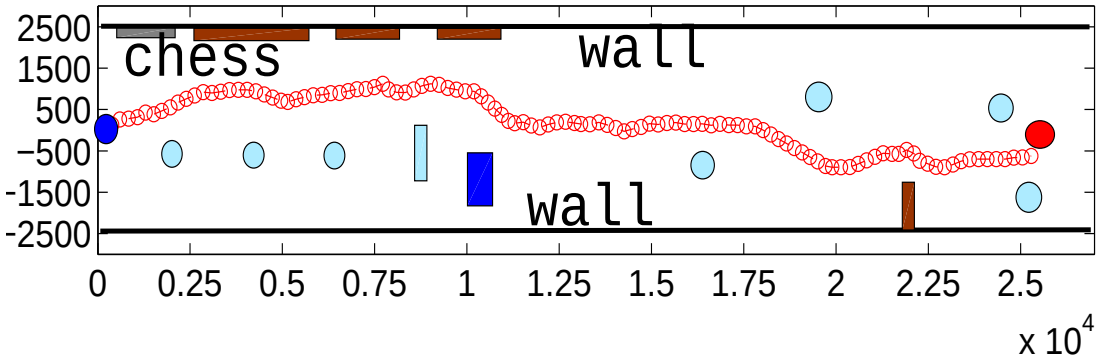


Figure 6.20: The longest trajectory along a corridor, with several obstacles placed along the path. All length units expressed in mm.

alternative to compute the robot position and thus to reduce the errors inherent to dead reckoning. Next section details the visual robocentric localization algorithm, designed to compute the robot position by fusing on an EKF the world coordinates of those features classified as ground points and the odometry information. Ground points have been chosen to be used in the localization process since their 2D position in the ground plane with respect to the robot is known with reasonable accuracy. The aim is to use part of the features gathered in the obstacle detection module to obtain a reliable robot pose data along the whole trajectory. Besides, the result of the continuous EKF execution will lead to the stabilization of the ground landmarks world coordinates.

CHAPTER 7

ROBOCENTRIC LOCALIZATION USING GROUND POINTS

7.1 Discrete Kalman Filters. EKF Localization

7.1.1 The Linear Kalman Filter

Kalman filters are original from Rudolph Emil Kalman and are techniques for recursively computing predictive data (so called *beliefs*) or state processing in linear evolutive systems. These filters are very powerful in the sense that they can support estimations of future states of a system, and they can be applied even when the nature of the system is unknown [135, 96, 198] .

Future system states are estimated from system models and sensor measurements. Data calculated at one execution is updated in the next consecutive execution. *Beliefs* at time t are the outputs of the filter, while the filter inputs are the beliefs at $t-1$. These inputs are then corrected or updated by the control vector u_t (the control vector typically includes robot pose or velocity information) and the environmental observations or measurements z_t.

The state vector, which generally contains the information to be predicted or updated, describes a process, discrete in time, characterized by a linear difference equation:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{7.1}$$

where x_t and x_{t-1} are the state vectors at time step t and $t-1$, respectively, u_t is the control vector at t, A_t is a square matrix of dimension $m \times m$ (being m the state vector dimension), B_t is a another matrix of dimension $m \times n$, (being n the dimension of u_t) and ϵ_t is an independent, additive, white, zero-mean Gaussian noise modeling the prediction uncertainty.

At discrete time intervals, the sensors gather state observations z_t, which in real robotic cases would be the sensors environmental readings. The sensor readings can be predicted from the state vector as:

$$z_t = H_t x_t + \delta_t \tag{7.2}$$

where H_t is a matrix of size $k \times m$ (being k the dimension of z_t), and δ_t indicates the measurement noise or the difference between observed and predicted data, again a zero-mean Gaussian distribution with covariance R_t.

Let us denote $\hat{x}_t$ as the mean of the state vector x_t at time step t, and P_t its covariance. The Kalman Filter estimation process runs in two steps, the prediction and the update steps.

The prediction step computes the estimated state vector by means of the state model characterized in the A_t and B_t matrices and the control vector:

$$\begin{aligned} \hat{x}_t^- &= A_t \hat{x}_{t-1} + B_t u_t, \\ P_t^- &= A_t P_{t-1} A_t^T + Q_t \end{aligned} \tag{7.3}$$

being $\hat{x}_t^-$ and P_t^- the mean and covariance, respectively, of the predicted state vector and Q_t the model noise covariance.

In the update step, these predictions are corrected using the observations given by the sensors. The Kalman Filter equations for this step are:

$$\begin{aligned} K &= P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \\ \hat{x}_t^+ &= \hat{x}_t^- + K\mu \\ \mu &= z_t - H_t(\hat{x}_t^-) \\ P_t^+ &= (I - K H_t) P_t^- \end{aligned} \tag{7.4}$$

being x_k^+ the state vector after the KF update, with mean $\hat{x}_t^+$ and covariance P_t^+ and

R_t representing the measurements uncertainty. μ can be seen as the discrepancy between the observations given by the sensors and the prediction of these observations given by the model as $H_t\hat{x}_t^-$. K is called the Kalman gain and it indicates the degree of relevance or participation of a certain measurement when it is incorporated in the estimation of the state vector.

7.1.2 The Extended Kalman Filter (EKF)

In real situations, finding systems that fulfill linearity affected by white zero-mean Gaussian noise is exceptional. Non-linear systems governed by non-linear functions can be managed with EKFs.

Now, the part of the linear equation 7.1 involving the A_t and B_t matrices is substituted by the nonlinear function g and the product $H_t x_t$ in equation 7.2 is substituted by the nonlinear function h:

$$\begin{aligned} x_t &= g(u_t, x_{t-1}) + \epsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned} \tag{7.5}$$

where h is called the observation function and it is dedicated to predict the measurements from the state vector, with a certain error δ.

The EKF computes an approximation of the true *belief* contrarily to the KF which computes an optimum estimation of the *beliefs* under the conditions of use. The key of the EKF is to linearize the functions g and h. Once these functions are linearized, the mechanics of this filter are the same as the ordinary KF. The EKF uses the Fist Order Taylor Expansion technique to linearize. The Taylor expansion gives a linear approximation of a function from the function values at some points and the partial derivatives at those function points.

The equations for the prediction step are:

$$\begin{aligned} \hat{x}_t^- &= g(\hat{x}_{t-1}, u_t), \\ P_t^- &= G_t P_{t-1} G_t^T + Q_t \end{aligned} \tag{7.6}$$

being G_t the Jacobian matrix used for linearization:

$$G_t = \frac{\partial g}{\partial x}(\hat{x}_t^-, u_t, 0)$$

and Q_t the noise covariance.

The equations for the update step are:

$$
\begin{aligned}
K &= P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \\
\hat{x}_t^+ &= \hat{x}_t^- + K\mu \\
\mu &= z_t - h_t(\hat{x}_t^-, 0) \\
P_t^+ &= (I - KH_t)P_t^-
\end{aligned}
\tag{7.7}
$$

being H_t the Jacobian needed to linearize the function h and R_t is again the measurements uncertainty:

$$
H_t = \frac{\partial h}{\partial x}(\hat{x}_t^-, 0)
$$

so called the observation matrix.

The main strength of this filter is the computational efficiency compared with other tools such as particle filters.

7.1.3 EKF Localization

As it has been previously mentioned in section 2.2.2, Extended Kalman Filters have been extensively used in the robotics context as data processing tool or state estimation, in applications such as visual localization, mapping or visual SLAM.

These solutions integrate the environmental information observed by the onboard sensors (usually landmark position) in the filter state together with control information such as the pose or the velocity of the robot.

In [198], Sebastian Thrun *et al* modeled the EKF-localization method for two different situations, namely the *localization with know correspondences* and the *localization with unknown correspondences*. In both cases, it was assumed that the map of the environment was represented by a set of uniquely identifiable features and that the EKF state vector was filled with the feature measurements and the estimated robot pose. The proposed model for localization with know correspondences additionally assumed that correspondences between features in consecutive observation instants were known and that the initial robot pose was also relatively well known.

The algorithm model requires in its input at time instant t: a) the Gaussian estimate of the robot pose, b) the control information, and c) the set of feature measurements at t, along with their correspondence information. The filter returns as output the estimates of the robot pose, in terms of its mean and covariance. Simulated theoretical results show as the uncertainty in the robot pose estimates

(represented by their error ellipses) increases if no landmarks are obseved. Either capturing new landmarks or visualizing old ones permits the system re-calculating and correcting the robot pose. This model algorithm can be applied in countless situations. The landmark measurements, the robot pose data, and the filter results will entirely depend on the sensorial robot equipment and the system model designed for each situation and environmental conditions.

The next section details the EKF-based localization algorithm implemented and integrated in the navigation approach presented in this thesis. It is an implementation of the well known problem of *localization with known correspondences*, and it uses the ground points world coordinates as the measurements of the observed natural landmarks and the robot odometry data as its pose estimation.

7.2 Visual Localization

Our proposal is to use the ground points that are obtained from the classification process, and not used until now, to perform localization. As all these points are located on the floor, their z coordinate is known to be zero and their local coordinates $[x, y]$ can be calculated from the corresponding image coordinates and the camera angles. In the context of this section, the ground points will be referred to as landmarks.

The subsequent ground point observations are fused with dead reckoning by means of an EKF (*Extended Kalman Filter*). The robot pose, as well as the landmarks themselves, are stored in the EKF state vector.

The EKF approach to localization involves the linearization of both sensor and motion models by means of a first-order Taylor series expansion. Due to the errors introduced by these linearizations, the filter ends up providing inconsistent pose estimates. Similarly to [39], these inconsistencies are particularly problematic in our localization proposal as ground points are removed from the filter when they are not observed. Thus, no loop closure is performed and the uncertainty with respect to a world-fixed reference frame always grows. As large uncertainties (i.e. covariances) are responsible for large EKF linearization errors, one of our goals is to reduce these uncertainties.

Our proposal is based on the robocentric approach [29]. Under this approach, ground points are stored and managed with respect to a coordinate frame locked to the robot. Thus, the ground points that are close to the robot will have lower

uncertainty and linearizations will be more valid. Our approach particularly benefits from this idea as only the visible ground points, which are close to the robot, take part in the localization process.

Next, the main steps involved in the robocentric localization algorithm are described.

7.2.1 Robocentric EKF prediction

The EKF prediction is in charge of estimating the state vector at time step k from the state vector at time step $k-1$ and the dead deckoning information. Let $x_k = N(\hat{x}_k, P_k)$ be the state vector at time k, being $\hat{x}_k$ its mean and P_k its covariance:

$$x_k = \begin{bmatrix} x_W^{R_k} \\ x_L^{R_k} \end{bmatrix} \tag{7.8}$$

where $x_W^{R_k} = N(\hat{x}_W^{R_k}, P_W^{R_k})$ denotes the pose of a world-fixed coordinate frame with respect to the robot and $x_L^{R_k} = N(\hat{x}_L^{R_k}, P_L^{R_k})$ represents the positions of the landmarks with respect to the robot. The term landmark refers to the aforementioned ground points. Note that, as this is a robocentric approach, every item in the state vector is represented with respect to the robot. Initially, the state vector will consist only of $x_W^{R_k}$, with zero mean and covariance. Let us denote by $x_{R_k}^{R_{k-1}} = N(\hat{x}_{R_k}^{R_{k-1}}, Q_k)$ the dead reckoning estimate of the robot motion from time step $k-1$ to time step k.

At this point, the next logical step would be to use the dead reckoning information to represent each landmark position, $x_{L_j}^{R_k}$, in the state vector with respect to the current robot pose so that the state vector remains robocentric. This could be easily done as $x_{L_j}^{R_k} = \ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}}$, where $\ominus$ and $\oplus$ are the inversion and compounding operators, widely used in stochastic mapping and SLAM [29]. However, as dead reckoning is likely to be the less precise component in the system, performing this transformation may introduce significant errors. Accordingly, the proposal in the robocentric approach is to delay this composition until the robot motion has been improved in the EKF update step. To this end, during the prediction step, the state vector comprise the state vector in the previous time step plus the dead reckoning estimate as an independent element:

$$\hat{x}_k^- = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{x}_{R_k}^{R_{k-1}} \end{bmatrix} \quad P_k^- = \begin{bmatrix} P_{k-1} & 0 \\ 0 & Q_k \end{bmatrix} \tag{7.9}$$

where $x_k^- = N(\hat{x}_k^-, P_k^-)$ denotes the predicted state vector.

7.2.2 Robocentric EKF update

By matching the new observations against the corresponding landmarks in the state vector, the filter performs the measurement update. As stated previously, the vision modules not only provide the landmark coordinates, but also their associations between consecutive frames and, thus, between the observed and the ones stored in the state vector. Thus, from the EKF point of view, the data association is assumed to be solved. Let us assume that at time step k a set of M observed ground points has been associated with M landmarks previously included in the state vector. Let z_k^i denote the $i-th$ ground point observed at time step k, which is known to be associated with the landmark $x_{L_j}^{R_{k-1}}$, stored in the state vector.

In the EKF context, the observation function $h_{j,k}^i$ is in charge of predicting the measurement z_k^i from the state vector. Accordingly, $h_{j,k}^i$, is:

$$h_{j,k}^i(\hat{x}_k^-) = \ominus \hat{x}_{R_k}^{R_{k-1}} \oplus \hat{x}_{L_j}^{R_{k-1}} \tag{7.10}$$

The observation matrix $H_{j,k}^i$ which is the Jacobian matrix of $h_{j,k}^i$, is the following:

$$H_{j,k}^i = \left.\frac{\partial h_{j,k}^i}{\partial x_k^-}\right|_{\hat{x}_k^-} = \left.\left[\frac{\partial h_{j,k}^i}{\partial x_W^{R_{k-1}}}, \frac{\partial h_{j,k}^i}{\partial x_{L1}^{R_{k-1}}} \cdots \frac{\partial h_{j,k}^i}{\partial x_{Lj}^{R_{k-1}}} \cdots \frac{\partial h_{j,k}^i}{\partial x_{LN}^{R_{k-1}}}, \frac{\partial h_{j,k}^i}{\partial x_{R_k}^{R_{k-1}}} \right]\right|_{\hat{x}_k^-} \tag{7.11}$$

It is easy to see that all the terms in the previous expression are zero except the two terms that depend on the landmark currently observed and on the dead reckoning estimate. The former can be obtained as follows:

$$\begin{aligned} \left.\frac{\partial h_{j,k}^i}{\partial x_{L_j}^{R_{k-1}}}\right|_{\hat{x}_k^-} &= \left.\frac{\partial(\ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}})}{\partial x_{L_j}^{R_{k-1}}}\right|_{\hat{x}_k^-} \\ &= J_{2\oplus}\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_{L_j}^{R_{k-1}}\} \end{aligned} \tag{7.12}$$

where $J_{2\oplus}$ is the second Jacobian matrix of the compounding operator, as described in [29]. The second of the mentioned non-zero terms is:

$$\begin{aligned}
\left.\frac{\partial h^i_{j,k}}{\partial x^{R_{k-1}}_{R_k}}\right|_{\hat{x}^-_k} &= \left.\frac{\partial(\ominus x^{R_{k-1}}_{R_k} \oplus x^{R_{k-1}}_{L_j})}{\partial x^{R_{k-1}}_{R_k}}\right|_{\hat{x}^-_k} \\
&= \left.\frac{\partial(\ominus x^{R_{k-1}}_{R_k} \oplus x^{R_{k-1}}_{L_j})}{\partial(\ominus x^{R_{k-1}}_{R_k})} \cdot \frac{\partial(\ominus x^{R_{k-1}}_{R_k})}{\partial x^{R_{k-1}}_{R_k}}\right|_{\hat{x}^-_k} \qquad (7.13) \\
&= J_{1\oplus}\{\ominus \hat{x}^{R_{k-1}}_{R_k}, \hat{x}^{R_{k-1}}_{L_j}\} J_{\ominus}\{\hat{x}^{R_{k-1}}_{R_k}\} \qquad (7.14)
\end{aligned}$$

where $J_{1\oplus}$ is the first Jacobian matrix of the compounding operator and $J_{\ominus}$ is the Jacobian matrix of the inversion, also described in [29].

The measurement vector z_k, observation function h_k and observation matrix H_k can be constructed using the M values from z^i_k, $h^i_{j,k}$, $H^i_{j,k}$ and C_k, where $C_k = \{< \alpha_1, \beta_1 >, < \alpha_2, \beta_2 >, \cdots, < \alpha_M, \beta_M >\}$ denotes the set of data associations so that the measurements $z^{\alpha_1}_k \cdots z^{\alpha_M}_k$ are associated with the landmarks $L_{\beta_1} \cdots L_{\beta_M}$ respectively:

$$z_k = \begin{bmatrix} z^{\alpha_1}_k \\ \vdots \\ z^{\alpha_M}_k \end{bmatrix}, \quad h_k = \begin{bmatrix} h^{\alpha_1}_{\beta_1,k} \\ \vdots \\ h^{\alpha_M}_{\beta_M,k} \end{bmatrix}, \quad H_k = \begin{bmatrix} H^{\alpha_1}_{\beta_1,k} \\ \vdots \\ H^{\alpha_M}_{\beta_M,k} \end{bmatrix}. \qquad (7.15)$$

The update step can be performed by applying the EKF update equations as follows:

$$\begin{aligned}
K &= P^-_k H^T_k (H_k P^-_k H^T_k + R_k)^{-1} \\
\hat{x}^+_k &= \hat{x}^-_k + K(z_k - h_k(\hat{x}^-_k)) \qquad (7.16) \\
P^+_k &= (I - KH_k) P^-_k
\end{aligned}$$

where R_k is a block diagonal matrix containing the error covariances associated to each measurement and $x^+_k = N(\hat{x}^+_k, P^+_k)$ denotes the state vector after the EKF update.

7.2.3 Robocentric composition and state augmentation

At this point, the robot motion $x^{R_{k-1}}_{R_k}$, which was included into the state vector in the prediction step, has been improved. Thus, this improved robot motion can now

be used to arrange the state vector so that it remains robocentric. This is the so called composition step. The resulting state vector is the following:

$$x_k = \begin{bmatrix} \ominus x_{R_k}^{R_{k-1}} \oplus x_W^{R_{k-1}} \\ \vdots \\ \ominus x_{R_k}^{R_{k-1}} \oplus x_{LN}^{R_{k-1}} \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_N \end{bmatrix} \tag{7.17}$$

where $x_W^{R_{k-1}}$ is the pose of the world-fixed coordinate frame with respect to the robot after the EKF update, all $x_{LN}^{R_{k-1}}$ correspond to the landmarks pose with respect to the robot also after the EKF update, and all the f_i have been introduced just to ease notation in further explanations. It is straightforward to derive the expressions for the mean of the state vector. In order to obtain the covariance of x_k, the following Jacobian matrix has to be computed:

$$J \doteq \left.\frac{\partial x_k}{\partial x_k^+}\right|_{\hat{x}_k^+} = \left.\begin{bmatrix} \frac{\partial f_0}{\partial x_W^{R_{k-1}}} & \cdots & \frac{\partial f_0}{\partial x_{LN}^{R_{k-1}}} \frac{\partial f_0}{\partial x_{R_k}^{R_{k-1}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_W^{R_{k-1}}} & \cdots & \frac{\partial f_N}{\partial x_{LN}^{R_{k-1}}} \frac{\partial f_N}{\partial x_{R_k}^{R_{k-1}}} \end{bmatrix}\right|_{\hat{x}_k^+} \tag{7.18}$$

The non-zero terms in this expression are computed similarly to Equations 7.12 and 7.14:

$$\left.\frac{\partial f_i}{\partial x_i^{R_{k-1}}}\right|_{\hat{x}_k^+} = J_{2\oplus}\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_i^{R_{k-1}}\} \tag{7.19}$$

$$\left.\frac{\partial f_i}{\partial x_{R_k}^{R_{k-1}}}\right|_{\hat{x}_k^+} = J_{1\oplus}\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_i^{R_{k-1}}\} J_{\ominus}\{\hat{x}_{R_k}^{R_{k-1}}\} \tag{7.20}$$

where $\hat{x}_i^{R_{k-1}}$ denotes $\hat{x}_W^{R_{k-1}}$ when $i = 0$ and $\hat{x}_{L_i}^{R_{k-1}}$ when $i > 0$. Using this Jacobian, the covariance is updated as $P_k = J \cdot P_k^+ \cdot J^T$.

Finally, the observed ground points that have no corresponding landmark in the state vector are considered to be new landmarks and included in the state vector. Because of the robocentric approach, adding new landmarks consists, simply, in including them in the state vector exactly as they are provided by the vision modules. Also at this point, the landmarks in the state vector that have not been observed during a certain time (i.e. 3 frames) are marginalized out and removed.

When needed, the robot pose with respect to the world-fixed coordinate frame can be recovered easily from the state vector as $x_{R_k}^W = \ominus x_W^{R_k}$.

7.3 Experimental Results

Concerning localization, the same robot Pioneer 3DX was used in the same scenarios as for the navigation experiments, and with exactly the same operative conditions, in terms of focal distance, frame rate, camera intrinsic parameters, camera height and tilt. All the length units shown in the incoming experiments are expressed in meters.

The camera motion between consecutive frames was obtained from the robot odometry and the relative position of the camera with respect to the robot center.

In the implementation, ground features are continuously tracked while they stay in the camera field of view. When a ground feature has not been seen during 3 frames, it is removed from the state vector, and when a new feature is fetched it is included in the filter. The system always tries to ensure a minimum number of observed landmarks in the state vector.

Several experiments were conducted both indoors and outdoors. Indoor experiments have been performed both inside the laboratory and throughout corridors in a university building, making the robot follow several different trajectories. The outdoor experiments were performed in the university campus. In these environments, the rough terrain was responsible for bad odometry estimates.

Finding a proper ground truth became necessary to quantitatively evaluate the obtained pose estimates. For the experiments performed inside the laboratory, the ground truth was calculated from the images captured by two calibrated, world-fixed, wide angle cameras facing perpendicularly to the ground plane at a height of 3 meters (see figure 7.1). The experiments were recorded using these two cameras and the ground truth trajectory was recovered from the gathered images.

A different approach was adopted for outdoor scenarios since no cameras were available to generate a ground truth. In these cases, the Pioneer 3-DX ultrasonic array was activated and the data it provided were recorded. The ultrasonic data correspond to distances of detected objects with respect to the robot. Then, the ultrasonic data was plotted according to the trajectory being evaluated. When plotted, sonar data must coincide with the reality if the trajectory of the robot is accurately calculated. If the robot trajectory presents drifts, the plotted sonar

Figure 7.1: World-fixed cameras for ground truth calculation, on top of the picture

data will significantly differ from the real surrounding environment. Consequently, the better the plotted sonar data coincides with the reality, the better is the robot trajectory used to plot it. Thus, although no ground truth is available in outdoor tests, sonar data makes it possible to qualitatively evaluate trajectories by visual inspection.

For all and each of the trajectories run inside the lab, five different levels of synthetic noise (namely, from now on, experiments 1 to 5) were added to the odometry data, in order to check our approach with different levels of dead reckoning errors, with lower and higher affectation. Especially relevant would be results of the filter estimates in front of severe odometric errors. The odometric noise was additive, zero-mean Gaussian with covariance $\Sigma = diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2)$, where σ_x and σ_y ranged from 0,0032 meters in the first experiment to 0,032 meters in the fifth experiment, and σ_θ ranged from 0,0032 radians in the first experiment to 0,032 radians in the fifth. These σ ranges correspond to σ^2 between approximately 0.00001 and 0.001, respectively. These values were chosen to represent, respectively, noise with low and high covariance, that is, systems under a low noise influence and systems highly affected by noise. For each one of these tests, with a certain trajectory and a certain noise level, 100 runs were executed in order to obtain statistically significative results. In all tests and for each robot pose estimate, the trajectory error was computed, both for dead reckoning and for our approach. The trajectory error is defined as the distance between the robot pose under evaluation and its corresponding ground truth.

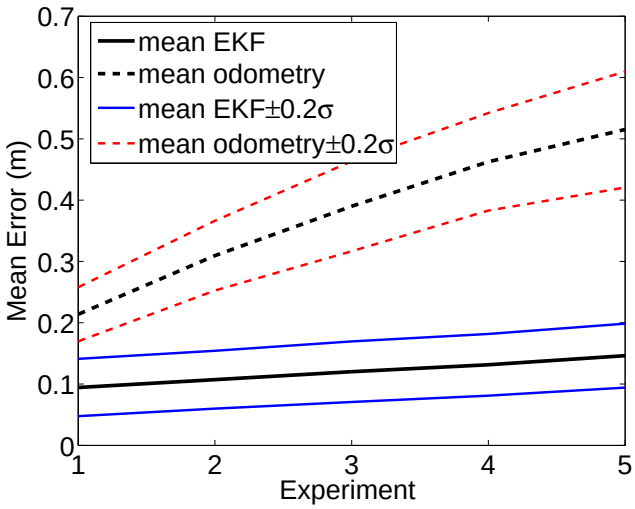
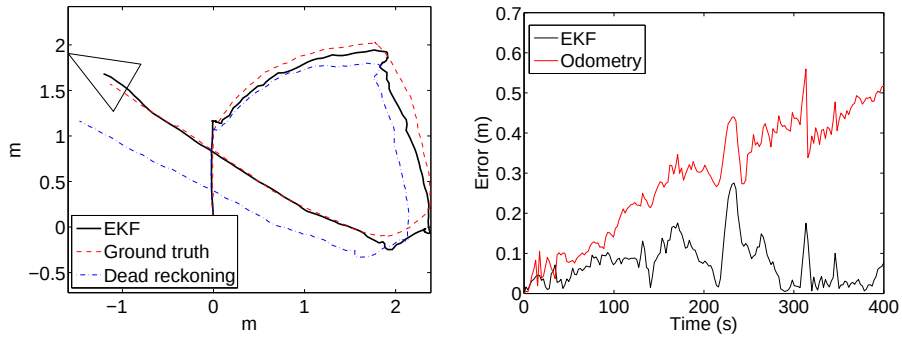


Figure 7.2: Mean and Mean$\pm 0.2\sigma$ for the trajectory errors.

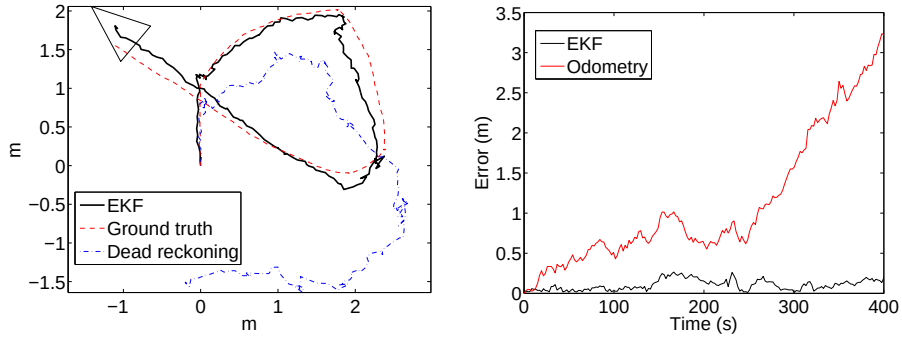
Figure 7.2 shows a comparison of the mean errors corresponding to the corrupted odometry with those corresponding to our approach, as a function of the 5 different noise levels and computed for all the trajectories. As can be observed, the mean error when using our approach, labeled as EKF, is considerably below the dead reckoning error, and so does the standard deviation of the error. This correction is much more evident as the odometry noise grows. In the figure, σ has been scaled down to 20% to provide a more clear plot.

Figure 7.3 shows some data corresponding to two of the trajectories conducted in the laboratory and used to perform the plot of figure 7.2. The plots show the dead reckoning trajectories corrupted with two different levels of noise, the corrected EKF trajectories and the evolution of the corresponding trajectory errors with time. These examples evidence that dead-reckoning trajectories with higher noise levels (with higher σ) present more distortion with respect to the ground truth than dead-reckoning routes slightly affected by noise. However, the proposed EKF localization algorithm corrected the odometry data even in the presence of high noise levels. Figure 7.4 shows some images captured from the robot during these trajectories. All images show the KLT features classified as obstacle points in red and those classified as ground points as circles in blue.

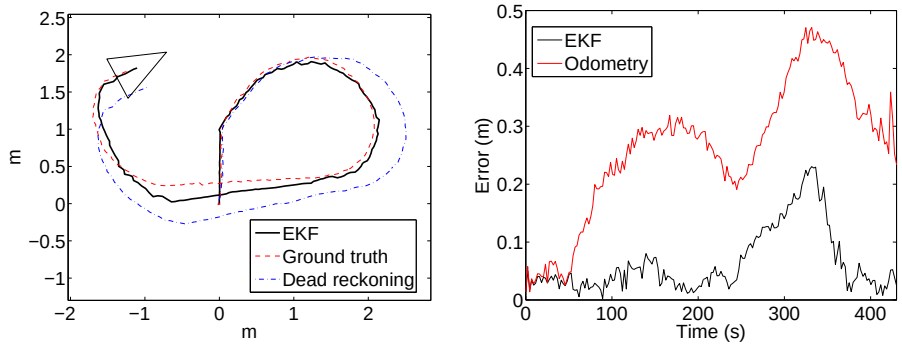
Figure 7.5 shows images and the plot of a rectilinear trajectory conducted along the center of a 2.50 meters wide corridor with a high textured floor. This is one of the aforementioned indoor scenarios where no ground truth was available. Wheels



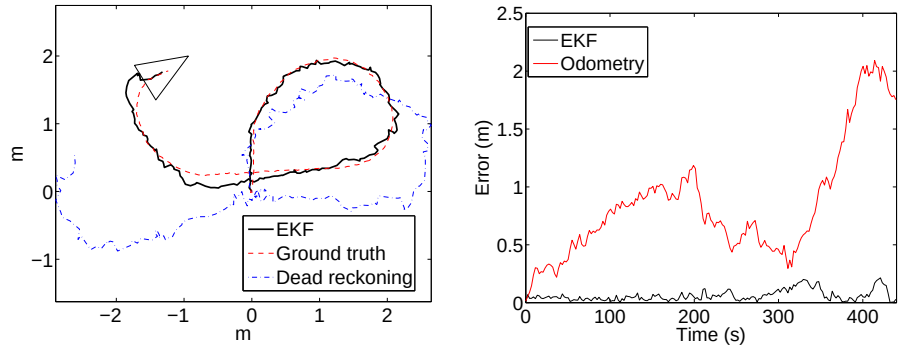
Trajectory 1: (a) (b)



Trajectory 1: (c) (d)



Trajectory 2: (e) (f)



Trajectory 2: (g) (h)

Figure 7.3: (a), (e) Two different trajectories with σ_x=σ_y=0.0055m for the odometry Gaussian noise. (c), (g) Trajectories with σ_x=σ_y=0.032m for the odometry Gaussian noise. (b), (d), (f) and (h) Trajectory errors.

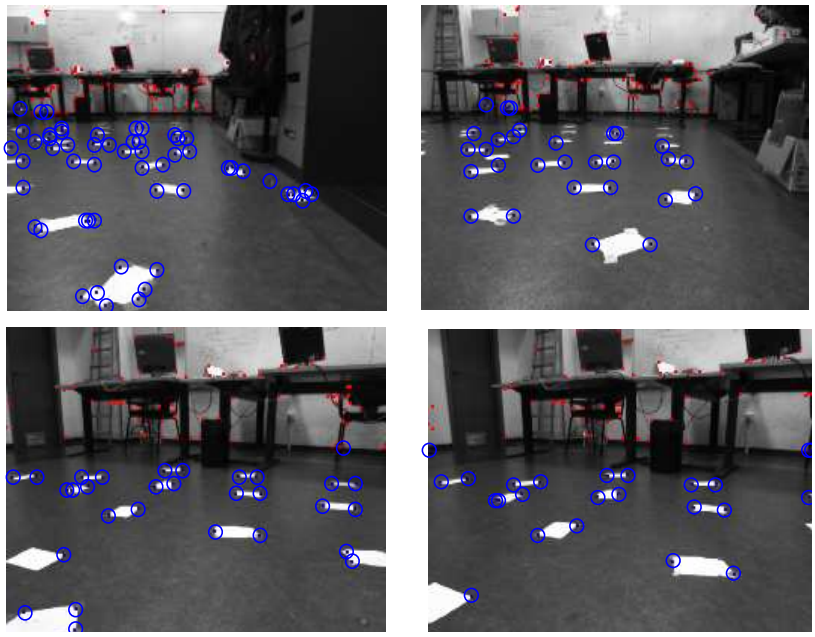


Figure 7.4: Images taken from the robot, during the trajectories 1 and 2 referenced in figure 7.3 .

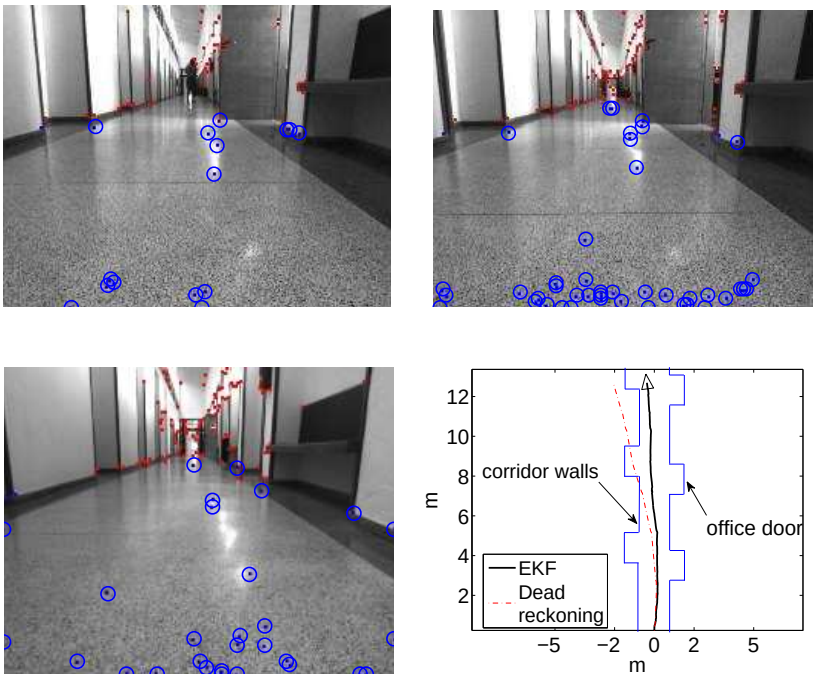


Figure 7.5: Images and trajectory on a straight corridor.

were slightly deflated, resulting in odometric error, but no synthetic noise was added in this case. Notice how odometry trajectory significantly deviates from the forward direction whilst the EKF trajectory approximates better the right ahead course.

Figure 7.6 shows some data concerning two different outdoor experiments. Both tests consisted on moving the robot parallel to the walls of different buildings. It is important to emphasize that the proximity of walls is only needed to collect the aforementioned sonar data, which is only used for human inspection. In these experiments, stones, grass, and small clumps of soil over the floor influenced the robot odometry estimates.

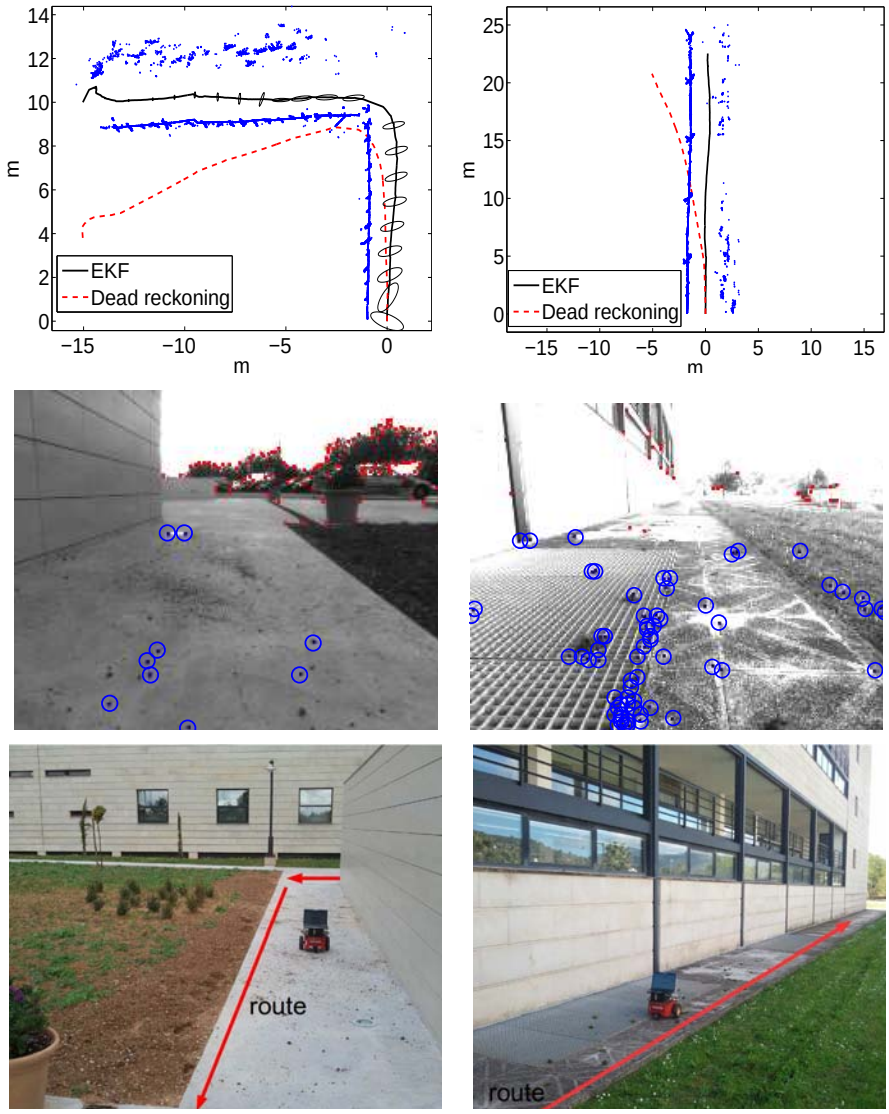


Figure 7.6: Trajectories and some images of outdoor experiments.

The top row shows, for both experiments, the EKF trajectory in black with the 2σ bound uncertainty ellipses in the picture on the left, the odometry trajectory in red and the sonar data with respect to every EKF robot position in blue. The middle row shows two images captured from the robot showing the KLT obstacle points in red and the features classified as ground circled in blue. The bottom row shows two images of the whole scene during the test. Notice how in outdoor environments, even if only a reduced number of landmarks is detected and tracked, the pose estimation is good.

Although no ground truth was available in these experiments, the plotted sonar data gives a clear idea regarding the correctness of the estimated trajectories. As it can be observed in the bottom row pictures, the building walls and angles are straight, and so it is reflected by the sonar readings, which are plotted using the robocentric pose estimates.

The exposed experiments evidence the potential of the presented localization approach. It gives to the navigation system exposed in this thesis the possibility of using part of the image features for improving the robot pose provided by dead-reckoning.

Part III

Conclusions

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

Navigation and localization abilities are essential in reactive autonomous mobile robots. The capacity of the robot to decide which movements are adequate in each different situation and at each moment is a basic feature to successfully accomplish the programmed missions. For reactive autonomous robots, it is also capital to be equipped with a precise localization algorithm, since accurate position is also a fundamental data to navigate along all subsequent targets.

This thesis has presented a novel visual obstacle detection and avoidance algorithm integrated in a new reactive navigation task, which both experimentally showed to be effective in autonomous missions, conducted in daily scenarios densely occupied by obstacles. Furthermore, the navigation system is complemented with a localization algorithm which integrates ground points and the dead-reckoning information in the state vector of an EKF to correct the pose given by the odometers. Experimental results of this localization process demonstrated a great level of success in the odometry data correction.

The main contributions of this work have already been detailed in section 3.5, however it is worth to review all of them again in this section since each one generates interesting conclusions and additional work. Summarizing:

1. An extensive survey of visual navigation from the early nineties up to nowadays,

2. the classifier of image features between obstacle and ground points,

3. the obstacle detection and avoidance algorithm,

4. an evaluation of different feature detectors and descriptors for our particular obstacle detector,

5. the reactive navigation strategy to safely move from one point to another, and

6. the localization method.

8.1 The Image Point Classifier

The new algorithm based on the IPT used to classify image main features in either obstacle or ground points presented an important degree of success, according to the evaluation results obtained applying ROC curves and their AUC. The method has been tested offline and online, indoors and outdoors giving in all cases promising results.

The point classifier assumes that the ground is plane, with no irregularities or changes in slope. This assumption is important for the classifier and its lack of compliance could mean a worse performance in the classification process. This ground-flat assumption has been extensively applied in robot visual navigation approaches based on feature tracking, homographies computation, or any other solution based on geometrical considerations. Relaxing the flat ground constraint is under study and would extend the scope of application of our system to terrains with certain irregularities.

The entire feature classification process is also very dependent on the discrepancy D and the previously defined threshold β. This dependability was minimized as explained in section 4.3.1, but still that empirical method based on histograming misclassified points could be changed by a general and theoretical method. Applying evolutive algorithms to retrieve the optimum β and those $D's$ that generated misclassified points is under consideration. This process would be previously run in order to anticipate these parameters before the autonomous navigation mission begins.

8.2 Obstacle Detection and Avoidance: The Local Occupancy Map

Image features classified as obstacle points are used to discriminate the boundaries of obstacles present in the scene from other edges located on the ground. This process permits computing the world coordinates of all these points where obstacles touch the ground. These obstacle-ground contact points are drawn in an occupancy map that qualitatively represents the immediate vicinity of the robot, giving a certain idea of which frontal areas are occupied and which ones are free. This method does not depend on different scene planes, homographies, or ground/obstacle textures. It also works, to a certain extend, under deficient illumination conditions, with reflexions and with specularities. The aim is not building an accurate composition of the real world, with extremely detailed measurements, but qualitatively seeing which areas are occupied and which are free. Experimental results, where a robot erratically moved avoiding all obstacles, showed how local maps were simple but effective in marking the areas occupied by obstacles.

The obstacle detection and localization processes described in this thesis are extremely dependent on the previous task of image feature detection and tracking. Each feature detector and descriptor gives different results in terms of the number of detected points, their location on the image and their robustness on the tracking process. The affectation of these parameters can be summarized as follows: a) very few image features or scarcely detected can lead to miss some obstacles or can produce bad results in the localization EKF process, b) a leak of robustness in the matching/tracking process, caused by, for example, changes on scale, rotations, changes on illumination or excessive differences on their descriptors, can generate an excess of outliers, being rejected from the process and leading to the previous situation of few points, c) correct matchings but between points with considerable differences in position or in their descriptors can difficult their classification and the determination of the threshold β (see section 4.3.1), d) in principle, it seems that the location of features in the image should not be a big deal, but, still, in order to guarantee a good performance in obstacle avoidance and localization it is important to gather features as in obstacle boundaries as in the part of the image that shows the ground, and e) the execution time is important, since slow solutions can be discarded, particularly in online applications.

Not all feature detectors and descriptors provide the same data, and in some

cases these data lead to inadequate results in our navigation algorithm. As a matter of fact, section 5 shows how different feature detectors give different results in the classification process and in the subsequent obstacle detection algorithm. Among all tested algorithms (SIFT, SURF, FAST and KLT), SIFT is the most robust detector in front of all unfavorable conditions, but it is too slow. Conversely, KLT has some inadequacies, for example, in frame rotations or scale changes, but it provides the best results as: a) it finds a considerably number of features in many sectors of the image, and specially over obstacle edges, b) at relatively high frame rates and with no variations of the camera position with respect to the robot center and the gimbal, changes on scale are imperceptible and there are no rotations of frames around their center; consequently, the KLT tracker generates the highest rate of inliers, c) the tracking process is robust, and all detected points are correctly classified and, d) it is the fastest.

8.3 The Navigation Strategy

It is important for autonomous robots to be endowed with an effective obstacle detector. But further from detecting obstacles, there is also an important and clear need of moving towards or between points that have been preprogrammed as different stages of a complete mission. These succesive set of goal points have to be reached in the most effective way without excessive deviations due to the presence of obstacles. To this end, navigation strategies are important because they define how to act in each different situation or scene configuration, to effectively combine the avoidance of obstacles and the guidance to the desired targets. The navigation strategy presented in this thesis is based on Bug-T^2 [5] and ND [136], and integrates the obstacle detector detailed in section 4.3. From Bug-T^2 our proposal inherits the two basic principles of Tenacity and Traversability. On the one hand, the ability to complete a highly credible occupancy map, build with a sparse set of obstacle-ground contact points and used to check the traversability of the terrain. On the other hand, the concept of tenacity to overcome obstacles and local minimas. From ND our system acquires: 1) the capacity of detecting gaps or discontinuities between obstacles in scenarios densely occupied, and 2) the reasoning methodology to plan each different situation with a certain rule to pass through the gaps towards the goal points.

The combination of the two strategies in a visual solution permits complementing

the deficiencies of one with the benefits of the other. Success in real missions between two or more pre-defined points in daily and normally used indoor scenarios show the utility of the navigation strategy defined in this thesis. However, evolving this version of the algorithm towards one that could guarantee convergence would permit to face all type of obstacles no matter how intricate they were.

8.4 The Robocentric Localization Algorithm

The same camera and the same gathered data are used for navigation and for localization. Errors and drifts in odometric pose estimates are corrected including all features classified as ground points in an EKF context. This EKF-based localization process is intended to complement the navigation strategy, supplying the system with an extra ability without a reduction of effectiveness and without an extra cost in terms of either additional equipment, time, or a rise in the software complexity.

Besides, the robocentric approach guarantees nearly eliminating errors in the pose estimates inherent to the EKF intrinsic linearizations.

The ground points inserted in the filter state vector are removed or unused when they fall from the camera FOV. Although close loops can not be performed as in SLAM solutions, the localization algorithm runs in a short bounded time, which is a certain advantage for online applications.

The results of some of the indoor experiments could be contrasted with a real ground truth, and, the performance of the outdoor experiments could be estimated using the sonar data provided by the robot ultrasound sensors. Experiments run indoors and outdoors reveal a sufficiently precise performance of the localization technique.

Changing the EKF for an UKF (*Unscented Kalman Filter*) or a particle filter is under study to evaluate the localization performance given by these other tools and to compare them with the results given by the EKF.

CHAPTER 9

RELATED PUBLICATIONS

The research work presented in this Thesis has given rise to the following publications:

1. Visual Navigation for Mobile Robots: a Survey. Journal of Intelligent and Robotic Systems, vol. 53, n_{br} 3, pp. 263-296, 2008.

2. A Novel Image Feature Classifier based on Inverse Perspective Transformation, Technical Report A-1-2008, 2008.

3. A novel Vision-based Reactive Navigation Strategy Based on Inverse Perspective Transformation, Proceedings of IEEE-IFAC International Conference on Informatics in Control, Automation and Robotics (ICINCO), Milan (Italy), pp. 141-146, 2009.

4. A Novel Inverse Perspective Transformation-based Reactive Navigation Strategy, Proceedings of European Conference on Mobile Robots (ECMR), Dubrovnik (Croatia), pp. 25-30, 2009.

5. Building a Qualitative Local Occupancy Grid in a new Vision-based Reactive Navigation Strategy, Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) ref. 004847, 2009.

6. A Visual Navigation Strategy Based on Inverse Perspective Transformation. Book: Robot Vision, chapter 5. Intech-Sciyo. ISBN 978-953-307-077-3 2010

7. Experimental Assessment of Different Feature Tracking Strategies for an IPT-based Navigation Task. Proceedings of the IFAC International Conference on Autonomous Vehicles (IAV), september 2010.

8. Towards Monocular Localization Using Ground Points. Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) , september 2010.

9. Visual Localization Using Ground Points. Artificial Intelligencce Research and Development. IOS Press, Frontiers in Artificial Intelligence Series. October 2010. Pg 301-310.

10. Combining Obstacle Avoidance with Robocentric Localization in a Reactive Visual Navigation Task. Proceedings of the IEEE International Conference on Industrial Technology (ICIT), March 2012, (in press)

11. A Monocular Mobile Robot Reactive Navigation Approach Based on the Inverse Perspective Transformation, Robotica, Cambridge University Press, (accepted with minor revisions).

12. Concurrent Visual Navigation and Localization using the Inverse Perspective Transformation. IET Electronic Letters, vol 48, n_{br} 5, pp 264-266, 2012.

GLOSSARY

D Discrepancy between backprojections of a same image point captured in two consecutive frames. 81

S_{dist} Security Distance. 122

T^2 Traversability and Tenacity. 110

AUC Area Under the ROC Curve. 80

EKF Extended Kalman Filter. 67

FOV Field of View. 119

IPT Inverse Perspective Transformation. 69

KLT Kanade-Lucas and Tomassi. 47

ND Nearness Diagram. 110

RANSAC Random Sample Consensus. 76

ROC Receiver Operating Characteristic. 69

ROI Region of Interest. 86

SIFT Scale Invariant Feature Transform. 47

SSD Sum of Squared Differences. 98

VFH Vector Field Histogram. 69

BIBLIOGRAPHY

[1] J. Abascal, E. Lazkano, and B. Sierra. Behavior-Based Indoor Navigation. *Ambient Intelligence for Scientific Discovery. Foundations, Theories, and Systems*, LNAI(3345):263–285, 2005.

[2] M. Agrawal, K. Konolige, and M.R. Blas. CensurE: Center Surround Extremas or Realtime Feature Detection and Matching. *Lecture Notes in Computer Science*, 5305(3):102–115, 2008.

[3] A. Angeli, S. Doncieux, J.A. Meyer, and D. Filliat. Incremental Vision-based Topological SLAM. In *Proc. of IROS*, pages 1031–1036, 2008.

[4] J. Antich and A. Ortiz. Development of the Control Architecture of a Vision-guided Underwater Cable Tracker. *International Journal of Intelligent Systems*, 20(5):477–498, 2005.

[5] J. Antich and A. Ortiz. Bug-based t2: A new globally convergent potential field approach to obstacle avoidance. In *Proceedings of 2nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[6] J. Antich, A. Ortiz, and G. Oliver. A Control Strategy for Fast Obstacle Avoidance in Troublesome Scenarios: Application in Underwater Cable Tracking. In *Proc. of 7th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*, 2006.

[7] R.C. Arkin. *Behaviour Based Robotics*. The MIT Press, 1998.

[8] S. Atiya and G.D. Hager. Real Time Vision-Based Robot Localization. *IEEE Transactions on Robotics and Automation*, 9(6):785–800, 1993.

[9] S. Badal, S. Ravela, B. Draper, and A. Hanson. A Practical Obstacle Detection and Avoidance System. In *Proc. of 2nd IEEE Workshop on Applications of Computer Vision*, pages 97–104, 1994.

[10] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[11] B. Balasuriya and T. Ura. Underwater Cable Following by Twin-Burger 2. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2001.

[12] P.H. Batavia, D.A. Pomerleau, and C. E. Thorpe. Overtaking vehicle detection using implicit optical flow. In *IEEE Conference on Intelligent Transportation System*, pages 729–734, Boston, MA, USA, November 1997.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110:346–359, 2008.

[14] N. Bellotto and H. Hu. Multisensor Integration for Human-Robot Interaction. *Journal of Intelligent Cybernetic Systems*, 1, 2005.

[15] A. Bernardino and J. Santos-Victor. Visual Behaviours for Binocular Tracking. *Robotics and Autonomous Systems*, 25(3-4):137–146, 1998.

[16] M. Bertozzi and A. Broggi. Gold: a parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81, 1998.

[17] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, 1997.

[18] F. Bonin, A. Ortiz, and G. Oliver. Visual Navigation for Mobile Robots: a Survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

[19] J. Borenstein and I. Koren. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *Journal of Robotics and Automation*, 7(3):278–288, 1991.

[20] J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[21] J. Borenstein and Y. Koren. Real Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 572–577, 1990.

[22] J. Borenstein and Y. Koren. The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[23] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker, 2000.

[24] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical roc curves. *Computer Vision and Image Understanding*, 84(1):77–103, 2001.

[25] C. Braillon, K. Usher, C. Pradalier, J.L. Crowley, and C. Laugier. Fusion of Stereo and Optical Flow Data Using Occupancy Grid. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 2302–2307, 2006.

[26] A. Burguera, Y. González, and G. Oliver. On the Use of Likelihood Fields to Perform Sonar Scan Matching Localization. *Springer Autonomous Robots*, 2009.

[27] T. Camus, D. Coombs, M. Herman, and T.H. Hong. Real-Time Single-Workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence. In *Proc. of 13th International Conference on Pattern Recognition (ICPR). Applications and Robotic Systems*, 1996.

[28] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679 – 698, 1986.

[29] J.A. Castellanos, R. Martínez-Cantín, J.D. Tardós, and J. Neira. Robocentric Map Joining: Improving the Consistency of EKF-SLAM. *Robotics and Autonomous Systems*, 2007.

[30] K. Çelik, S. Chung, M. Clausman, and A.K. Somani. Monocular Vision SLAM for Indoor Aerial Vehicles. In *Proc. of IROS*, pages 1566–1573, 2009.

[31] E. Celaya, J.L. Albarral, P. Jimenez, and C. Torras. Visually-Guided Robot Navigation: From Artificial to Natural Landmarks. In *Proc. of International Conference on Field and Service Robotics*, pages 287–296, 2007.

[32] M.T. Chao, T. Braunl, and A. Zaknich. Visually-Guided Obstacle Avoidance. In *Proc. of 6th Int'l Conf. on Neural Information Processing ICONIP*, volume 2, pages 650–655, 1999.

[33] R. Chatila and J.P. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 138–145, 1985.

[34] Y. Cheng, M. Maimone, and L. Matthies. Visual Odometry on the Mars Exploration Rovers. In *Proc. of Int'l Conference on Systems, Man and Cybernetics*, volume 1, pages 903–910, 2005.

[35] A. Cherian, V. Morellas, and N. Papanikolopoulos. Accurate 3D Ground Plane Estimation from a Single Image. In *Proc. of ICRA*, pages 2243–2249, 2009.

[36] Y.H. Choi and S.Y. Oh. Visual Sonar Based Localization Using Particle attraction and Scattering. In *Proc. of IEEE International Conference on Mechatronics and Automation*, pages 449–454, July 2005.

[37] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, , and S. Thrun. *Principles of Robot Motion*. The MIT Press, 2005.

[38] H.I. Christensen, N.O. Kirkeby, S.Kristensen, and L.Knudsen. Model-Driven Vision for Indoor Navigation. *Robotics and Autonomous Systems*, 12:199–207, 1994.

[39] J. Civera, O.G. Grasa, A.J. Davison, and J.M.M. Montiel. 1-Point RANSAC for EKF-Based Structure from Motion. In *Proc. of IEEE IROS*, 2009.

[40] B. Cohen and J. Byrne. Inertial Aided SIFT for Time to Collision Estimation. In *Proc. of ICRA*, pages 1613–1614, 2009.

[41] T. Cornall and G. Egan. Optic Flow Methods Applied to Unmanned air Vehicles. *Academic Research Forum, Department of Electrical and Computer Systems Engineering, Monash University*, 2003.

[42] F.R. Dalgleish, S.W. Tetlow, and R.L. Allwood. Hammerhead: an AUV with an Integral Laser Imaging Sensor. *Oceanology*, 2004.

[43] F.R. Dalgleish, S.W. Tetlow, and R.L. Allwood. Vision-Based Navigation of Unmanned Underwater Vehicles: a Survey. Part I: Vision Based Cable-, Pipeline- and Fish Tracking. *Proc. of the Institute of Marine Engineering, Science and Technology. Part B, Journal of Marine Design and Operations*, B(7):51–56, 2004.

[44] F.R. Dalgleish, S.W. Tetlow, and R.L. Allwood. Vision-Based Navigation of Unmanned Underwater Vehicles: a Survey. Part II: Vision Based Station Keeping and Positioning. *Proc. of the Institute of Marine Engineering, Science and Technology. Part B, Journal of Marine Design and Operations*, B(8):13–19, 2004.

[45] N.X. Dao, B. You, and S. Oh. Visual Navigation for Indoor Mobile Robots Using a Single Camera. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 1992–1997, 2005.

[46] A.J. Davison. Real Time Simultaneous Localisation and Mapping with a Single Camera. *Proc. of International Conference on Computer Vision (ICCV)*, 2003.

[47] A.J. Davison, Y. González, and N. Kita. Real-Time 3D SLAM with Wide-Angle Vision. In *Proc. of IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[48] A.J. Davison and N. Kita. Sequential Localization and Map Building for Real Time Computer Vision and Robotics. *Robotics and Autonomous Systems*, 36(4):171–183, 2001.

[49] G.N. DeSouza and A.C. Kak. Vision for Mobile Robot Navigation : A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[50] A. Dev, B. Krse, and F. Groen. Navigation of a Mobile Robot on the Temporal Development of the Optic Flow. In *Proc. of IEEE Int'l Conf. of Intelligent Robots and Systems (IROS)*, pages 558–563, 1997.

[51] A. Diosi and L. Kleeman. Fast Laser Scan Matching Using Polar Coordinate. *The International Journal of Robotics Research*, 2007.

[52] R.O. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons Publisher, USA, 1973.

[53] G. Dudek, M. Jenkin, C. Prahacs, A. Hogue, J. Sattar, P. Giguère, A. German, H. Liu, S. Saunderson, A. Ripsman, S. Simhon, L.A. Torres-Mendez, E. Milios, P. Zhang, and I. Rekleitis. A Visually Guided Swimming Robot. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[54] J.W. Durham and F. Bullo. Smooth Nearness-Diagram Navigation. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[55] Y. Fan and B. Balasuriya. Optical Flow Based Speed Estimation in AUV Target Tracking. *Proc. of IEEE Oceans*, 2001.

[56] J. Fasola, P.E. Rybski, and M. Veloso. Fast Goal Navigation with Obstacle Avoidance Using a Dynamic Local Visual Model. In *Proc. of SBAI'05, The VII Brazilian Symposium of Artificial Intelligence*, 2005.

[57] J. Fasola and M. Veloso. Real-Time Object Detection using Segmented and Grayscale Images. In *Proc. of IEEE Int'l Conf. on Robotics and Automations (ICRA)*, pages 4088–4090, 2006.

[58] F.Dellaert, D.Fox, W.Burgard, and S.Thrun. Monte carlo localization for mobile robots. In *Proc. of IEEE Int'l Conference on Robotics and Automation (ICRA)*, pages 1322–1328, 1999.

[59] J. Ferruz and A. Ollero. Real-time Feature Matching in Image Sequences for Non-structured Environments. Applications to Vehicle Guidance. *Journal of Intelligent and Robotic Systems*, 28:85–123, 2000.

[60] S.D. Fleischer, R.L. Marks, and S.M. Rock. Improving Real-Time Video Mosaicing of the Ocean Floor. In *Proc. of IEEE Oceans*, 1995.

[61] G.L. Foresti and S. Gentili. A Hierarchical Classification System for Object Recognition in Underwater Environments. *IEEE Journal of Oceanic Engineering*, 1(27):66–78, 2002.

[62] R. Garcia, X. Cufi, and Ll. Pacheco. Image Mosaicking for Estimating the Motion of an Underwater Vehicle. In *Proc. of 5th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*, 2000.

[63] R. Gartshore, A. Aguado, and C. Galambos. Incremental Map Buildig Using an Occupancy Grid for an Autonomous Monocular Robot. In *Proc. of Seventh International Conference on Control, Automation, Robotics and Vision ICARCV*, pages 613–618, 2002.

[64] R. Gartshore and P. Palmer. Exploration of an Unknown 2D Environment Using a View Improvement Strategy. *Towards Autonomous Robotic Systems*, pages 57–64, 2002.

[65] R. Gartshore, P. Palmer, and J. Illingworth. A Novel Exploration Algorithm Based on a Improvement Strategy. *International Journal of Advanced Robotic Systems*, 2(4):287–294, 2005.

[66] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-Based Navigation and Environmental Representations with an Omni-directional Camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000.

[67] C. Georgiades, A. German, A. Hogue, H. Liu, C. Prahacs, A. Ripsman, R. Sim, L.A. Torres, P. Zhang, M. Buehler, G. Dudek, M. Jenkin, and E. Milios. An aquatic

Walking Robot. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[68] S.B. Goldberg, M. W. Maimone, and L. Matthies. Stereo Vision and Rover Navigation Software for Planetary Exploration. In *Proc. of IEEE Aerospace conference Proceedings*, pages 5–2025,5–2036, 2002.

[69] N. Gracias and J. Santos-Victor. Underwater Video Mosaics as Visual Navigation Maps. *Computer Vision and Image Understanding*, 1(79):66–91, 2000.

[70] V. Graefe. Driveless Highway Vehicles. In *Proc. of Int'l Hi Tech Forum*, pages 86–95, 1992.

[71] V. Graefe. Vision for Autonomous Mobile Robots. In *Proc. of IEEE Workshop on Advanced Motion Control*, pages 57–64, 1992.

[72] V. Graefe. Vision for Intelligent Road Vehicles. In *Proc. of the IEEE Symposium of Intelligent Vehicles*, pages 135–140, 1993.

[73] V. Graefe and K.Kuhnert. Vision-Based Autonomous Road Vehicles. *Vision Based Vehicle Guidance, Springer-Verlag New York*, pages 1–29, 1992.

[74] A. Grau, J. Climent, and J. Aranda. Real-time Architecture for Cable Tracking Using Texture Descriptors. In *Proc. of IEEE Oceans Conference*, volume 3, pages 1496–1500, 1998.

[75] W.E. Green, PY. Oh, and G.L. Barrows. Flying Insects Inspired Vision for Autonomous Aerial Robot Maneuvers in Near-earth Environments. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 2347–2352, 2004.

[76] W.E. Green, PY. Oh, K. Sevcik, and G.L. Barrows. Autonomous Landing for Indoor Flying Robots Using Optic Flow. In *Proc. of IMECE International Mechanical Engineering Congress*, volume 2, pages 1341–1346, 2003.

[77] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive Navigation in Outdoor Environments Using Potential Fields. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 1232–1237, 1998.

[78] J. Hallset. Testing the Robustness of an Underwater Vision System. *Laplante P, Stoyenko A (eds) Real-time imaging: theory, techniques, and applications. IEEE Press*, pages 225–260, 1996.

[79] K. Han and G.N. DeSouza. Multiple Targets Geolocation Using SIFT and Stereo Vision on Airbone Video Sequences. In *Proc. of IROS*, pages 5327–5332, 2009.

[80] D.J. Hand and R. J. Till. A simple generalization of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

[81] J. A. Hanley and B. McNeil. The meaning and use of the area under a receiver operating charateristic (ROC) curve. *Radiology*, 143(1):381–395, 1982.

[82] C. Harris and M. Stephens. Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference*, pages 147–151, Manchester, (UK), 1988.

[83] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, Cambridge, UK, 2003.

[84] M. Hashima, F. Hasegawa, S. Kanda, T. Maruyama, and T. Uchiyama. Localization and Obstacle Detection for a Robot for Carrying Food Trays. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 345–351, 1997.

[85] R. Haywood. Acquisition of a Micro Scale Photographic Survey Using an Autonomous Submersible. In *Proc. of IEEE Oceans*, volume 18, pages 1423– 1426, 1986.

[86] I. Horswill. Visual Collision Avoidance by Segmentation. In *In Proc. of ARPA94*, pages II:1135–1141, 1994.

[87] A. Howard. Real Time Stereo Visual Odometry for Autonomous Ground Vehicles. In *Proc. of IROS*, 2008.

[88] A. Howard, E. Tunstel, D. Edwards, and A. Carlson. Enhancing Fuzzy Robot Navigation Systems by Mimicking Human Visual Perception of Natural Terrain Traversability. In *IFSA World Congress and 20th NAFIPS International Conference*, volume 1, pages 7–12, 2001.

[89] S. Hrabar, G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined Optic-Flow and Stereo-Based Navigation of Urban Canyons for a UAV. In *Proc. of IEEE Int'l Conf. of Intelligent Robots and Systems (IROS)*, pages 3309–3316, 2005.

[90] S. Huang and Dissanayake. Convergence and Consistency Analysis for Extended Kalman Filter based SLAM. *IEEE Transactions on Robotics*, 23(5):1036–1049, 2007.

[91] T.M. Jochem, D.A. Pomerleau, and C.E. Thorpe. Vision Based Neural Network Road and Intersection Detection and Traversal. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 344–349, 1995.

[92] A.E. Johnson, S. Goldberg, Y. Cheng, and L. Matthies. Robust and Efficient Stereo Feature Tracking for Visual Odometry. In *Proc. of ICRA*, pages 39–46, 2008.

[93] S.D. Jones, C. Andresen, and J.L. Crowley. Appearance Based Processes for Visual Navigation. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 551–557, 1997.

[94] M. Kabuka and A.E. Arenas. Position Verification of a Mobile Robot Using Standard Pattern. *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 3(6):505–516, 1987.

[95] A.C. Kak and G.N. de Souza. Robotic Vision: What Happened to the Visions of Yesterday? In *Proc. of 16th International Conference on Pattern Recognition*, pages 839–847, 2002.

[96] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME. Journal of Basic Engineering*, 1(82):35–45, 1960.

[97] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. In *IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[98] K. Kidono, J. Miura, and Y. Shirai. Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience. *Robotics and Autonomous Systems*, 40(2-3):121–130, 2002.

[99] D. Kim and R. Nevatia. Simbolic Navigation with a Generic Map. In *Proc. of IEEE Workshop on Vision for Robots*, pages 136–145, 1995.

[100] Y. Koren and J. Borenstein. Potential Fields Methods and Their Inherent Limitations for Mobile Robot Navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[101] A. Kosaka and A.C. Kak. Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties. *Computer Vision, Graphics, and Image Understanding*, 56(3):271–329, 1992.

[102] J. Kosecka, L. Zhou, P. Barber, and Z. Duric. Qualitative Image Based Localization in Indoors Environments. In *Proc. of IEEE Intl Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 3–8, 2003.

[103] E. Krotkov and M. Herbert. Mapping and Positioning for a Prototype Lunar Rover. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 2913–2919, 1995.

[104] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[105] S. Lenser and M. Veloso. Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 886–891, 2003.

[106] J. Leonard and H. F. Durrant-Whyte. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *Proc. of IEEE Intl Workshop on Intelligent Robots and Systems*, pages 1442–1447, 1991.

[107] B. Liang and N. Pears. Visual Navigation Using Planar Homographies. In *Proc. of IEEE Int'l Conf. on Robotics and Automations (ICRA)*, volume 1, pages 205–210, 2002.

[108] M. Liu, D. Scaramuzza, C. Pradalier, R. Siegwart, and Q. Chen. Scene Recognition with Omnidirectional Vision for Topological Map using Lightweight Adaptive Descriptors. In *Proc. of IROS*, pages 116–121, 2009.

[109] L.M. Lorigo and R.A. Brooks nad W.E. Grimson. Visually-Guided Obstacle Avoidance in Unstructured Environments. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 373–379, 1997.

[110] J-F. Lots, D.M. Lane, and E. Trucco. Application of a 2 1/2 D Visual Servoing to Underwater Vehicle Station-Keeping. In *Proc. of IEEE Oceans*, 2000.

[111] D.G. Lowe. Object Recognition From Local Scale Invariant Features. In *Proc. of International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[112] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[113] G. Ma, S.B. Park, S. Mller-Schneiders, A. Ioffe, and A. Kummert. Vision-based pedestrian detection - reliable pedestrian candidate detection by combining ipm and a 1d profile. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 137–142, Seattle, WA, USA, September 2007.

[114] JM. Maja, T. Takahashi, Z.D. Wang, and E. Nakano. Real-Time Obstacle Avoidance Algorithm for Visual Navigation. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 925–930, 2000.

[115] H.A. Mallot, H.H. Buelthoff, J.J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biomedical and Life Sciences, Computer Science and Engineering*, 64(3):177–185, 1991.

[116] A. Manessis and A. Hilton. Scene Modelling from Sparse 3D Data. *Journal of Image and Vision Computing*, 23(10):900–920, 2005.

[117] A. Manessis, A. Hilton, P. Palmer, P. McLauchlan, and X. Shen. Reconstruction of Scene Models from Sparse 3D Structure. In *Proc. of IEEE Intl Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2673, 2000.

[118] R.L. Marks, S. Rocks, and M.J. Lee. Real-Time Video Mosaicing of the Ocean Floor. In *Proc. of Symposium on Autonomous Underwater Vehicle Technology*, pages 21–27, 1994.

[119] S. Martens, P. Gaudian, and G.A. Carpenter. Mobile Robot Sensor Integration with Fuzzy ARTMAP. In *Proc. of the 1998 IEEE ISIC/CIRA/ISAS Joint Conferenc*, pages 307–312, 1998.

[120] M. C. Martin. Evolving Visual Sonar: Depth from Monocular Images. *Pattern Recognition Letters*, 27:1174–1180, 2006.

[121] C. Marzban. The ROC Curve and the Area Under it as Performance Measures. *Weather and Forecasting*, 19(6):1106–1114, 2004.

[122] S. Matsumoto and Y. ito. Real-time Based Tracking of Submarine Cables for AUV/ROV. In *Proc. of IEEE Oceans*, volume 3, pages 1997–2002, 1995.

[123] Y. Matsumoto, K. Ikeda, M. Inaba, and H. Inoue. Visual Navigation Using Omnidirectional View Sequence. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 317–322, 1999.

[124] Y. Matsumoto, M. Inaba, and H. Inoue. Visual Navigation Using View Sequenced Route Representation. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, volume 1, pages 83–88, 1996.

[125] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue. View-Based Approach to Robot Navigation. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)* , pages 1702–1708, 2000.

[126] L. Matthies. *Dynamic Stereo Vision*. Cmu-cs-89-195, Dept of Computer Science, Carnegie Mellon University, 1989.

[127] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars Microrover Navigation: Performance Evaluation and Enhancement. In *Proc. of IEEE Int'l Conf. of Intelligent Robots and Systems (IROS)*, volume 1, pages 433–440, 1995.

[128] L. Matthies and S.A. Shafer. Error Modeling in Stereo Navigation. *IEEE Journal of Robotics and Automation*, 3(3):239–248, 1987.

[129] P. McLauchlan and D. Murray. A Unifing Framework for Structure and Motion Recovery from Image Sequences. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 314–320, 1995.

[130] L.O. Mejias, S. Saripalli, G.S. Sukhatme, and P.C. Cervera. Detection and Tracking of External Features in an Urban Environment Using an Autonomous Helicopter. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 3972–3977, 2005.

[131] M. Meng and A.C. Kak. Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models. In *Proc. of IEEE Control Systems*, pages 30–39, 1993.

[132] M. Meng and A.C. Kak. NEURO-NAV: A Neural Network Based Architecture for Vision-Guided Mobile Robot Navigation Using non-Metrical Models of the Environment. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, volume 2, pages 750–757, 1993.

[133] K. Mikolajczyk and C. Schmid. Indexing Based on Scale Invariant Interest Points. In *Int'l Conference on Computer Vision (ICCV)*, pages 525–531, Vancouver, (Canada), 2001.

[134] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE TPAMI*, 27(10):1615–1630, 2005.

[135] K.S. Miller and D.M. Leskiw. *An Introduction to Kalman Filtering with Applications*. Krieger Publishing Company, 1987.

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

[137] J. Minguez, J. Osuna, and L. Montano. A "Divide and Conquer" Strategy based on Situations to Achieve Reactive Collision Avoidance in Troublesome Scenarios. In *Proc. of IEEE Int Conference on Robotics and Automation (ICRA)*, pages 3855 – 3862, 2004.

[138] H.P. Moravec. The Stanford Cart and the CMU Rover. In *Proceedings of IEEE*, volume 71, pages 872–884, 1983.

[139] H. Morita, M. Hild, J. Miura, and Y. Shirai. Panoramic View-Based Navigation in Outdoor Environments Based on Support Vector Learning. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 2302–2307, 2006.

[140] H. Morita, M. Hild, J. Miura, and Y. Shirai. Panoramic View-Based Navigation in Outdoor Environments Based on Support Vector Learning. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 1885–1890, 2006.

[141] A. Mourikis, N. Trawny, S. Roumeliotis, A. Johnson, E. Ansar, and L. Matthies. Vision-Aided Inertial Navigation for Spacecraft Entry, Descent and Landing. *IEEE Transactions on Robotics*, 25(2):264–280, 2009.

[142] A.C. Murillo, J.J. Guerrero, and C. Sags. Topological and Metric Robot Localization Through Computer Vision Techniques. In *IEEE ICRA*, pages 79–85, Rome, (Italy), 2007.

[143] S. Negahdaripour, X. Xu, and L. Jin. Direct Estimation of Motion From Seafloor Images for Automatic Station-Keeping of Submersible Platforms. *IEEE Journal of Oceanic Engineering*, 24(3):370–382, 1999.

[144] T. Netter and N. Franceschini. A Robotic Aircraft that Follows Terrain Using a Neuromorphic Eye. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 129–134, 2002.

[145] D. Nistr, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the IEEE Int'l Conference on Computer Vision and Pattern Recognition (CVPR)*, volume Vol. 1, pages 652–659, Princeton, NJ, USA, June 2004.

[146] T. Ohno, A. Ohya, and S. Yuta. Autonomous Navigation for Mobile Robots Referring Pre-Recorded Image Sequence. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 672–679, 1996.

[147] A. Ollero, J. Ferruz, F. Caballero, S. Hurtado, and L. Merino. Motion Compensation and Object Detection for Autonomous Helicopter Visual Navigation in the COMETS System. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 19–24, 2004.

[148] C.F. Olson, L. H. Matthies, M. SChoppers, and M.W. Maimone. Robust Stereo Ego-Motion for Long Distance Navigation. In *Proc. of CVPR*, volume 2, pages 453–458, 2000.

[149] G. Oriolo, G. Ulivi, and M. Vendittelli. On-Line Map Building and Navigation for Autonomous Mobile Robots. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 2900–2906, 1995.

[150] J. Pan, D.J. Pack, A. Kosaka, and A.C. Kak. FUZZY-NAV: A Vision-Based Robot Navigation Architecture Using Fuzzy Inference for Uncertainty-Reasoning. In *Proc. of IEEE World Congress Neural Networks*, volume 2, pages 602–607, 1995.

[151] I. Parra, M.A. Sotelo, and L. Vlacic. Robust Visual Odometry for Complex Urban Environments. In *Proc. of IVS*, pages 440–445, 2008.

[152] L.M. Paz, P. Piniés, J.D. Tardós, and J. Neira. Large Scale 6DOF SLAM with a Stereo Camera in Hand. *IEEE Transactions on Robotics*, 2007.

[153] N. Pears and B. Liang. Ground Plane Segmentation for Mobile Robot Visual Navigation. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 1513–1518, 2001.

[154] P. Pinis and J.D. Tards. Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision. *IEEE Transactions on Robotics*, 24(5), 2008.

[155] D.A. Pomerleau. ALVINN: an Autonomous Land Vehicle in a Neural Network. Technical Report CMU-CS-89-107, Carnegie Mellon University, 1989.

[156] V. Ayala Ramirez, J.B. Hayet, F. Lerasle, and M. Devy. Visual localization of a mobile robot in indoor environments using planar landmarks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[157] A. Remazeilles, F. Chaumette, and P. Gros. Robot Motion Control from a Visual Memory. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 4695–4700, 2004.

[158] A. Remazeilles, F. Chaumette, and P. Gros. 3D Navigation Based on a Visual Memory. In *Proc. of IEEE Int'l Conf. on Robotics and Automations (ICRA)*, pages 2719–2725, 2006.

[159] J. Rife and S. Rock. A Low Energy Sensor for AUV-Based Jellyfish Tracking. In *Proc. of 12th International Symposium on Unmanned Untethered Submersible Technology*, 2001.

[160] P. Rives and J.J Borelly. Visual Servoing Techniques Applied to an Underwater Vehicle. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 20–25, 1997.

[161] R. Roberts, H. Nguyen, N. Krishnamurthi, and T. Balch. Memory-based Learning for Visual Odometry. In *Proc. of ICRA*, pages 47–52, 2008.

[162] R. Rodrigo, M. Zouqi, Z. Chen, and J. Samarabandu. Robust and efficient feature tracking for indoor navigation. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 39(No. 3):658–671, 2009.

[163] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *ECCV*, volume 1, pages 430–443, Gratz, (Austria), 2006.

[164] E. Royer, J. Bom, M. Dhome, B. Thuillot, M. Lhuillier, and F. Marmoiton. Outdoor Autonomous Navigation Using Monocular Vision. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 3395–3400, 2005.

[165] P. Saeedi, P.D. Lawrence, and D. G. Lowe. Vision-Based 3-D Trajectory Tracking for Unknown Environments. *IEEE Transactions on Robotics*, 22(1):119–136, 2006.

[166] J. Santos-Victor and G. Sandini. Visual-Based Obstacle Detection: a Purposive Approach Using the Normal Flow. In *Proc. of International Conference on Intelligent Autonomous Systems*, 1995.

[167] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergence Stereo for Robot Navigation: Learning from Bees. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 1993.

[168] D. Schleicher, L. M. Bergasa, R. Barea, E. Lopez, and M. Ocaa and. Real-Time Simultaneous Localization and Mapping Using a Wide-Angle Stereo Camera. In *Proc. of IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 55–60, 2006.

[169] S. Se, D.G. Lowe, and J.Little. Mobile Robot Localization and Mapping with Uncertainty Using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, 21(8):735–758, 2002.

[170] S. Se, D.G. Lowe, and J. Little. Vision-Based Global Localization and Mapping for Mobile Robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005.

[171] A.J. Serrano, E. Soria, J.D. Martin, R. Magdalena, and J. Gomez. Feature Selection Using ROC Curves on Classification Problems. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2010.

[172] J. Shen and H. Hu. Visual Navigation of a Museum Guide Robot. In *The Six World Congress on Intelligent Control and Automation (WCICA)*, volume 2, pages 9169–9173, 2006.

[173] J. Shi and C. Tomasi. Good Features to Track. In *Proc. of IEEE Intl Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[174] Y. Shu and Z. Tan. Vision-based lane detection in autonomous vehicle. In *Proceedings of the Congress on Intelligent Control and Automation*, pages 5258–5260, Xi'an Jiaotong, China, June 2004.

[175] C. Siagian and L. Itti. Biologically Inspired Mobile Robot Vision Localization. *IEEE Transactions on Robotics*, 25(4):861–873, 2009.

[176] C. Silpa-Anan, T. Brinsmead, S. Abdallah, and A. Zelinsky. Preliminary Experiments in Visual Servo Control for Autonomous Underwater Vehicle. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, volume 4, pages 1824–1829, 2001.

[177] R. Sim and G. Dudek. Learning Generative Models of Scene Features. In *Proc. of IEEE Intl Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 406–412, 2001.

[178] R. Sim and G. Dudek. Effective Exploration Strategies for the Construction of Visual Maps. In *Proc. of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, volume 3, pages 3224–3231, 2003.

[179] R. Sim, P. Elinas, M. Griffin, A. Shyr, and J. J. Little. Design and Analysis of a Framework for Real-Time Vision-Based SLAM Using Rao-Blackwellised Particle Filters. In *Proc. of the 3rd Canadian Conference on Computer and Robotic Vision*, 2006.

[180] R. Sim and J. J. Little. Autonomous Vision-Based Exploration and Mapping Using Hybrid Maps and Rao-Blackwellised Particle Filters. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[181] N. Simond and M. Parent. Obstacle detection from ipm and super-homography. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 4283–4288, California, Sant Diego, USA, November 2007.

[182] R. Smith, M. Self, and P. Cheeseman. Estimating Uncertain Spatial Relationship in Robotics. *Autonomous Robot Vehicles (I. Cox and G.T. Wilfong eds)*, pages 167–193, 1990.

[183] S. Smith and J. Brady. SUSAN - a New Approach to Low Level Image Processing. *Journal of Computer Vision*, 23(1):45–78, 1997.

[184] V. Srinivasan, M. Lehrer, W.H. Kircner, and S.W. Zhang. Range Perception Through Apparent Image Speed in Freely Flying Honeybees. *Visual neuroscience*, 6(5):519–35, May 1991.

[185] V. Srinivasan, S. Thurrowgood, and D. Soccol. An Optical System for Guidance of Terrain Following in UAVs. In *Proc. of the IEEE International Conference on Video and Signal Based Surveillance (AVSS)*, pages 51–56, 2006.

[186] V. Srinivasan, S. W. Zhang, J. S. Chahl, G. Stange, and M. Garratt. An Overview of Insect-Inspired Guidance for Application in Ground and Airborne Platforms. *Proc. of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 218(6):375–388, 2004.

[187] S. Stephen, D.G. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, Vol. 21(No. 3):364–375, 2005.

[188] K. Sugihara. Some Location Problems for Robot Navigation Using a Single Camera. *Computer Vision, Graphics, and Image Processing*, 42:112–129, 1988.

[189] A. Talukder, S. Goldberg, L. Matties, and A. Ansar. Real-time Detection of Moving Objects in a Dynamic Scene from Moving Robotic Vehicles. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 1308–1313, October 2003.

[190] A. Talukder and L. Matties. Real-time Detection of Moving Objects from Moving Vehicles Using Dense Stereo and Optical FLow. In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 3718–3725, October 2004.

[191] S. Temizer and L. P. Kaelbling. Optical Flow Based Local Navigation. Web Page. Massachusetts Institute of Technology, 2003. http://people.csail.mit.edu/lpk/mars/.

[192] C. Thorpe. FIDO: Vision and Navigation for a Mobile Robot. In *PhD dissertation, Dept Computer Science, Carnegie Mellon University*, 1983.

[193] C. Thorpe, M.H. Hertt, T. Kanade, and S.A. Shafer. Vision and Navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–372, 1988.

[194] C. Thorpe, T. Kanade, and S.A. Shafer. Vision and Navigation for the Carnegie-Mellon Navlab. In *Proc. of Image Understand Workshop*, pages 143–152, 1987.

[195] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[196] S. Thrun. Probabilistic Algorithms in Robotics. Technical Report CMU-CS-00-126, Carnegie Mellon University, 2000.

[197] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, and D. Fox. MINERVA: A Second-Generation Museum Tour-Guide robot. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, volume 3, pages 1999–2005, May 1999.

[198] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press Cambridge, 2005.

[199] S. Thrun, J.S. Gutmann, D. Fox, W. Burgard, and B.J. Kuipers. Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach. In *Association for the Advancement of Artificial Intelligence (AAAI)-98*, pages 989–995, 1998.

[200] M. Tomono. 3-D Object Map Building Using Dense Object Models with SIFT-Based Recognition Features. In *Proc. of IEEE Int'l Conf. of Intelligent Robots and Systems (IROS)*, pages 1885–1890, 2006.

[201] P. Torr, A.W. Fitzgibbon, and A. Zisserman. Maintaining a Multiple Motion Model Hypotheses Over Many Views to Recover Matching and Structure. In *Proc. of International Conference on Computer Vision (ICCV)*, pages 485–491, 1998.

[202] E. Trucco and K. Plakas. Video Tracking : A Concise Survey. *IEEE Journal of Oceanic Engineering*, 31(2):520–529, 2006.

[203] M.A. Truk, D.G. Morgenthaler, K.D. Gremban, and M. Marra. VITS - A Vision System for Autonomous Land Vehicle Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):342–361, 1988.

[204] S. Tsugawa, T. Yatabe, T. Hisrose, and S. Matsumoto. An Automobile With Artificial Intelligence. In *Proc. of Sixth International Joint Conference on Artificial Intelligence*, pages 893–895, 1979.

[205] T.Tsubouchi and S. Yuta. Map-Assisted Vision System of Mobile Robots for Reckoning in a Building Environment. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 1978–1984, 1987.

[206] T. Tuytelaars and K. Mikolajczyk. Local Invariant Feature Detectors: a Survey. *Foundations and Trends in Computer Graphics and Vision.*, 3(3):177–280, 2008.

[207] S. van der Zwaan and J. Santos-Victor. An Insect Inspired Visual Sensor for the Autonomous Navigation of a Mobile Robot. In *Proc. of the Seventh International Sysposium on Intelligent Robotic Systems (SIRS)*, 1999.

[208] B. Wilcox, L. Matties, D. Gennery, B. Copper, T. Nguyen, T. Litwin, A. Mishkin, and H. Stone. Robotic Vehicles for Planetary Exploration. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 175–180, 1992.

[209] B. Williams and I. Reid. On Combining Visual SLAM and Visual Odometry. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 3494–3500, 2010.

[210] N. Winters and J. Santos-Victor. Omnidirectional Visual Navigation. In *Proc. of IEEE Int'l Symposium on Intelligent Robotic Systems (SIRS)*, pages 109–118, 1999.

[211] D. Wooden. A Guide to Vision-Based Map Building. *IEEE Robotics and Automation Magazine*, 13:94–98, 2006.

[212] X. Xu and S. Negahdaripour. Mosaic-based positioning and improved motion estimation methods for automatic navigation of sumersible vehicles. *IEEE Journal of Oceanic Engineering*, 27(1):79–99, 2002.

[213] Kuk-Jin Yoon and InSo Kweon. Artificial Landmark Tracking Based on the Color Histogram . In *Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, volume 4, pages 1918–1923, 2001.

[214] C. Zhou, Y. Wei, and T. Tan. Mobile Robot Self-Localization Based on Global Visual Appearance Features. In *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 1271–1276, 2003.

[215] J. Zhou and B. Li. Homography-based Ground Detection for a Mobile Robot Platform Using a Single Camera. In *IEEE ICRA*, pages 4100–4101, Arizona, (Tempe, USA), 2006.