UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesis Doctoral

BEARING-ONLY SLAM METHODS

Rodrigo Francisco Munguía Alcalá

Director: Antoni Grau Saldes

Dept. d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial

Juliol de 2009

A mi hijo Roderic

Resumen

SLAM (Simulatenous Localization and Mapping) es quizá el problema más importante a solucionar en robótica para construir robots móviles verdaderamente autónomos. El SLAM es acerca de cómo un robot móvil opera en un entorno a priori desconocido, utilizando únicamente sus sensores de abordo, mientras construye un mapa de dicho entorno que al mismo tiempo utiliza para localizarse.

Los sensores del robot tienen un gran impacto en los algoritmos usados en SLAM. Los primeros enfoques se centraron en el uso de sensores de rango como sonares o láseres. Sin embargo hay algunas desventajas relacionadas con su utilización: La asociación de datos es difícil, son costosos, habitualmente están limitados a mapas 2D y tienen alto costo computacional debido al gran número de características (*features*) que producen.

Lo anterior ha propiciado que enfoques recientes se estén moviendo hacia el uso de cámaras como sensor principal. Estas se han vuelto muy atractivas para los investigadores de la robótica, dado que generan mucha información, facilitan la asociación de datos, están bien adaptadas para sistemas embebidos: son ligeras, baratas y ahorran energía. Usando visión, un robot puede localizarse así mismo usando objetos comunes como *landmarks*.

Sin embargo, a diferencia de los sensores de rango, que proveen información angular y de rango, una cámara es un sensor proyectivo que mide el *bearing* (ángulo) respecto a objetos de la imagen. Por lo que la profundidad (*range*) no puede ser obtenida en una sola toma. Este hecho ha motivado la aparición de una nueva familia de métodos de SLAM: Los *Bearing-Only SLAM methods*, los cuales están basados en técnicas especiales para la inicialización de *features*, permitiendo el uso de sensores de *bearing* en SLAM.

Esta tesis se centra en el estudio de la problemática del Bearing-Only SLAM: da una descripción extensa del tema, recapitula los retos actuales a resolver y propone nuevos métodos y algoritmos enfocados a tratar diferentes sub problemas concernientes esta problemática en general. Estos sub problemas deben de ser tratados, de manera que sea posible construir sistemas capaces de operar en entornos diversos y complejos.

La investigación descrita en esta disertación ha sido dividida en tres partes:

3DOF Bearing-Only SLAM: El proceso de inicialización de nuevas *features* es quizá el sub problema más importante a tratar en Bearing-Only SLAM. En esta parte de la tesis se introduce un nuevo método llamado *Delayed Inverse Depth Features Initialization* (para 3DOF y asumiendo odometría). Este método utiliza una parametrización inversa, donde la profundidad e incertidumbre iníciales de cada *feature* son dinámicamente estimadas previamente a que una *feature* sea declarada como un nuevo *landmark* en el mapa estocástico. También se presenta un sistema de SLAM basado en sonido, llamado SSLAM el cual usa fuentes de sonido como *features* del mapa. La contribución del SSLAM es demostrar la viabilidad de la inclusión del sentido auditivo en SLAM y mostrar que es factible utilizar sensores alternativos en Bearing-Only SLAM.

Métodos de asociación de datos para SLAM basado en visión: El problema de la asociación de datos es quizá uno de los problemas más difíciles en robótica y también uno de los sub problemas más importantes a tratar en SLAM. Consiste en determinar si las mediciones de un sensor tomadas en tiempos diferentes, corresponden al mismo objeto físico del mundo. En esta parte de la tesis, se proponen diferentes métodos que tratan el problema de la asociación de datos en un contexto de SLAM basado en visión.

SLAM monocular de 6DOF: El SLAM monocular de 6DOF quizá representa la variante más extrema del SLAM, dado que una cámara en mano es utilizada como la única entrada sensorial del sistema. En esta parte de la tesis, se extiende el algoritmo de 2DOF Bearing-Only SLAM para ser aplicado en un contexto de SLAM monocular. También se propone un nuevo esquema llamado SLAM Monocular Distribuido, enfocado en el problema de construir y mantener mapas consistentes de grandes entornos en tiempo real. La idea es dividir la estimación total del sistema en dos procesos de estimación concurrentes. Primero un método actual de SLAM monocular (Virtual Sensor) es modificado como un complejo sensor virtual que emula sensores típicos, como el laser para medición de rango y encoders para odometría. Después otro método tradicional de SLAM (Global SLAM) es acoplado para construir y mantener el mapa final.

Numerosas referencias bibliográficas, graficas, comparaciones, simulaciones y experimentos con datos reales de sensores, son presentador con el fin de mostrar el desempeño de los métodos propuestos.

Abstract

Simultaneous Localization and Mapping (SLAM) is perhaps the most fundamental problem to solve in robotics in order to build truly autonomous mobile robots. SLAM is about on how can a mobile robot operate in an *a priori* unknown environment and use only onboard sensors to simultaneously build a map of its surroundings and use it to track its position.

The robot's sensors have a large impact on the algorithm used for SLAM. Early SLAM approaches focused on the use of range sensors as sonar rings or lasers. Nevertheless there are some disadvantages with the use of range sensors in SLAM: Correspondence or data association is difficult. They are expensive. They are generally limited to 2D maps and computational overhead due to large number of features.

The aforementioned issues have propitiated that recent work is moving towards the use of cameras as the primary sensing modality. Cameras have become more and more interesting for the robotic research community, because it yield a lot of information allowing reliable data association. Cameras are well adapted for embedded systems: they are light, cheap and power saving. Using vision, a robot can localize itself using common objects as landmarks.

On the other hand, at difference of range sensors (i.e. sonar or laser) which provides range and angular information, a camera is a projective sensor which measures the bearing of images features. Therefore depth information (range) cannot be obtained in a single frame. This fact has propitiated the emergence of a new family of SLAM methods: The Bearing-Only SLAM methods, which mainly relies in especial techniques for features system-initialization in order to enable the use of bearing sensors (as cameras) in SLAM systems.

This thesis is focused on the study of the Bearing-Only SLAM problematic: It gives an extensive overview of the subject. It point out the principal challenges nowadays. And it presents new methods and algorithms which address different sub problems concerning to the Bearing-Only SLAM problematic. These sub problems must be solved, in order to build systems capable of operating in extremely diverse and complex environments.

The research described in this dissertation has been divided into three parts:

3DOF Bearing-Only SLAM: The initialization process for new features is perhaps the most important sub problem for addressing in Bearing-Only SLAM. In this part of the thesis we introduce a novel method called *Delayed Inverse Depth Features Initialization* for a 3DOF odometry-available context. In this method, which uses an inverse depth parameterization, initial depth and uncertainty of each feature are dynamically estimated priors to add the new landmark in the stochastic map. We also present a Sound-based SLAM system, called SSLAM, which uses "Sound Sources" as map's features. The main contributions of the SSLAM are demonstrating the viability on the inclusion of the hearing sense in SLAM and show that is straightforward to use alternative bearing in SLAM systems.

Data association methods for camera-based SLAM: the data association problem is possibly one of the hardest problems in robotic and also one of the most important sub problems to solve in SLAM. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world. In this part of the thesis, we propose different methods for addressing the data association problem in a context of vision-based SLAM.

6DOF Monocular SLAM: 6-DOF monocular SLAM possibly represents the harder variant of SLAM, since a low cost hand-held camera is used as the only sensory input to the system. In this part of the thesis, we extend our 2DOF Bearing-Only SLAM algorithm for being used in a monocular SLAM context. Also a novel framework called Distributed Monocular SLAM is proposed for addressing the problem of building and maintaining a global and consistent map of large environments at real time. The key idea is to divide the whole estimation into two concurrent estimation processes. First a state of the art monocular SLAM method (Called Virtual Sensor) is modified as a complex virtual sensor that emulates typical sensors such as laser for range measurement and encoders for dead reckoning. Afterward, a classic SLAM method (called Global SLAM) is plugged in for building and maintaining the final map.

Several references, graphics, comparisons, simulations and experiments with real data are presented in order to demonstrate the performance of the methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis has to do with robotics. When I say this, some persons can imagine that this work is about of educating a clumsy shiny-gold humanoid, but of course nowadays, it is matter of science fiction. Others can think that this dissertation is about programming a robot arm for assembling cars or other goods in a production line. Nevertheless this thesis intends to contribute to experimental robotics instead of the well-understood area of the industrial robotics. Actually in our work, the robot itself as an electromechanical device is more or less irrelevant, since it focus in general algorithms which can be implemented using different hardware configurations. The problem addressed in this work is very significant to the pursuit of building truly autonomous mobile robots capable to operate in extremely diverse environments ranging from the Martian surface to a living room. Basically the problem is: how does a mobile robot can autonomously know where it is?

Having a machine answer "where am I?" is an engineering problem that has been at the heart of mobile robotics research for over two decades, and while a great progress has been made, there is a long way in order have mobile robots that operates robustly in exotic locations where humans can't reach or are unwilling to work. Exploring and realizing different tasks in different environments (known or unknowns) implies that the robot using their onboard sensors becomes aware of its location at every time. This is analogous to the manner as humans using their senses (mainly vision) moves in their daily live.

At first glance, one could think that a GPS (global positioning system) mounted in a mobile robot will solve fully the problem. However GPS doesn't work sub-sea, underground, in buildings or on Mars. Even for outdoor mobile robots localization, where GPS has obvious advantages: position data are directly given in an absolute frame, and the required infrastructure is reduced to a sole fixed station in the case of differential systems. Yet, the use of this solution raises a number of issues, such as the satellite maskings, or the existence of the so-called GPS latency which delays the output of the localization data.

Another approach could be providing to the robot in advance with a map of the environment or information about in how it looks like. Actually in some scenarios like a factory or in an office building, can be viable provide the robot with an a priori map, often in the form of artificial landmarks fixed to strategic locations of the environment. The problem is that it is not always possible, let alone convenient or cheap, to install this infrastructure, and once installed it is inflexible and imposes unnatural constraints on the workspace. Even a small change in a "route" of the robot or the working area can involve a huge effort for reengineering the "fixed-map".

Now, consider an autonomous vehicle traveling through *a priori* unknown environment. While the vehicle is moving, makes different measurements (using their onboard sensors). At the same time it uses these measurements in order to (incrementally) learn a map. It could then use this map to answer the "where am I" question, to locate itself. This scenario is called the Simultaneous Localization and Mapping (SLAM) problem and can be stated as follows:

"How can a mobile robot operate in an *a priori* unknown environment and use only onboard sensors to simultaneously build a map of its workspace and use it to navigate?"

The tricky part is that this is a chicken and egg problem: to build a map you need to know where you (the observer) are but at the same time you need a map to figure out where you are. The simplicity of the SLAM problem statement is fascinating. It is after all something that we humans, with varying degrees of success, do naturally - for example when stepping out of a hotel lobby in a new city. Yet SLAM in particular is a topic that has challenged the robotics research community for over a decade. The SLAM problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots.

1.1 Motivation

Nowadays, due to its great potential in a huge range of applications, SLAM is one of the most active research fields in robotics. SLAM has had excellent results during the last years, but until recently it was mainly restricted to the use of sonar and laser range-finder sensors.

There are some disadvantages with the use of range sensors in SLAM:

- Correspondence or data association is difficult.
- They are expensive.
- Generally limited to 2D maps.
- Computational overhead due to large number of features.

In this dissertation we explore an alternative to the range sensors: The use of Bearing Sensors in SLAM, so called Bearing-Only SLAM. In counterpart to the range-finder sensors, which provides range and angular information, bearing sensors only offers angular information. At first glance this fact could represent a disadvantage of the bearing sensors respect to the range sensors, but as we will see, there are several compensations in the use of a bearing sensor in order to perform SLAM.

A camera is a projective sensor which measures the bearing of images features. Cameras have become more and more interesting for the robotic research community as sensors, because they yield a lot of information. It is a sensor from which 3D information can be extracted. Even for indoor robots whose pose can be represented in 2D, the ability to gather 3D information on the environment is essential. Cameras are well adapted for embedded systems: they are light, cheap and power saving. A wide variety of algorithms can be obtained from the vision research community in order to extract high level primitives from the image, and matching them with primitives stored in the map thus allowing reliable data association, which is one the most important problems to solve in SLAM.

Some of the advantages for the use of cameras can be summarized as follows:

- A wide variety of algorithms can be obtained from the vision research community for extract high level primitives from the image.
- Cameras are well adapted for embedded systems; they are light, cheap and power saving.
- A lot of data available, thus allowing to reliable data association.
- 3D information can be extracted.

The use of cameras in SLAM involves several challenges but also implies a huge potential. As computational power grows, an inexpensive camera can be used to perform range and appearance–based sensing simultaneously, by replacing typical sensors as laser and sonar rings for range measurement and encoders for dead reckoning. Currently the objectives in vision-based SLAM is that emulates (and ultimately surpass) the results in large-scale mapping achieved using laser range-finder sensors, aiming to build vision-only SLAM systems with the potential of guiding autonomous robots in their exploration and operation in large and complex environments.

For this thesis, it is important to note that, when we are talking about vision-based SLAM systems, we are specially discussing about monocular-based vision systems.

That is, systems where a single camera is the only sensorial input. This is in counterpart to the stereo vision systems which commonly are seen as a sole vision-based sensor that provides bearing among depth information. Therefore a stereo vision sensor could be seen more like range-finder sensor. The regular SLAM algorithms assume that depth information is measured or estimated in the side of the sensor. On the other hand, Bearing-Only SLAM assumes that the sensor, no matter which is, provides only bearing information (in terms of spatial information). In Bearing-Only SLAM systems, depth information is estimated using two or more measurements of the sensor and this estimation process is done as an implicit part of the whole SLAM algorithm. A Monocular SLAM system can be viewed as specific case of a general Bearing-Only SLAM system using a single camera as the solely sensorial input.

As was stated early, this dissertation aims to contribute to the robotics field, nevertheless the use of cameras in SLAM is not limited to the purely robotics applications. In that sense, the family of algorithms analyzed in this work can be straightforward used in a wide range of applications. Some examples are:

- Mobile robot autonomous navigation:
  - **Position estimation and map building (this thesis)**
  - Path planning and control.
- Wearable robotics:
  - Motion estimation for camera equipped devices worn by humans.
- Telepresence:
  - Head motion estimation using an out-world-looking camera.
- Television:
  - Camera motion estimation for live augmented reality.
- Walking assistance system:
  - Smart cane for elderly and visually impaired.
  - Obstacle detection and avoidance.

Cameras are by far the most popular bearing sensor. Nevertheless Bearing-Only SLAM is not limited to the use of cameras, in that sense, other sensorial capabilities (much less explored), as the auditory sense, can be investigated in the context of Bearing-Only SLAM. In this case, sound sources (artificial or natural) are used as landmarks for being included in the robot's map in order to localize it along the time.

This dissertation intent to cover most of the problematic related with the use of bearing sensors in SLAM. In that sense, a considerable part of the work focuses on monocular SLAM. But we analyze as well the applications of alternative bearing sensors in SLAM. For example, a sound sensor mounted on a small mobile robot, capable of detecting and tracking a sound source, is also used in experiments in order to perform SLAM.

1.2 Objectives

The primary goal of this thesis is the study, analysis, design, implementation and experimentation of Bearing-Only SLAM algorithms and methods, focusing in the robotics field.

Sub objectives:

- To study and analyzing the state of the art simultaneous localization and mapping algorithms, in order to determines general issues that must be improved in order to build systems capable of operating in complex environments.

- Bearing-only SLAM is a partially observable SLAM problem, in which the sensor, used for perceiving the robot´s environment, provides only angular information respect to the landmarks, and therefore does not give enough information to compute the full state (bearing and depth) of a landmark from a single observation. Therefore the initialization of new features in Bearing-Only SLAM is a fundamental problem to be addressed. A sub objective of this thesis is to propose a new algorithm of Bearing-Only SLAM. It should offer improvements respect to similar approaches.

- The closing loop problem after long trajectories at real-time operation is currently one of the open challenges in SLAM. One of the sub objectives of this thesis is to contribute to the goal of make possible in a future that a Bearing-Only SLAM algorithm can be applied to large and complex environments at real-time.

- Cameras are by far the most popular bearing sensor, among other things because provides a huge amount of information useful for addressing the data association which is a fundamental problem to solve in order to implement SLAM. A sub objective is to make contributions to the problematic of the data association in a SLAM.

- To demonstrate that Bearing-Only SLAM systems, which are commonly based on cameras as primary sensors, can be straightforward extended for its application based on alternative bearing sensors.

1.3 Thesis Statement and Contributions

The research described in this dissertation has been divided into three parts: (i) 3-DOF Bearing-Only SLAM with application to Sound Based SLAM in mobile robots, (ii) Data association methods for vision-based SLAM (iii) 6-DOF Monocular SLAM and Distributed framework:

- In part I of the research, we propose a novel method for initializing new features for a general scenario of Bearing-Only SLAM; this method is called *Delayed-Inverse Depth Features Initialization*. The method is introduced in a 2D context (3-DOF) and assuming availability of odometry. Theoretical and experimental bases which inspired this method are given, and several simulations are presented in order to demonstrate its performance and feasibility. A comparison with the *Un-delayed Inverse-Depth Feature Initialization* is also presented. Finally in this part of the thesis, based in our general 3-DOF *Bearing-Only SLAM algorithm*, it is presented a Sound-based SLAM system (called SSLAM) which uses "Sound Sources" as map's features. The main contributions of the SSLAM is demonstrating the viability of the inclusion of the hearing sense in SLAM and demonstrating that is straightforward modify a Bearing-Only method (originally conceived for vision-based systems) for its use based in an alternative bearing sensor.

- In part II of the research, we propose different methods for addressing the data association problem in a context of vision-based SLAM. First a novel image feature descriptor called ICAD is presented. ICAD descriptors are based on ICA (Independent Component Analysis). Later, a framework for capturing the variability of image features descriptors based on statistical methods is presented, in order to make features descriptors more robust to changes in illumination of point of view. It seen that some of the techniques currently available in the literature have shown to be an excellent option for addressing the data association problem, nevertheless their direct use in SLAM applications is often not straightforward. Finally a simple but effective framework is proposed in order to take advance of the state-of-the-art image features descriptors in a SLAM context.

- In part III of the research, a novel approach for monocular SLAM is presented. This method, called *Delayed Inverse Depth Monocular SLAM* method, is an extension of our general *Bearing-Only SLAM algorithm*,

described in part I, for be used in monocular SLAM. In this context, the 6-DOF monocular camera case (Monocular SLAM) possibly represents the harder variant of SLAM, since a low cost hand-held camera is used as the only sensory input to the system. One of the major challenges in monocular SLAM consist in extend the application of the current methods to large and dynamic environments. In this part of the research, a novel framework, called Distributed Monocular SLAM is also proposed, for addressing the problem of building and maintaining a global and consistent map of large environments at real time.

In the appendix A the publications derived from the research presented in this dissertation are listed.

1.4 Outline of the Thesis

In the following, the content of each chapter is summarized.

In chapter 2, antecedents regarding to the thematic addressed in the thesis are presented. First, a historical overview in localization and mapping problem is given; here some earliest approaches and taxonomies are referenced. Later some of the factors that make SLAM a challenging problem are listed. In this chapter is clarified why is convenient to solve the localization and mapping problems concurrently. Finally an outline of the mathematical basis of Simultaneous Localization and Mapping problem (SLAM) is given; a general formulation of the SLAM problem is presented and also the most important approaches for solve it, are introduced. This section is relevant since gives an introduction to several topics and concepts used along the thesis. This chapter includes several references to relevant work regarding to the research area addressed by the thesis.

Chapter 3 details the most popular solution for the SLAM problem: the Extended Kalman Filter (EKF) general solution for SLAM. The EKF based SLAM is very relevant to this work because our proposed methods for Bearing-Only SLAM fits in this family of SLAM algorithms. In this chapter, the basic components of a general SLAM system are exposed. In that sense, several equations that will be used recurrently along the thesis are given in their simplest form. Several experimental simulations are also presented in order to illustrate the behavior and main objectives for SLAM systems.

Chapter 4 is devoted to expose the part I of the research regarding to this thesis. In the first part of this chapter, a depth introduction to the Bearing-Only problematic is given. It mainly focuses into the initialization of new features, which represent the main

challenge in order to implement Bearing-Only SLAM systems. Using simulations, the viability of this kind of methods is also reviewed. Later, related work representative of the state of the art, together whit a brief taxonomy and a summary of methods are presented. In the second part of this chapter one of the most important contributions of this thesis is introduced: the *Delayed Inverse-Depth Feature Initialization*, which is a novel method for adding new features to the map in Bearing-Only SLAM systems. This method is based on the *unified inverse depth method* [1] which due to its clarity and scalability is a good option for monocular SLAM implementations. The experiments with the unified inverse depth method show that, when initial reference points are used for establishing a metric scale in the map, the initial features depths have to be tuned, otherwise, is likely that new features added to the map never converges respect to the metric reference. In the monocular SLAM case, when features are initialized in the first observed frame (un-delayed initialization), usually the weak long-term image features are added to the map. Therefore it is difficult to match them in subsequent frames. When a minimum number of active image features want to be maintained, it could happen that unnecessary initializations are realized. Our proposed method overcome the aforementioned issues treating the initial inverse depth and their associated initial uncertainty before they are added to the system state instead of using a fixed initial depth and uncertainty. At the same time features can be implicitly tested prior to be added to map in order to prune weak long-term features. In chapter 4 our *Delayed Inverse-Depth Feature Initialization* method is introduced assuming a 2D context with availability of odometry. First the theoretical and experimental bases of the method are given, later the method is widely outlined and finally several simulations are presented in order to demonstrate its performance and feasibility. In this part a comparison with the *Un-delayed Inverse-Depth Feature Initialization* is also presented. The third and final part of this chapter is dedicated to presents a Sound-based SLAM system (called SSLAM) which uses “Sound Sources” as map’s features. The SSLAM system which is based in our *Delayed Inverse-Depth Bearing-Only SLAM* method, demonstrates the feasibility of the inclusion of the hearing sense in SLAM. Experimental results with real data, obtained from the sensors of a small mobile robot, are presented In order to show the performance of the SSLAM method.

Chapter 5 presents the part II of the research regarding to this thesis. In this chapter several vision-based techniques, for addressing the data association problem in a SLAM context, are described. An additional aim of this chapter is to be a “bridge” between the chapter 4, where the Bearing-Only SLAM general problematic is introduced, and the chapter 5, where the Monocular SLAM problematic is studied as a sub-class of Bearing-Only SLAM based on monocular cameras. In the first part of this chapter, a small survive to the data association problem based on local image features is given. Related work and methods, representatives the state of the art, are also briefly exposed. Later a method for address the data problem based on a novel image feature descriptor is presented. This

novel descriptor called ICAD is based on Independent Component Analysis (ICA). Several experimental results are presented in order to show the performance of the method. In the second part of the chapter a new framework for capturing the variability of image features descriptors based on statistical methods is presented. This framework aims to improve the robustness of features descriptors to changes in illumination and point of view. Experimental results of the comparison of methods are presented. State of the art image descriptors (i.e. SIFT) have shown to be an excellent choice for addressing the data association problem. Nevertheless these kinds of descriptors are difficult to apply directly into a camera-based SLAM context. In the third part of this chapter, a simple but effective framework is presented in order to adapt state of the art image descriptors for their use in camera-based SLAM methods.

Chapter 6 describes the part III of the research corresponding to this thesis. In the first part of the chapter, an introduction to the monocular SLAM problematic and a small survive are presented. In this section the current challenges are pointed out. The particularities and differences of the monocular SLAM in relation to the general bearing-only SLAM are also explained. In the second part of this chapter, a novel approach for monocular SLAM is presented. This approach, called *Delayed Inverse Depth Monocular SLAM* method, is an extension of our general *Bearing-Only SLAM* algorithm (described in chapter 4) for being used in a camera-based SLAM context. Our approach aims to contribute to the robustness of Monocular SLAM systems. Several experiments using real data, captured with a low cost camera, are presented. A comparison with the *Un-delayed Inverse-Depth Feature Initialization* is also presented in order to demonstrate the performance of our proposed method. In the third part of this chapter, a novel framework, called Distributed Monocular SLAM, is proposed in order for addressing the problem of building and maintaining a global and consistent map of large environments at real time. The general idea is to adapt a monocular SLAM method (as our *Delayed Inverse Depth Monocular SLAM* method) as a complex real-time "Virtual Sensor". This Virtual Sensor provides appearance-based sensing in the form of features descriptors and emulates typical sensors as lasers for bearing and range sensing and also emulates encoders for odometry estimation. Afterward a classic SLAM method is plugged in (decoupled from the camera's frame rate) taking as its input the output of the Virtual Sensor. This process, called Global SLAM, models the drift of the Virtual Sensor estimations as independent uncertainty propagation, in order to estimate the global camera-robot pose and map. In our implementation, both estimation processes, the Virtual Sensor and the Global SLAM, run concurrently in different PCs in a local network, communicated with TCP/IP protocol. Experiments with real data are presented in order to show the performance of the method.

Chapter 7 summarizes the conclusions of this thesis and discusses possible avenues of future research.

Chapter 2

Antecedents

Robotic Localization and Mapping has been a highly active research area in robotics and AI for at least two decades. Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robot and localization addresses the problem of estimating the position of the mobile robot while it's moving through its environment. The Localization and mapping problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. Despite significant progress in this area, it still poses great challenges. At present, we have robust methods for mapping environments that are static, structured, and of limited size. Mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.

This chapter intends to present a general overview of the problematic addressed in this thesis. Several references to relevant work are included. For a serious background concerned to this dissertation, the reader is invited to study some of the articles referenced in this chapter.

First in section 2.1, a brief historical overview of the localization and mapping problem is given; here some of the earliest algorithms and taxonomies are cited.

In section 2.2, some of the aspects that make localization and mapping a very challenging problem, are introduced.

Finally in section 2.3, an overview of the Simultaneous Localization and Mapping problem (SLAM) is given. SLAM represents the research field where this thesis intends to make its most important contributions. This section is very relevant since gives an

introduction to several fundamentals topics addressed along the thesis. A general formulation of the SLAM problem is presented and also the most important approaches for the solution of the SLAM problem are introduced.

2.1 Historical Overview

Robotic mapping research has a long history. In the 1980s and early 1990s, the field of mapping was widely divided into metric and topological approaches. Metric maps capture the geometric properties of the environment, whereas topological maps describe the connectivity of different places.

An early representative of the metric approach was Elfes and Moravec's important *occupancy grid mapping algorithm* [2], [3], [4], which refers to a family of computer algorithms in probabilistic robotics for mobile robots which address the problem of generating maps from noisy and uncertain sensor measurement data, with the assumption that the robot Pose is known. The basic idea of the occupancy grid is to represent a map of the environment as an evenly spaced field of binary random variables each representing the presence of an obstacle at that location in the environment. Occupancy grid algorithms compute approximate posterior estimates for these random variables [5]. In others words, represents maps by fine-grained grids that model the occupied and free space of the environment. This approach has been used in a great number of robotic systems, such as [6], [7], [8], [9], [10], [11].

An alternative metric mapping algorithm was proposed by Chatila and Laumond [12], using sets of polyhedra to describe the geometry of environments.

Topological maps represent environments as a list of significant places that are connected via arcs. Arcs are usually annotated with information on how to navigate from one place to another. However, the distinction between metric and topological has always been fuzzy, since virtually all working topological approaches rely on geometric information. In practice, metric maps are finer grained than topological ones. Higher resolution comes at a computational price, but it helps to solve various hard problems, such as the correspondence problem discussed further below. Examples of topological are [13], [14], [15], [16], [17], [18], [19], [20], [21].

Historically, a second taxonomy of mapping algorithms is *world-centric* versus robot-centric. World-centric maps are represented in a global coordinate space. The entities in the map do not carry information about the sensor measurements that led to their discovery. Robot-centric maps, in contrast, are described in measurement space. They describe the sensor measurements a robot would receive at different locations. At first glance, robot-centric maps might appear easier to build, since no 'translation' of robot measurements into world coordinates is needed. However, robot-centric maps suffer two disadvantages. First, it is often difficult to extrapolate from individual

measurements to measurements at nearby, unexplored places an extrapolation that is typically straightforward in world-centric approaches. Put differently, there is usually no obvious geometry in measurement space that would allow for such extrapolation. Second, if different places look alike, robot-centric approaches often face difficulties to disambiguate them, again due to the lack of an obvious geometry in measurement space. For these reasons, the dominant approaches to date generate world-centric maps. An example of robot-centric approach is presented in [22], in this case the robot-centric approach can be useful because the main objective is navigating while the robot avoids obstacles in its path.

Since the 1990s, the field of robot mapping has been dominated by probabilistic techniques. A series of seminal papers by Smith, Self, and Cheeseman [23], [24] introduced a powerful statistical framework for simultaneously solving the mapping problem and the induced problem of localizing the robot relative to its growing map. Since then, robotic mapping has commonly been referred to as SLAM or CML, which is short for simultaneous localization and mapping [25], [26] and concurrent mapping and localization [27], [28] respectively. One family of probabilistic approaches employ Kalman filters to estimate the map and the robot location [29], [30], [31], [32], [33], [34]. The resulting maps usually describe the location of landmarks, or significant features in the environment, although recent extensions exist that represent environments by large numbers of raw range measurements [35]. An alternative family of algorithms [36], [19], [10], [28] is based on Dempster's expectation maximization algorithm [37], [38]. These approaches specifically address the correspondence problem in mapping, which is the problem of determining whether sensor measurement recorded at different points in time correspond to the same physical entity in the real world. A third family of probabilistic techniques seek to identify objects in the environment, which may correspond to ceilings, walls [39], [40] doors that might be open or closed of furniture and other objects that move. Many of this technique have counterparts in the computer vision and photogrammetry literature, [41], [42], [43], [44], [45] [46], [47], a connection that is currently been exploited.

2.2 The Robotic Localization and Mapping Problem

The problem of robotic mapping is that of acquiring a spatial model of a robot's environment. Maps are commonly used for robot navigation (e.g., localization) [48], [49]. To acquire a map, robots must possess sensors that enable it to perceive the outside world. Sensors commonly brought to bear for this task include cameras, range finders using sonar, laser, and infrared technology, radar, tactile sensors, compasses, and GPS. However, all these sensors are subject to errors, often referred to as measurement noise. More importantly, most robot sensors are subject to strict range limitations. For example,

light and sound cannot penetrate walls. These range limitations makes it necessary for a robot to navigate through its environment when building a map. The motion commands (controls) issued during environment exploration carry important information for building maps, since they convey information about the locations at which different sensor measurements were taken. Robot motion is also subject to errors, and the controls alone are therefore insufficient to determine a robot's pose (location and orientation) relative to its environment.

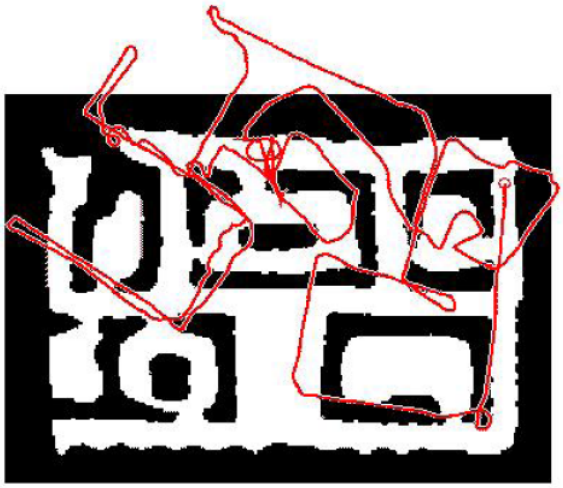


Figure 2.1 Example of odometry error: Shown here is a robot's path as obtained by its odometry, relative to a given map. Small odometry (or control) errors can have large effects on later position estimates.

2.2.1 Measurement Noise

A key challenge in robotic mapping arises from the nature of the measurement noise. Modeling problems, such as robotic mapping, are usually relatively easy to solve if the noise in different measurements is statistically *independent*. If this were the case, a robot could simply take more and more measurements to cancel out the effects of the noise. Unfortunately, in robotic mapping, the measurement errors are statistically dependent. This is because errors in control accumulate over time, and they affect the way future sensor measurements are interpreted. This is illustrated in Figure 2.1, which shows an example path of a mobile robot in a given map of the environment. As this example shows, a small rotational error on one end of a long corridor can lead to many meters of error on the other. As a result, whatever a robot infers about its environment is plagued by systematic, correlated errors. Accommodating such systematic errors is key to building maps successfully, and it is also a key complicating factor in robotic mapping.

Many existing mapping algorithms are therefore surprisingly complex, both from a mathematical and from an implementation point of view.

2.2.2 Dimensionality

The second complicating aspect of the robot mapping problem arises from the high dimensionality of the entities that are being mapped. To understand the dimensionality of the problem, the reader may consider how many numbers it may take to describe an environment like his/her own home. If one confines oneself to the description of major topological entities, such as corridors, intersections, rooms and doors, a few dozen numbers might suffice. A detailed two-dimensional floor plan, which is an equally common representation of robotic maps, often requires thousands of numbers. But a detailed 3D visual map of a building (or of an ocean floor) may easily require millions of numbers. From a statistical point of view, each such number is a dimension of the underlying estimation problem. Thus, the mapping problem can be extremely high dimensional.

2.2.3 Data Association

A third and possibly the hardest problem in robotic mapping is the *correspondence problem*, also known as the *data association problem*. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world. An instance of this problem is when a robot attempts to map a large cyclic environment. When closing the cycle, the robot has to find ou t where it is relative to its previously built map. This problem is complicated by the fact that at the time of cycle closing, the robot's accumulated pose error might be unboundedly large. The correspondence problem is difficult, since the number of possible hypotheses can grow exponentially over time. Most scientific progress on the correspondence problem has emerged in the past years, after a long period in which the problem was basically ignored in the robot community.

2.2.4 Dynamical Environments

Fourth, environments change over time. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time. Others are faster, such as the change of door status or the location of furniture items, such as chairs. Even faster may be the change of location of other agents in the environment, such as cars or people. The dynamism of robot environments creates a big challenge, since it adds yet another way in which seemingly inconsistent sensor measurements can be explained. To see, imagine a robot facing a closed door that previously was modeled as open. Such an

observation may be explained by two hypotheses, namely that the door status changed, or that the robot is not where it believes to be. Unfortunately, there are almost no mapping algorithms that can learn meaningful maps of dynamic environments. Instead, the predominant paradigm relies on a static world assumption, in which the robot is the only time-variant quantity (and everything else that moves is just noise). Consequently, most techniques are only applied in relatively short time windows, during which the respective environments are static.

2.2.5 Robotic Exploration

A fifth and final challenge arises from the fact that robots must choose their way during mapping. The task of generating robot motion in the pursuit of building a map is commonly referred to as *robotic exploration*. While optimal robot motion is relatively well-understood in fully modeled environments, exploring robots have to cope with partial and incomplete models. Hence, any viable exploration strategy has to be able to accommodate contingencies and surprises that might arise during map acquisition. For this reason, exploration is a challenging planning problem, which is often solved sub-optimally via simple heuristics.

When choosing where to move, various quantities have to be traded off: the expected gain in map information, the time and energy it takes to gain this information, the possible loss of pose information along the way, and so on. Furthermore, the underlying map estimation technique must be able to generate maps in real-time, which is an important restriction that rules out many existing approaches.

2.2.6 Localization and Mapping

As noted above, the literature refers to the mapping problem often in conjunction with the localization problem, which is the problem of determining a robot's pose. The reason for suggesting that both problems—the problem of estimating where things are in the environment and the problem of determining where a robot is—have to be solved in conjunction will become a bit more obvious below, when we state the basic statistical estimators that underlie all state-of-the-art techniques. In essence, both the robot localization and the map are uncertain, and by focusing just on one the other introduces *systematic* noise. Thus, estimating both at the same time has the pleasing property that both the measurement and the control noise are independent with regards to the properties that are being estimated (the state). Postponing further detail on this issue to further below, we notice that the robot mapping problem is like a chicken and egg problem:

If the robot's pose was known all along, building a map would be quite simple. Conversely, if we already had a map of the environment, there exist computationally

elegant and efficient algorithms for determining the robot's pose at any point in time [48], [50]. In combination, however, the problem is much harder.

Today, Simultaneous localization and mapping is largely considered the most difficult perceptual problem in robotics. Progress in robot mapping is bound to impact a much broader range of related perceptual problems, such as sensor based manipulation and interaction with people.

2.3 The SLAM Problem

The on-line robot estimation position from measurements of self-mapped features is a class of problem called, in the robotics community, as Simultaneous Localization and Mapping (SLAM) problem, which is one of the fundamental problems in robotics. SLAM consist in incrementally building a consistent map of the environment and, at the same time, localizing the position of the robot while explores its world. In SLAM is necessary to consider the uncertainties introduced into the map by potential noise sources in the robot motion and measurement process. The "solution" of the SLAM problem has been one of the notable successes of the robotics community over the past decade. SLAM has been formulated and solved as a theoretical problem in a number of different forms. SLAM has also been implemented in a number of different domains from indoor robots to outdoor, underwater, and airborne systems. At a theoretical and conceptual level, SLAM can now be considered a solved problem, reaching it a state of maturity sufficient to permit practical implementations in challenging environments. However, substantial issues remain in practically realizing more general SLAM solutions and notably in building and using perceptually rich maps as part of a SLAM algorithm.

Work by Smith and Cheesman [51] and Durrant-Whyte [52] established a statistical basis for describing relationships between landmarks and manipulating geometric uncertainty. A key element of this work was to show that there must be a high degree of correlation between estimates of the location of different landmarks in a map and that, indeed, these correlations would grow with successive observations.

At the same time Ayache and Faugeras [53] were undertaking early work in visual navigation, Crowley [54] and Chatila and Laumond [55] were working in sonar-based navigation of mobile robots using Kalman filter-type algorithms. These two strands of research had much in common and resulted soon after in the landmark paper by Smith et al. [23]. This paper showed that as a mobile robot moves through an unknown environment taking relative observations of landmarks, the estimates of these landmarks are all necessarily correlated with each other because of the common error in estimated vehicle location [56]. The implication of this was profound: A consistent full solution to the combined localization and mapping problem would require a joint state composed of the vehicle pose and every landmark position, to be updated following each landmark

observation. In turn, this would require the estimator to employ a huge state vector (on the order of the number of landmarks maintained in the map) with computation scaling as the square of the number of landmarks.

Crucially, this work did not look at the convergence properties of the map or its steady-state behavior. Indeed, it was widely assumed at the time that the estimated map errors would not converge and would instead exhibit a random-walk behavior with unbounded error growth. Thus, given the computational complexity of the mapping problem and without knowledge of the convergence behavior of the map, researchers instead focused on a series of approximations to the consistent mapping problem, which assumed or even forced the correlations between landmarks to be minimized or eliminated, so reducing the full filter to a series of decoupled landmark to vehicle filters ([56] and [57] for example). Also for these reasons, theoretical work on the combined localization and mapping problem came to a temporary halt, with work often focused on either mapping or localization as separate problems.

The conceptual breakthrough came with the realization that the combined mapping and localization problem, once formulated as a single estimation problem, was actually convergent. Most importantly, it was recognized that the correlations between landmarks, which most researchers had tried to minimize, were actually the critical part of the problem and that, on the contrary, the more these correlations grew, the better the solution. The structure of the SLAM problem, the convergence result and the coining of the acronym

SLAM was first presented in a mobile robotics survey paper presented at the 1995 International Symposium on Robotics Research [52]. The essential theory on convergence and many of the initial results were developed by Csorba [30], [58]. Several groups already working on mapping and localization, notably at the Massachusetts Institute of Technology [59], Zaragoza [60], [61], the ACFR at Sydney [62], [34], and others [63], [64], [28], [65], [66], [67], [68] began working in earnest on SLAM–also called *concurrent mapping and localization* (CML) at this time in indoor, outdoor, and subsea environments.

2.3.1 Formulation of the SLAM Problem

SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location.

Consider a mobile robot moving through an environment taking relative observations of a number of unknown landmarks using a sensor located on the robot as shown in Figure 2.2. At a time instant k, the following quantities are defined:

$\hat{\mathbf{x}}_k$: the state vector describing the location and orientation of the vehicle.

u_k: the control vector, applied at time k-1 to drive the vehicle to a state $\hat{x}_k$ at time k.

m_i: a vector describing the location of the *i-th* landmark whose true location is assumed time invariant.

z_{ik}: an observation taken from the vehicle of the location of the *i-th* landmark at time k. When there are multiple landmark observations at any one time or when the specific landmark is not relevant to the discussion, the observation will be written simply as z_k.

In addition, the following sets are also defined:

- $X_{0:k} = \{ \hat{x}_0, \hat{x}_1, \cdots, \hat{x}_k \} = \{ X_{0:k-1}, \hat{x}_k \}$: the history of vehicle locations.
- $U_{0:k} = \{u_1, u_2, \cdots, u_k\} = \{U_{0:k-1}, u_k\}$: the history of control inputs.
- $M = \{m_1, m_2, \cdots, m_n\}$: the set of all landmarks.
- $Z_{0:k} = \{z_1, z_2, \cdots, z_k\} = \{Z_{0:k-1}, z_k\}$: the set of all landmark observation.

2.3.2 Probabilistic SLAM

In probabilistic form, the simultaneous localization and map building (SLAM) problem requires that the probability distribution

$$P(\hat{x}_k, M \mid Z_{0:k}, U_{0:k}, \hat{x}_0) \tag{1.1}$$

be computed for all times k. This probability distribution describes the *joint* posterior density of the landmark locations and vehicle state (at time k) given the recorded observations and control inputs up to and including time k together with the initial state of the vehicle. In general, a recursive solution to the SLAM problem is desirable. Starting with an estimate for the distribution $P(\hat{x}_{k-1}, M \mid Z_{0:k-1}, U_{0:k-1})$ at time $k-1$, the joint posterior, following a control u_k and observation z_k, is computed using Bayes theorem. This computation requires that a state transition model and an observation model are defined describing the effect of the control input and observation respectively.

The *observation model* describes the probability of making an observation z_k when the vehicle location and landmark locations are known and is generally described in the form

$$P(z_k \mid \hat{x}_k, M) \tag{1.2}$$

It is reasonable to assume that once the vehicle location and map are defined, observations are conditionally independent given the map and the current vehicle state.

The *motion model* for the vehicle can be described in terms of a probability distribution on state transitions in the form

$$P(\hat{x}_k \mid \hat{x}_{k-1}, u_k) \tag{1.3}$$

That is, the state transition is assumed to be a Markov process in which the next state $\hat{\mathrm{x}}_k$ depends only on the immediately preceding state $\hat{\mathrm{x}}_{k-1}$and the applied control u_k and is independent of both the observations and the map.

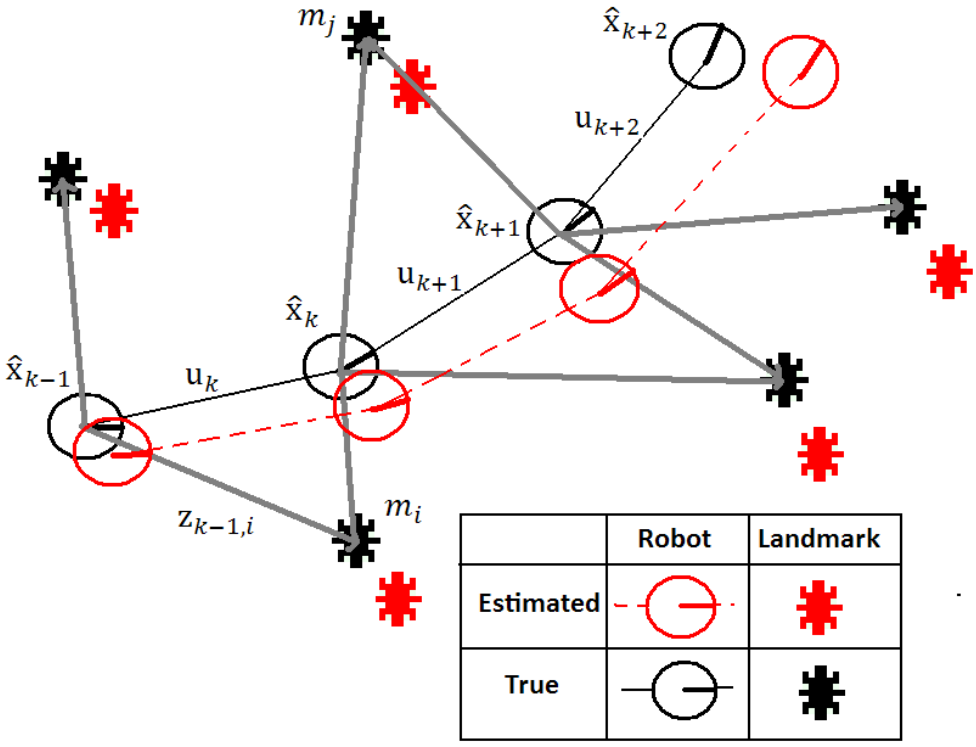


Figure 2.2 The essential SLAM problem. A simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations.

The SLAM algorithm is now implemented in a standard two-step recursive (sequential) prediction (time-update) correction (measurement-update) form:

Time-update

$$P(\hat{\mathrm{x}}_k, \mathrm{M}|\, \mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k}, \hat{\mathrm{x}}_0) = \int P(\hat{\mathrm{x}}_k|\hat{\mathrm{x}}_{k-1}, \mathrm{u}_k) \times P(\hat{\mathrm{x}}_{k-1}, \mathrm{M}|\, \mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k-1}, \hat{\mathrm{x}}_0)\mathrm{d}\hat{\mathrm{x}}_{k-1} \quad (1.4)$$

Measurement Update

$$P(\hat{\mathrm{x}}_k, \mathrm{M} \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \hat{\mathrm{x}}_0) = \frac{P(\mathrm{z}_k \mid \hat{\mathrm{x}}_k, \mathrm{M})\, P(\hat{\mathrm{x}}_k, \mathrm{M} \mid \mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k}, \hat{\mathrm{x}}_0)}{P(\mathrm{z}_k \mid \mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k})} \tag{1.5}$$

Equations (1.4) and (1.5) provide a recursive procedure for calculating the joint posterior $P(\hat{\mathrm{x}}_k, \mathrm{M} \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{x}_0)$ for the robot state $\hat{\mathrm{x}}_k$ and map M at a time k based on all observations $\mathrm{Z}_{0:k}$ and all control inputs $\mathrm{U}_{0:k}$ up to and including time k. The recursion is a function of a vehicle model $P(\hat{\mathrm{x}}_k \mid \hat{\mathrm{x}}_{k-1}, \mathrm{u}_k)$ and an observation model $P(\mathrm{z}_k \mid \hat{\mathrm{x}}_k, \mathrm{M})$.

It is worth noting that the map building problem may be formulated as computing the conditional density $P(\mathrm{M} \mid \mathrm{X}_{k-1}, \mathrm{Z}_{0:k}, \mathrm{U}_{0:k})$. This assumes that the location of the vehicle $\hat{\mathrm{x}}_k$ is known (or at least deterministic) at all times, subject to knowledge of initial location. A map M is then constructed by fusing observations from different locations. Conversely, the localization problem may be formulated as computing the probability distribution $P(\hat{\mathrm{x}}_k \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{M})$. This assumes that the landmark locations are known with certainty, and the objective is to compute an estimate of vehicle location with respect to these landmarks.

2.3.3 Structure of Probabilistic SLAM

The observation model $P(\mathrm{z}_k \mid \hat{\mathrm{x}}_k, \mathrm{M})$ makes explicit the dependence of observations on both the vehicle and landmark locations. It follows that the joint posterior cannot be partitioned in the obvious manner

$$P(\hat{\mathrm{x}}_k, \mathrm{M} \mid \mathrm{z}_k) \neq P(\hat{\mathrm{x}}_k \mid \mathrm{z}_k) \mathrm{P}(\mathrm{M} \mid \mathrm{z}_k) \tag{1.6}$$

and indeed it is well known from the early papers on consistent mapping [52], [24] that a partition such as this leads to inconsistent estimates. However, the SLAM problem has more structure than is immediately obvious from these equations.

Referring again to Figure 2.2, it can be seen that much of the error between estimated and true landmark locations is common between landmarks and is in fact due to a single source; errors in knowledge of where the robot is when landmark observations are made. In turn, this implies that the errors in landmark location estimates are highly correlated. Practically, this means that the relative location between any two landmarks, $\mathrm{m}_i - \mathrm{m}_j$, may be known with high accuracy, even when the absolute location of a landmark m_i is quite uncertain. In probabilistic form, this means that the joint probability density for the pair of landmarks $P(\mathrm{m}_i, \mathrm{m}_j)$ is highly peaked even when the marginal densities $P(\mathrm{m}_i)$ may be quite dispersed.

The most important insight in SLAM was to realize that the correlations between landmark estimates increase *monotonically* as more and more observations are made. (These results have only been proved for the linear Gaussian case [31]. Formal proof for the more general probabilistic case remains an open problem.) Practically, this means that knowledge of the relative location of landmarks always improves and never diverges,

regardless of robot motion. In probabilistic terms, this means that the joint probability density on all landmarks $P(\mathrm{M})$ becomes monotonically more peaked as more observations are made.

This convergence occurs because the observations made by the robot can be considered as "nearly independent" measurements of the *relative* location between landmarks. Referring again to Figure 2.2, consider the robot at location $\hat{\mathrm{x}}_k$ observing the two landmarks m_i and m_j. The relative location of observed landmarks is clearly independent of the coordinate frame of the vehicle, and successive observations from this fixed location would yield further independent measurements of the relative relationship between landmarks.

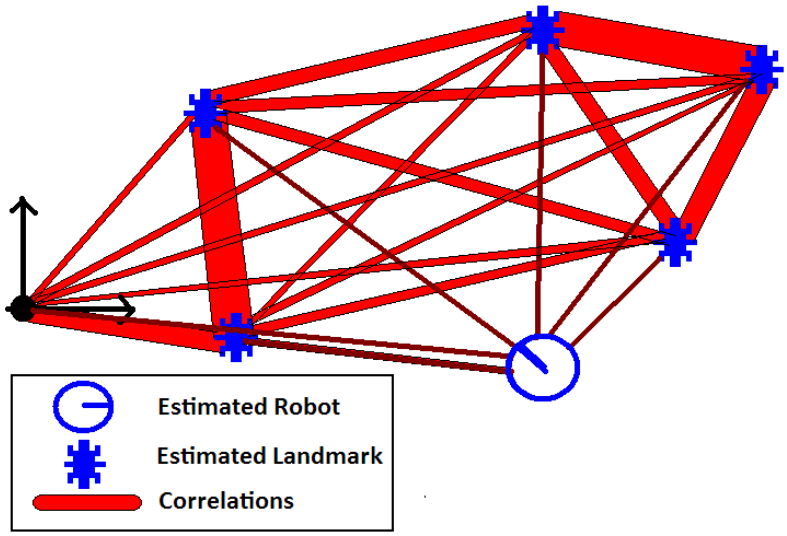


Figure 2.3 Spring network analogy. The landmarks are connected by springs describing correlations between landmarks. As the vehicle moves back and forth through the environment, spring stiffness or correlations increase (red links become thicker). As landmarks are observed and estimated locations are corrected, these changes are propagated through the spring network. Note, the robot itself is correlated to the map.

Now, as the robot moves to location $\hat{\mathrm{x}}_{k+1}$, it again observes landmark m_j this allows the estimated location of the robot and landmark to be updated relative to the previous location $\hat{\mathrm{x}}_k$. In turn, this propagates back to update landmark m_i —even though this landmark is not seen from the new location. This occurs because the two landmarks are highly correlated (their relative location is well known) from previous measurements. Further, the fact that the same measurement data is used to update these two landmarks

makes them *more* correlated. The term *nearly independent measurement* is appropriate because the observation errors will be correlated through successive vehicle motions. Also note that in Figure 2.2 at location $\hat{\mathrm{x}}_{k+1}$, the robot observes two new landmarks *relative* to m_j. These new landmarks are thus immediately linked or correlated to the rest of the map. Later updates to these landmarks will also update landmark m_j and through this landmark m_i and so on. That is, all landmarks end up forming a network linked by relative location or correlations whose precision or value increases whenever an observation is made.

This process can be visualized (Figure 2.3) as a network of springs connecting all landmarks together or as a rubber sheet in which all landmarks are embedded. An observation in a neighborhood acts like a displacement to a spring system or rubber sheet such that its effect is great in the neighborhood and, dependent on local stiffness (correlation) properties, diminishes with distance to other landmarks. As the robot moves through this environment and takes observations of the landmarks, the springs become increasingly (and monotonically) stiffer. In the limit, a rigid map of landmarks or an accurate *relative* map of the environment is obtained. As the map is built, the location accuracy of the robot measured relative to the map is bounded only by the quality of the map and relative measurement sensor. In the theoretical limit, robot relative location accuracy becomes equal to the localization accuracy achievable with a given map.

2.3.4 Solutions to the SLAM Problems

Solutions to the probabilistic SLAM problem involve finding an appropriate representation for both the observation model (1.2) and motion model (1.3) that allows efficient and consistent computation of the prior and posterior distributions in (1.4) and (1.5). By far, the most common representation is in the form of a state-space model with additive Gaussian noise, leading to the use of the extended Kalman filter (EKF) to solve the SLAM problem. One important alternative representation is to describe the vehicle motion model in (1.3) as a set of samples of a more general non-Gaussian probability distribution. This leads to the use of the Rao-Blackwellized particle filter, or FastSLAM algorithm, to solve the SLAM problem. While EKF-SLAM and FastSLAM are the two most important solution methods, newer alternatives, which offer much potential, have been proposed, including the use of the information-state form [69].

2.3.5 Extended Kalman Filter based SLAM

The Extended Kalman filter (EKF) is an efficient recursive filter which estimates the state of dynamic system from a series of loud and incomplete measurements. In that sense, the most common probabilistic approach to solving the problem of SLAM is to use EKF. It is frequently used to solve the SLAM problem, principally due to its theoretical

clarity and its facility of implementation. There are some issues related to EKF based SLAM solutions:

Convergence: In the EKF-SLAM problem, convergence of the map is manifest in the monotonic convergence of the determinant of the map covariance matrix $\mathrm{P}_{mm,k}$ and all landmark pair submatrices, toward zero. The individual landmark variances converge toward a lower bound determined by initial uncertainties in robot position and observations.

Computational Effort: The observation update step requires that all landmarks and the joint covariance matrix be updated every time an observation is made. Naively, this means computation grows quadratically with the number of landmarks. There has been a great deal of work undertaken in developing efficient variants of the EKF-SLAM solution and real-time implementations with many thousands of landmarks have been demonstrated [70], [59].

Data Association: The standard formulation of the EKF-SLAM solution is especially fragile to incorrect association of observations to landmarks [71]. The loop-closure problem, when a robot returns to reobserve landmarks after a large traverse, is especially difficult. The association problem is compounded in environments where landmarks are not simple points and indeed look different from different viewpoints.

Nonlinearity: EKF-SLAM employs linearized models of nonlinear motion and observation models and so inherits many caveats. Nonlinearity can be a significant problem in EKF-SLAM and leads to inevitable and sometimes dramatic, inconsistency in solutions [72]. Convergence and consistency can only be guaranteed in the linear case.

2.3.6 Rao-Blackwellized Filter

The FastSLAM algorithm, introduced by Montemerlo et al. [73], marked a fundamental conceptual shift in the design of recursive probabilistic SLAM. Previous efforts focused on improving the performance of EKF-SLAM, while retaining its essential linear Gaussian assumptions. FastSLAM, with its basis in recursive Monte Carlo sampling, or particle filtering, was the first to directly represent the nonlinear process model and non-Gaussian pose distribution. (FastSLAM still linearizes the observation model, but this is typically a reasonable approximation for range-bearing measurements when the vehicle pose is known.) This approach was influenced by earlier probabilistic mapping experiments of Murphy [74] and Thrun [75].

The high dimensional state-space of the SLAM problem makes direct application of particle filters computationally infeasible. However, it is possible to reduce the samplespace by applying Rao-Blackwellization.

Statistically, FastSLAM algorithm suffers degeneration due to its inability to forget the past. Marginalizing the map introduces dependence on the pose and measurement history, and so, when resampling depletes this history, statistical accuracy is lost [76].

Nevertheless, empirical results of Fast-SLAM in real outdoor environments [77] show that the algorithm is capable of generating an accurate map in practice.

2.4 Conclusions

The Localization and mapping problem is generally regarded as ones of the most important problems in the pursuit of building truly autonomous mobile robots. The goal of this chapter was to survey several concepts, in order to gives to the reader a background for the main themes addressed in this thesis. Several references to most of the relevant work on the field of localization and mapping are included. It is quite remarkable that virtually all state-of-the-art algorithms in robotic mapping share the same mathematical foundation: They are all probabilistic. Moreover, they are all versions of Bayes filters and an underlying generative probabilistic description of robot motion and perception. This development parallels a much broader trend in mobile robotics, where probabilistic techniques are commonly the method of choice over more ad hoc approaches, such as behavior-based techniques.

Overall, the situation in robot localization and mapping is encouraging. In that sense, it was seen that: *Measurement Noise*, *Dimensionality*, *Data Association*, *Dynamical Environments* and *Robotics Exploration* represent ones of the fundamentals issues for taken into account, in order to design robust methods well suited for complex, large and dynamical environments. If truly autonomous mobile robots wants to be built, the localization and mapping problems have to be solved simultaneously, in that sense SLAM is largely considered the most difficult and important perceptual problem in robotics.

In section 2.3, the general SLAM problem was formulated in a probabilistic context. In that sense, particle filters based methods and Kalman filters based methods represents both the major families of algorithms in order to solve the SLAM problem.

Chapter 3

Kalman Filter based SLAM

Extended Kalman Filter techniques (EKF) are frequently used to solve the SLAM problem, principally due to its theoretical clarity and its facility for implementing it. On the other hand, possibly its main weakness resides in the assumption of representing uncertainties with a single Gaussian distribution. In many real ones situations is not possible to model uncertainties in this way, this fact has motivated the use of alternative estimations techniques like the Particles Filters, capable of representing uncertainties with multivariate distributions. Nevertheless it's seen that the SLAM schemes based in Particles Filters have not finished consolidated, and more over, in the literature, continue to emerge new approaches based on the Kalman Filter.

This chapter aims to expose a considerable amount of the theoretical and mathematical basis that contends to this thesis. In that sense, the basic components of a general SLAM system are exposed in a manner like a "SLAM tutorial". Several equations that will be used recurrently along the thesis are given in their simplest form.

First the basic equations (prediction and update) of the Kalman Filter are exposed. Later a simple differential model for a vehicle is introduced in order to illustrate the effect of the uncertainty propagation in this kind of systems. At this point, the convenience of using external references (landmarks), in order to minimize the drift in the estimation of the robot's location, is analyzed. The convenience of a probabilistic framework in SLAM is also pointed out. Later the localization and mapping problem are exposed in a separated manner. For the localization case, it is assumed that the robot knows perfectly the locations of a group of landmarks but it has uncertainty about its location, in that sense it's seen how the measurement to external environment reference improves the estimation of the robot's location and how when the sensor is "turned off" the robot begin to get lost. Later we assume the opposite in the mapping case, where the robot's location is perfectly know over the time, and the robot must estimate a features

map of its environment. In this case it's seen how the uncertainty in the feature's location is minimized over time while the robot is moving.

Both, Mapping and Localization problem, where the counterpart are know, (in Localization problem we know the map, in Mapping problem we know the localization) are simple comparing with the Simultaneous Localization and Mapping problem where both, map and robot's locations have to be estimated simultaneously. In the last section we put all the pieces together in order to describe the general EKF-based SLAM algorithm.

A considerable part of this chapter is based in a series of excellent SLAM tutorial like [78], [79] or [33]. All the simulations presented in this chapter are implemented in MATLAB, and based in the public code provided by the author of [33].

3.1 Kalman Filtering

The Kalman Filter is a general statistical tool for the analysis of time-varying physical systems in the presence of noise. A system is modeled by a state vector x which has entries for each of the quantities of interest. The passage of time is divided into small intervals Δt, and knowledge of the expected evolution of the sate vector is encapsulated in the state transition function f.

The filter permits continuous and efficient estimation of the state as the system evolves, incorporating the information provided by any measurements z which is made of quantities depending on the sate. The current state estimate is stored in the vector $\hat{\mathrm{x}}$ and the covariance matrix P, which is square and symmetric with dimension of the number of elements in x, represents the uncertainty in $\hat{\mathrm{x}}$. If the dependence of both f and the measurement function h on x is linear, and the statistical distribution of noise in the state transition and measurements is Gaussian, then the solution produced is optimal.

3.1.1 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is a simple extension of the Kalman Filter to cope with systems whose state transition and measurement functions are non-linear, or whose noise distributions are non-Gaussian. This is true of most physical systems to which the filter has been applied, and acts only as an approximation in these cases, where the downside of its efficiency is oversimplification of the mathematical forms of the functions and distributions involved. Usually the results are quite adequate, but in some recent vision applications, such as tracking the outlines of rapidly moving objects against cluttered backgrounds, the Kalman Filter has been found to fail (in this case due to its inability to use multi-modal distributions to represent multiple hypotheses about the location of an object)and replacement algorithms have been develop [80]. In our systems, the EKF has been found to work well.

3.1.2 Prediction

In a step time during which no measurements are made, our estimate of the state of a system changes according to the state transition function f describing the dynamics. The covariance matrix changes, reflecting the increase of uncertainty in the state, due to noise Q presented in the state transition. (due to random effects or factors which are not accounted for in the dynamic model). Both f and Q depend on u, the current control vector, for example in a vehicle, specifying demanded velocity and steering angle. The label k denotes an incrementing time step.

$$\hat{\mathrm{x}}(k+1) = f(\hat{\mathrm{x}}(k), u(k)) \tag{3.1}$$

$$P(k+1) = \frac{\partial f}{\partial x} P(k) \frac{\partial f}{\partial x}^{\mathsf{T}} + \frac{\partial f}{\partial u} Q \frac{\partial f}{\partial u}^{\mathsf{T}} \tag{3.2}$$

The trick behind the EKF is to linearize the non-linear models around the "best" current estimate (best meaning prediction ($k+1$) or last estimate (k). This is done using a Taylor series expansion.

The term $\partial f/\partial x$ is understood to be the Jacobian of f with respect to x evaluated at an elsewhere specified point:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \tag{3.3}$$

3.1.3 Update

When a measurement is made, the state estimate improves with the new information and the uncertainty represented by P will reduce. v is the difference between the actual measurement z and the prediction h calculated from the current state, and is called the innovation. R is the covariance matrix of the noise in the measurement. The update equations for state and covariance are:

$$\hat{\mathrm{x}}(k) = \hat{\mathrm{x}}(k+1) + Wv(k) \tag{3.4}$$

$$P(k) = P(k+1) - WSW^{\mathsf{T}} \tag{3.5}$$

where the innovation is

$$v(k) = z(k) - h(x(k+1)) \tag{3.6}$$

W, the Kalman gain, is

$$W = P(k+1)\frac{\partial h^{\mathsf{T}}}{\partial x}\, S^{-1} \tag{3.7}$$

and S, the innovation covariance is

$$S = \frac{\partial h}{\partial x} P(k+1) \frac{\partial h^{\mathsf{T}}}{\partial x} + R \tag{3.8}$$

The innovation covariance represents the uncertainty in v, the amount by which a true measurement differs from its predicted value. Where $\partial h/\partial x$ is the Jacobian of the prediction equation h with respect to the state x:

$$\frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_m} \end{bmatrix} \tag{3.9}$$

3.2 Vehicle Models and Odometry

A simple differential drive model for a mobile robot is introduced and an analysis of the uncertainty propagation with this model is presented.

3.2.1 Differential Drive Model

A differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion. If both the wheels are driven in the same direction and speed, the robot will go in a straight line. Otherwise, depending on the speed of rotation and its direction, the centre of rotation may fall anywhere in the line joining the two wheels. Since the direction of the robot is dependent on the rate and direction of rotation of the two driven wheels, these quantities should be sensed and controlled precisely. This usually creates some problem. If both wheels are turned with equal speed in opposite directions, the robot will rotate about the central point of the axis.

A differentially steered robot is similar to the differential gears used in automobiles in that both the wheels can have different rates of rotations, but unlike the differential gearing system; a differentially steered system will have both the wheels powered. Differential wheeled robots are used extensively in robotics, since their motion is easy to program and can be well controlled. Virtually all consumer robots on the market today use differential steering primarily for its low cost and ultra-simplicity.

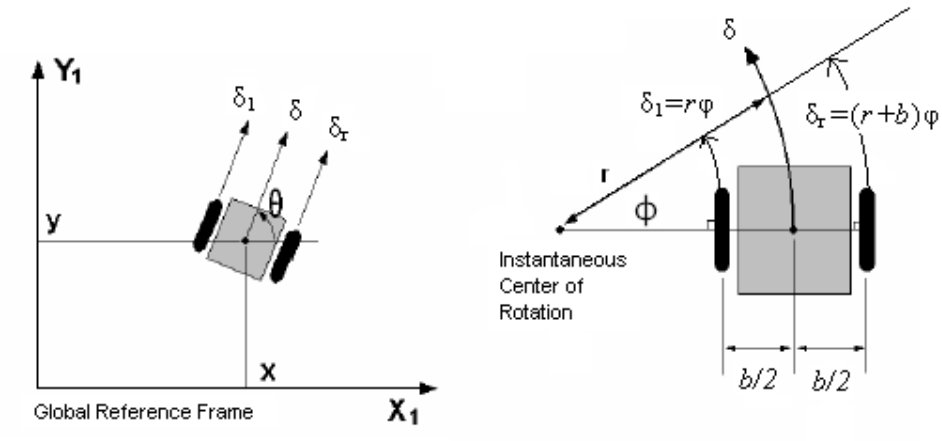


Figure 3.1 Vehicle with differential steering

In Figure 3.1 (right) is possible to see a schematic model of traction differential while turning, and what the parameters are that affect to turn and displacement. We are interesting in a model of dead reckoning; in traditional aviation or nautical navigation, the term "dead reckon" means to estimate position without external references. Position is dead reckoned using knowledge about a vehicle's course and speed over a period of time. In robotics, the necessary information is often obtained by measuring wheel revolutions. Devices called encoders are coupled to a robot's drive wheels and act like digital odometers. Although things can go wrong (as when the robot "spins out" on a slippery floor), encoders generally provide a good estimate of displacements δ_l δ_r for the left and right wheels, respectively. These displacement values can, in turn, be used to determine global (x,y,θ) position based on odometry calculations Figure 3.1 (left).

The equations bellow, taken from [81] estimates the current global vehicle pose (x,y,θ) (position and orientation) as a function of the wheels velocities V_l V_r, the axes separation b, the time t, and the initial robot position (x_0,y_0,θ_0).

$$[x, y, \theta] = f(V_l, V_r, t, b, x_0, y_0, \theta_0) \tag{3.10}$$

$$\theta(t) = \frac{(V_l - V_r)t}{b} + \theta_0 \tag{3.11}$$

$$x(t) = x_0 + \frac{b(V_l + V_r)}{2(V_l - V_r)}\left[\sin\left(\frac{(V_l - V_r)t}{b} + \theta_0\right) - \sin\theta_0\right] \tag{3.12}$$

$$y(t) = y_0 + \frac{b(V_l + V_r)}{2(V_l - V_r)}\left[\cos\left(\frac{(V_l - V_r)t}{b} + \theta_0\right) - \cos\theta_0\right] \tag{3.13}$$

Nevertheless many popular authors on robotics as [82], recommend the formulas shown in below as a way to avoid the complications of equations (3.11), (3.12) and (3.13). The methods in (3.15), (3.16) and (3.17) are especially well suited to small robot applications where on-board computing power is limited (and floating-point operations might not be available).

$$\delta = (\delta_l + \delta_r)/2 \tag{3.14}$$

$$\theta = \frac{\delta_l - \delta_r}{b} + \theta_0 \tag{3.15}$$

$$x = \delta \cos\theta + x_0 \tag{3.16}$$

$$y = \delta \sin\theta + y_0 \tag{3.17}$$

Where the current global vehicle pose (x,y,θ) is in function of the distance traveled for each wheel δ_l, δ_r and the axes separation b. The formulas above are simple and convenient. They work well for algorithms as long it is remember that they are approximations. Essentially computes the robot's total rotation and distance traveled, and then treats it as if it had completed the full rotation as a pivot the very beginning of the maneuver and then traveled in a straight line. Of course, this kind of "point-and-shoot" description is an incomplete representation of the reality. But as long as the robot's actual path doesn't involve too much of a turn, works fine. In this thesis, these equations are used.

3.2.2 Evolution of Uncertainty

We allow the vehicle to move on a 2D surface (a floor) and point in arbitrary directions. We parameterize the vehicle pose $\hat{\mathrm{x}}_v$ (the joint of position and orientation) as

$$\hat{\mathrm{x}}_v = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix} \tag{3.18}$$

Then the robot discrete motion prediction model, using the equations of section 3.2.1, is:

$$\hat{\mathrm{x}}_{v(k+1)} = f\left(\hat{\mathrm{x}}_{v(k)}, u(k)\right); u(k) = \begin{bmatrix} \delta_l \\ \delta_r \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} x_{v(k+1)} \\ y_{v(k+1)} \\ \theta_{v(k+1)} \end{bmatrix} = \begin{bmatrix} x_{v(k)} + \delta\cos\theta' \\ y_{v(k)} + \delta\sin\theta' \\ \theta_{v(k)} + \theta' \end{bmatrix} \tag{3.20}$$

being δ the distance traveled by the robot's center and θ' the turn realized by the robot at each instant k:

$$\delta = \frac{\delta_l + \delta_r}{2}, \theta' = \frac{\delta_l - \delta_r}{b} \tag{3.21}$$

and derived from the distance traveled for each robot's wheels δ_l, δ_r and the separation b between them. Usually δ_l, δ_r are obtained from the robot's encoders.

Note that equation (3.20) is a model of a perfect, noiseless vehicle. Clearly this is a little unrealistic; therefore we need to model uncertainty. One popular way to do this is to insert noise terms into the control signal u such that:

$$u(k) = u_n(k) + v(k) \tag{3.22}$$

where $u_n(k)$ is a nominal (intended) control signal and $v(k)$ is a zero mean Gaussian distributed noise vector:

$$v(k) \sim \aleph(0, \begin{bmatrix} \sigma_{\delta_l}^2 & 0 \\ 0 & \sigma_{\delta_r}^2 \end{bmatrix}) \tag{3.23}$$

$$u(k) \sim \aleph(u_n(k), \begin{bmatrix} \sigma_{\delta_l}^2 & 0 \\ 0 & \sigma_{\delta_r}^2 \end{bmatrix}) \tag{3.24}$$

This completes a simple probabilistic model of a differential drive vehicle. The propagation of this model affects uncertainty in vehicle pose over time. If the robots moves when only the control signal u is available then an initial uncertainty in vehicle pose increases over time.

The differential drive model (3.20) is non-linear and so we will have to use the non-linear form of the predicted Kalman step.

Assume at time k we have been given a previous best estimate of the vehicle pose $\hat{x}_v(k\text{-}1|k\text{-}1)$ and an associated covariance $P_v(k\text{-}1|k\text{-}1)$. Equations (3.1) and (3.2) have that:

$$\hat{x}_v(k+1) = f(\hat{x}(k), u(k)) \tag{3.25}$$

$$P(k+1) = \nabla F_x P(k) \nabla F_x^{\top} + \nabla F_v Q \nabla F_v^{\top} \tag{3.26}$$

For clarity, in equation (3.26) the notation for the jacobian has been change from $\partial f/\partial x$ to ∇F_x and $\partial f/\partial u$ to ∇F_v, also note that we are considering a constant noise Q instead of varying noise $Q(k)$. In this case:

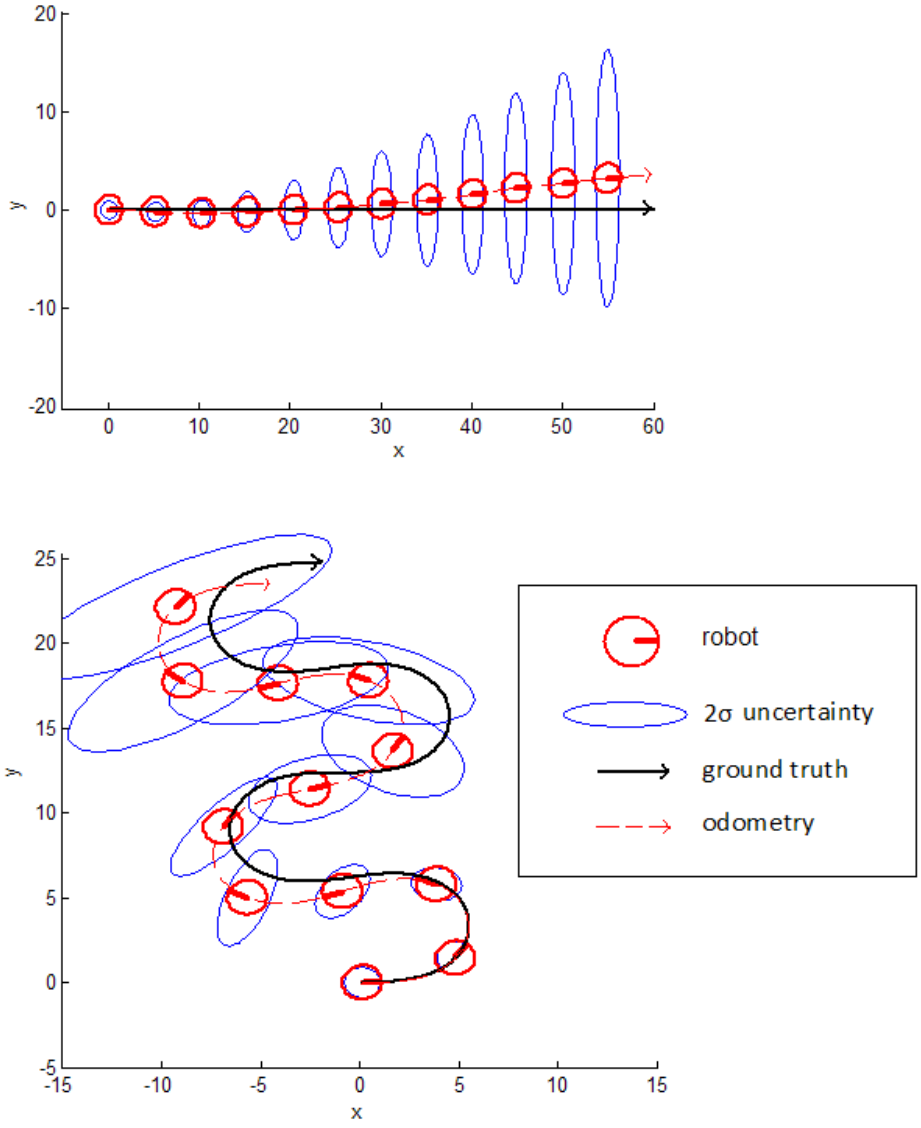


Figure 3.2 Simulation of the uncertainty evolution in a differential drive robot for two paths; a simple straight trajectory (Upper graphic) and a sinus trajectory (Lower graphic). Robot position 2σ uncertainty is illustrated by the ellipses. Note the divergence between true and dead-reckoned (odometry) estimations. This is typical of all dead reckoning methods; the only thing that can be changed is the rate of divergence.

$$Q = \begin{bmatrix} \sigma_{\delta_l}^2 & 0 \\ 0 & \sigma_{\delta_r}^2 \end{bmatrix} \tag{3.27}$$

The Jacobians of (3.20) with respect to state and control noise at $\hat{x}(k\text{-}1|k\text{-}1)$ are:

$$\nabla F_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.28}$$

$$\nabla F_u = \begin{bmatrix} 2\cos\theta' - \dfrac{\delta \sin\theta'}{b} & 2\cos\theta' + \dfrac{\delta \sin\theta'}{b} \\ 2\sin\theta' + \dfrac{\delta \cos\theta'}{b} & 2\sin\theta' - \dfrac{\delta \cos\theta'}{b} \\ 1/_b & -1/_b \end{bmatrix} \tag{3.29}$$

Figure 3.2 shows the results of iterating equations (3.25) and (3.26). The uncertainty injected into the system via the noisy encoders measurement makes the estimated covariance of the vehicle grows without bound. In actual real life the real robot is integrating the noisy control signal (odometry measurements). The true trajectory will therefore always drift away from the trajectory estimated by the algorithms running by the robot. Of course it is expected a gross accumulation of error as the time spent moving "open loop" increases. All the measurements such as the encoders lectures are measured in the vehicle frame and must be integrated, along with the noise on the measurements. This always leads to what is called "dead reckoning drift". The main cause of this divergence on land vehicles is wheel slip. Typically robot wheels are fitted with encoders that measure the rotation of each wheel. Position is then an integral-function of these "wheel counts". The problem is a wheel or radius *r* may have turned through θ but due to slip/skid the distance travelled over the ground is only $(1\text{-}n)r\theta$ where n is an unobservable slip parameter.

Some methods for odometry calibration, as the proposed in [83], [84] and [85] can considerably reduce the rate in dead reckoning drift, nevertheless, always there is going to be some moment where the drift will reach an unsustainable level.

3.3 Feature Based Mapping and Localization

If only internal measurements (odometry) are used in order to estimate the global position of the robot then an increasing drift will be present in the estimations while the robot moves. In order to minimize the drift, the robot can use others sensors in order to incorporate major information to the system. In that sense the more popular

approach is to measure (using diverse kind of sensors) the relative position, respect to the robot, of static and easy-detectable spatial locations of the environment, this “landmarks” are called map *features.* In this case if there is a previous knowledge of the features global position, then the robot can be localized in the features global frame.

3.3.1 Map Representation

A continuous-valued map is one method for exact decomposition of the environment. The position of environmental features can be annotated precisely in continuous space. Mobile robot implementations to date use continuous maps only in 2D representations, higher dimensionality can result in computational explosion. In chapter 2, it was seen that one of the first representations of continuous-valued (and metric) maps was the occupancy grid algorithm. This kind of representation has some advantages; facilitate the planning of navigation paths using the free space on the map. Make easy some Bayesian sensory fusion approaches. When the probability of occupation is upgraded along the time, a degree of dynamic adaptation in the map is archived. And is a kind of representation easy to understand by humans. On the other hand the maps based on grids of occupation suffer from some drawbacks; the size of the map in memory of the robot grows to the size of the environment, and when a small cell size is used, this growth in requirements is becoming untenable. In that sense, the occupancy grids maps are not compatible with the compactness that provides the assumption of a "closed world" (closed-world-assumption).

This means that one assumes that the representation will specify all environmental objects in the map, and that any area in the map that is devoid of objects has no objects in the corresponding portion of the environment. Thus, the total storage needed in the map is proportional to the density of objects in the environment, and a sparse environment can be represented by a low-memory map.

In contrast maps of the occupation have to reserve memory for each cell in the grid. In addition, any method of decomposition imposes grid geometry in the world, rejecting a priori details of the environment. This may be inappropriate when the geometry is the more outstanding feature of the environment as may be the case of less structured exterior (outdoor) spaces.

A common approach is to combine the exactness of a continuous representation with the compactness of the closed-world assumption. One example of such a representation is a 2D representation in which polygons represent all obstacles in a continuous-valued coordinate space. This is similar to the method used by Latombe [86], [87] and others to represent environments for mobile robot path-planning techniques.

In the case of [[86], [87]] most of the experiments are in fact simulations run exclusively within the computer’s memory. Therefore, no real effort would have been

expended to attempt to use sets of polygons to describe a real-world environment, such as a park or office building.

In other work in which real environments must be captured by the maps, one sees a trend toward selectivity and abstraction. The human map maker tends to capture on the map, for localization purposes, only objects that can be detected by the robot's sensors and, furthermore, only a subset of the features of real-world objects.

It should be immediately apparent that geometric maps can capably represent the physical locations of objects without referring to their texture, color, elasticity, or any other such secondary features that do not relate directly to position and space. In addition to this level of simplification, a mobile robot map can further reduce memory usage by capturing only aspects of object geometry that are immediately relevant to localization. For example, all objects may be approximated using very simple convex polygons, sacrificing map felicity for the sake of computational speed.

An approach which benefits of an accurate metric map representing the environment on a continuous basis along with assuming a closed world (closed-world-assumption) may be useful for building accurate maps of large structured or less structured environments, even in 3D: This is the case of the *Sparse Maps* which represents the most common environment representation nowadays.

Figure 3.3 The perceptual pipeline: from sensor readings to knowledge models.

In that sense, sparse statistics called *features* are used in order to create a model that is rich enough to represent the environment and sparse yet to be stored and processed efficiently.

A landmark is a feature of environment with geographic meaning, a landmark in mobile robotics is seen as a passive object of the environment that provides a considerable degree of precision in the location when the landmark is located within the field of view of the robot.

More properly, *features* are recognizable structures of elements in the environment. They usually can be extracted from measurements and mathematically described. Good features are always perceivable and easily detectable from the environment. We distinguish between low-level features (geometric primitives) like lines, circles, or polygons, and high level features (objects) such as edges, doors, tables, or a

trash can. At one extreme, raw sensor data provide a large volume of data, but with low distinctiveness of each individual quantum of data. Making use of raw data has the potential advantage that every bit of information is fully used, and thus there is a high conservation of information. Low-level features are abstractions of raw data, and as such provide a lower volume of data while increasing the distinctiveness of each feature. The hope, when one incorporates low-level features, is that the features are filtering out poor or useless data, but of course it is also likely that some valid information will be lost as a result of the feature extraction process.

High-level features provide maximum abstraction from the raw data, thereby reducing the volume of data as much as possible while providing highly distinctive resulting features.

The abstraction process has the risk of filtering away important information, potentially lowering data utilization. Figure 3.3 shows the perceptual pipeline from sensor readings to knowledge models.

Although features must have some spatial locality, their geometric extent can range widely. For example, a corner feature inhabits a specific coordinate location in the geometric world. In contrast, a visual "fingerprint" identifying a specific room in an office building applies to the entire room, but has a location that is spatially limited to the one particular room.

In mobile robotics, features play an especially important role in the creation of environmental models. They enable more compact and robust descriptions of the environment, helping a mobile robot during both map-building and localization. When designing a mobile robot, a critical decision revolves around choosing the appropriate features for the robot to use.

3.3.2 Features and Maps

All the definitions presented in this chapter are formulated in a 2D context, but of course, are valid for the full 3D case. In following chapters we will analyze the 3D case.

We suppose that the world is occupied by a set of discrete *landmarks* or *features* whose location-orientation and geometry (with respect to a defined coordinate frame) can be described by a set of parameters expressed into a feature vector $\hat{y}_n$.

We call a *Map* a collections of n features such that $\mathrm{M} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3 \ldots \hat{y}_n\}$. We will use M to denote the map vector which is simply the concatenation of all the features in the vector M:

$$M = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} \tag{3.30}$$

We begin considering to using the simplest feature possible: a point feature such that for the i^{th} feature:

$$\hat{\mathrm{y}}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.31}$$

where x_i and y_i are the Euclidean coordinates of the point in a global frame reference. Point features are not uncommon. Points occur at the intersection of lines, corners of rectangles, edges of objects, etc.

3.3.3 Observations

We define two types of observations, all denoted as z: vehicle relative and absolute.

Absolute: Absolute observations are made with the help of some external device and usually involve a direct measurement of some aspect of the vehicle´s pose. The best examples are a GPS and a compass. They depend on external infrastructure and are nothing to do with the map.

Vehicle Relative: This kind of observations involves sensing the relationship between the vehicle and its immediate surroundings (especially the map), Figure 3.4. An example is the measurement of the angle and distance to a point feature, (with respect to the robot´s own frame of reference). Odometry is another example of vehicle-relative measurement because it is not a direct measurement of the vehicle´s state.

3.3.4 Probabilistic Framework

It is informative to describe the localization and mapping task in terms of likehoods (observation Probability distribution Function, pdf) and priors.

3.3.5 Probabilistic Localization

For the localization task, we look for a pdf for the vehicle pose given map an observations $p(\mathrm{x_v},\mathrm{M},\mathrm{Z}^K)$ in this case:

$$p(\mathrm{x}_v|\mathrm{M},\mathrm{Z}^k) = \frac{p(\mathrm{Z}^k|\mathrm{x}_v,\mathrm{M})p(\mathrm{x}_v|\mathrm{M})}{\int_{-\infty}^{\infty} p(\mathrm{Z}^k|\mathrm{x}_v,\mathrm{M})p(\mathrm{x}_v|\mathrm{M})} \tag{3.32}$$

3.3.6 Probabilistic Mapping

For the mapping task, we have the vehicle location and sequence of relative observations. We wish to find the distribution of M conditioned on Z^K and $\mathrm{x_v}$, in this case:

$$p(\mathrm{M}|\mathrm{x}_v, \mathrm{Z}^k) = \frac{p(\mathrm{Z}^k|\mathrm{x}_v, \mathrm{M})p(\mathrm{M}|\mathrm{x}_v)}{\int_{-\infty}^{\infty} p(\mathrm{Z}^k|\mathrm{x}_v, \mathrm{M})p(\mathrm{M}|\mathrm{x}_v)} \tag{3.33}$$

In both cases, localization and mapping the integrals in the denominators are just normalizing constants Equations. The Kalman filter would appear to be an excellent way in which to implement these equations. If we parameterize the random vectors x_v and M with first and second order statistics (mean and variance) then the Kalman Filter will calculate the minimum mean square error (MMSE) estimate of the posterior. If it is assuming Gaussian distributions, then the Kalman filter is the optimal Bayesian estimator. The Kalman filter provides a real time way to perform state estimation.

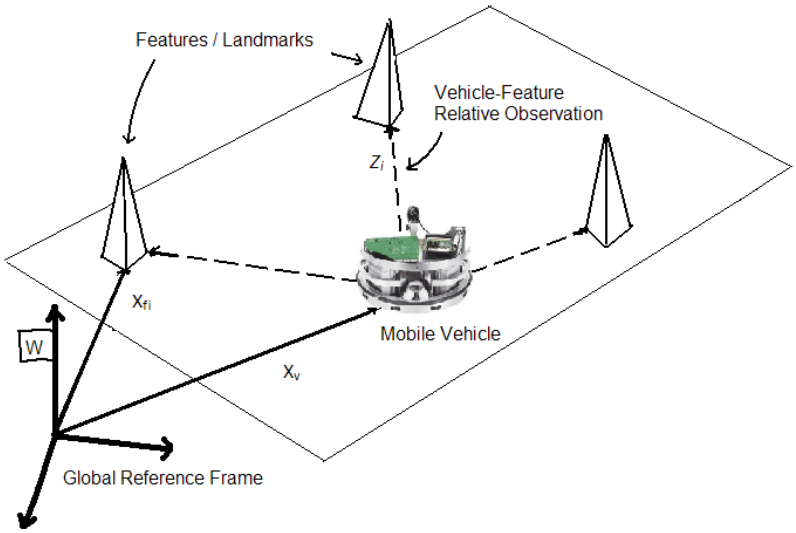


Figure 3.4 Feature based Navigation and Mapping.

3.3.7 Feature Based Localization

In this case, we are given a map M containing a set of features and a stream of observations of measurements between the vehicle and these features. We assume that the associations between measurements and observed features are given (Data association problem obviated). For the simulations, we use the differential drive model previously explained (Section 3.2) in order to have a sequence of dead-reckoned positions as the input into the prediction stage.

We have already the prediction equations from (3.1) and (3.2):

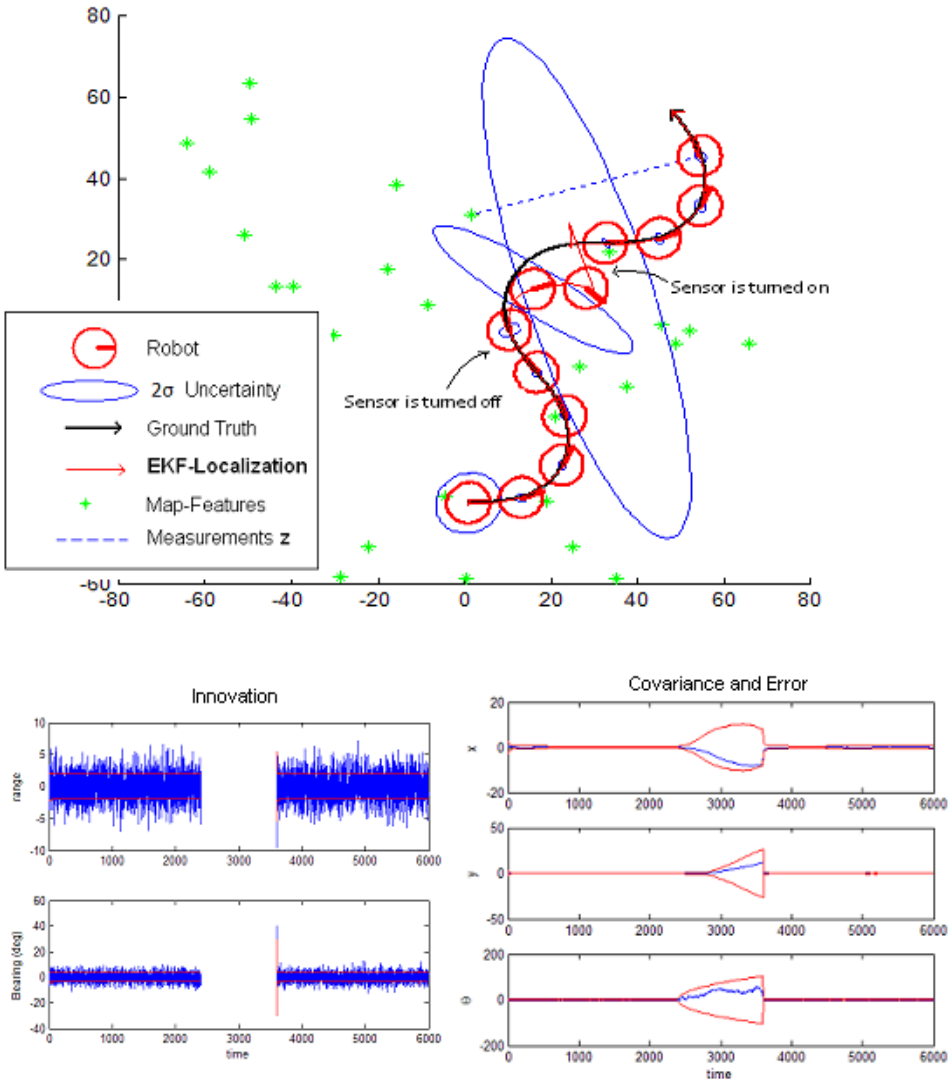


Figure 3.5 Feature Based Localization simulation: The vehicle moves through a field of random point features (Upper Graphic). The sensor is turned off for a while at the middle of the trajectory. In the innovation and error-covariance plots (Lower graphics) note how the position uncertainty grows rapidly when there is not incoming measurements, but reduces when a feature is re-observed.

$$\hat{\mathrm{x}}_v(k+1) = f(\hat{\mathrm{x}}(k), u(k)) \tag{3.34}$$

$$P(k+1) = \nabla F_x P(k) \nabla F_x^{\mathsf{T}} + \nabla F_v Q \nabla F_v^{\mathsf{T}} \tag{3.35}$$

The observation equation is simply a range r_i and bearing θ_i to the i^{th} feature:

$$\mathrm{z}(k) = \mathrm{h}(\hat{\mathrm{x}}(k), \mathrm{w}(k)) \tag{3.36}$$

$$\mathrm{z}(k) = \begin{bmatrix} r \\ \theta \end{bmatrix} \tag{3.37}$$

$$\begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_v(k))^2 + (y_i - y_v(k))^2} \\ atan2\left(\dfrac{y_i - y_v(k)}{x_i - x_v(k)}\right) - \theta_v \end{bmatrix} \tag{3.38}$$

We differentiate (3.38) with respect to $\hat{\mathrm{x}}_v$ for obtaining the observation model Jacobian (3.9):

$$\nabla \mathrm{H}_x = \frac{\partial \mathrm{h}}{\partial \mathrm{x_v}} \tag{3.39}$$

$$\nabla \mathrm{H}_x = \begin{bmatrix} \dfrac{x_i - x_v(k)}{r} & \dfrac{y_i - y_v(k)}{r} & 0 \\ \dfrac{y_i - y_v(k)}{r^2} & -\dfrac{x_i - x_v(k)}{r^2} & -1 \end{bmatrix} \tag{3.40}$$

We assume independent errors on range and bearing measurements and use a diagonal observation covariance matrix R:

$$\mathrm{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{3.41}$$

The equations above are enough for implement the Prediction and Update equations in an Extended Kalman Filter (Section 3.3.2). Figure 3.5 shows the trajectory of the vehicle as it moves through a field of random point features. In this trajectory it is simulated a sensor failure. During this time the vehicle becomes more and more lost. When the sensor comes back on line there is a jump in estimated vehicle location back to one close to the true position.

3.3.8 Feature based Mapping

The Mapping is the dual of Localization. In this case we assume that the vehicle knows exactly where it is, but not what is in the environment. The state vector for this problem is larger due to containing the concatenation of all point features. The

observation equation is the same as for the localization case only now the feature coordinates are the free variables.

The prediction model for the state is trivial. We assume that features don`t move and therefore $\hat{\mathrm{x}}(k+1|k)_{map} = \hat{\mathrm{x}}(k|k)_{map}$. Initially the map is empty and so a method is needed in order to add new discovered features to the map.

The feature initialization function Y takes as arguments the old state vector and an observation to a landmark and returns a new, longer state vector with the new feature at its end.

$$\hat{\mathrm{x}}(k)_{new} = y\big(\hat{\mathrm{x}}(k), \mathrm{z}(k)\big) \tag{3.42}$$

$$= \begin{bmatrix} \hat{\mathrm{x}}(k) \\ g(\mathrm{x}(k), \mathrm{z}(k), \hat{\mathrm{x}}_v(k)) \end{bmatrix} \tag{3.43}$$

For the case of a range bearing measurement:

$$\hat{\mathrm{x}}(k)_{new} = \begin{bmatrix} \hat{\mathrm{x}}(k) \\ x_v + r\cos(\theta + \theta_v) \\ y_v + r\sin(\theta + \theta_v) \end{bmatrix} \tag{3.44}$$

Where the coordinates of the new feature are given by the function g:

$$\hat{y}_{new} = \mathrm{g}(\mathrm{x}(k), \mathrm{z}(k), \mathrm{x}_v(k)) \tag{3.45}$$

$$= \begin{bmatrix} x_v + r\cos(\theta + \theta_v) \\ y_v + r\sin(\theta + \theta_v) \end{bmatrix} \tag{3.46}$$

When a new feature is added to the state, the covariance matrix P should be transformed. In this case the jacobian of the transformation is used:

$$\mathrm{P}(k)_{new} = \nabla \mathrm{Y}_{\mathrm{x,z}} \begin{bmatrix} \mathrm{P}(k) & 0 \\ 0 & \mathrm{R} \end{bmatrix} \nabla \mathrm{Y}_{\mathrm{x,z}}{}^{\mathsf{T}} \tag{3.47}$$

Where

$$\nabla \mathrm{Y}_{\mathrm{x,z}} = \begin{bmatrix} \mathrm{I}_{n\times n} & 0_{n\times n} \\ \nabla \mathrm{G}_{\mathrm{x}} & \nabla \mathrm{G}_{\mathrm{z}} \end{bmatrix} \tag{3.48}$$

Being $\nabla \mathrm{G}_{\mathrm{x}} = \partial \mathrm{g}/\partial \mathrm{x}$ and $\nabla \mathrm{G}_{\mathrm{z}} = \partial \mathrm{g}/\partial \mathrm{h}$. For the purely mapping case where the state vector contains only features, the new features not depend on any element of the state vector. Therefore $\nabla \mathrm{G}_{\mathrm{x}}=0$ and:

$$\mathrm{P}(k)_{new} = \begin{bmatrix} \mathrm{P}(k) & 0 \\ 0 & \nabla \mathrm{G}_{\mathrm{z}} \mathrm{R} \nabla \mathrm{G}_{\mathrm{z}}{}^{\mathsf{T}} \end{bmatrix} \tag{3.49}$$

In the case of point features:

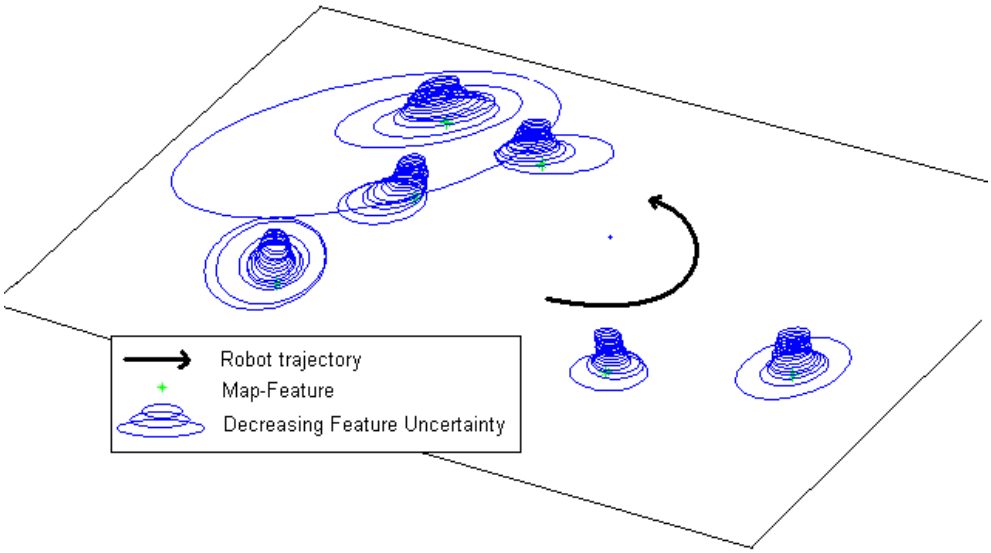


Figure 3.6 Mapping Simulations. The vertical axis represents time (k). Covariance ellipses for each feature are plotted in the z=K planes. The filter converges to a perfect map because no process noise is added in the prediction step.

$$\nabla G_z = \begin{bmatrix} \cos(\theta + \theta_v) & -r\sin(\theta + \theta_v) \\ \sin(\theta + \theta_v) & r\cos(\theta + \theta_v) \end{bmatrix} \tag{3.50}$$

In the mapping case the observation Jacobian is now "long and thin". When observing feature i it is only non-zero at the "location" (indexes) of the features in the state vector:

$$\nabla H_x = \begin{bmatrix} \underbrace{\cdots 0 \cdots}_{other\ features} & \underbrace{\nabla H_{y_i}}_{observed\ feature} & \underbrace{\cdots 0 \cdots}_{other\ features} \end{bmatrix} \tag{3.51}$$

In this case ∇H_{yi} is the Jacobian of the measurement equation respect to observed feature y_i for the measurement equation (3.38) is:

$$\nabla H_{y_i} = \frac{\partial h}{\partial y_i} \tag{3.52}$$

$$\nabla \mathrm{H}_{y_i} = \begin{bmatrix} -\dfrac{x_i - x_v(k)}{r} & -\dfrac{y_i - y_v(k)}{r} \\ -\dfrac{y_i - y_v(k)}{r^2} & \dfrac{x_i - x_v(k)}{r^2} \end{bmatrix} \tag{3.53}$$

Figure 3.6 shows the evolution of the map over time. Because no process noise is ever added to the system the uncertainty in feature locations after initializations is always decreasing. In the limit the map will be known perfectly. The vehicle is not changing its position estimate as a function of landmark observations. As a consequence all features are independent and each observation of them simply adds information reducing their uncertainty. This is in contrast to SLAM case where landmark observations will be allowed to adjust map the vehicle estimations simultaneously.

3.4 Simultaneous Localization and Mapping (SLAM)

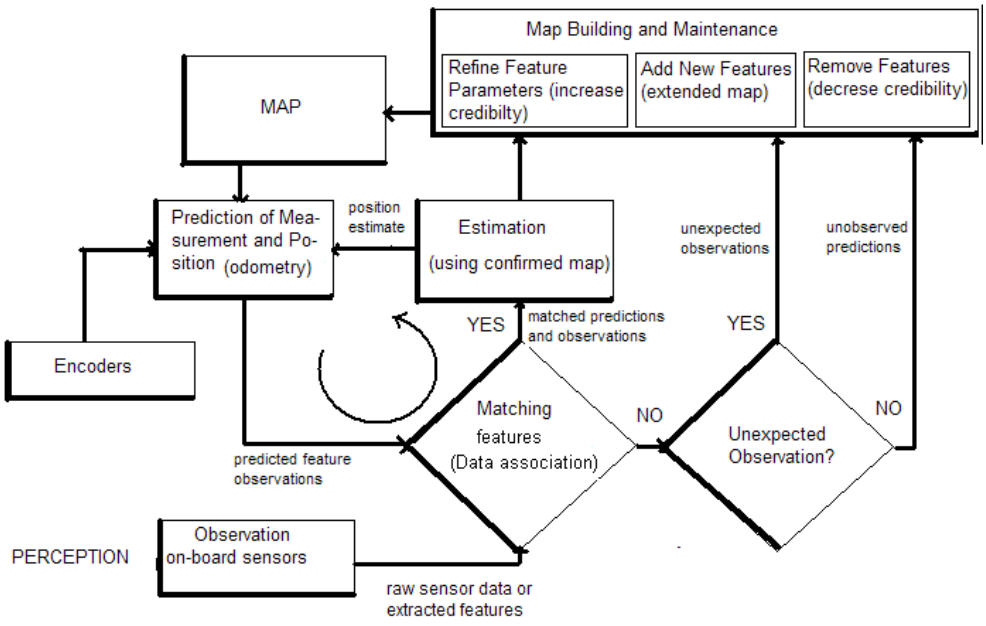


Figure 3.7 General SLAM scheme: The diagram incorporates Map building and maintaining inside the standard localization cycle. The arcs representing the additional information flow required when an imperfect matching between observations and predictions occurs. [88].

As was seen in chapter 2, the on-line robot estimation position from measurements of self-mapped features is a class of problem called. In the robotics community, as Simultaneous Localization and Mapping (SLAM) problem, this is one of the fundamental problems in robotics. SLAM consist in incrementally building a consistent map of the environment and, at the same time, localizing the position of the robot while explores its world.

In SLAM no use is made of prior maps or external infrastructure such as GPS. A SLAM algorithm builds a consistent estimate of both environment and vehicle trajectory using only noisy prioceptive (e.g., inertial, odometric) and vehicle-centric (e.g., radar, camera and laser) sensors. In this case, even though the observations are made relative to the robot frame, they are fused to create a global map.

Figure 3.7 shows a general SLAM scheme, this diagram includes the data association between observations and predictions, in this chapter we have obviated this problem, assuming that we have an "oracle" that can tell us exactly the relations between the already mapped features and the current measurements. Nevertheless the data association problem is fundamental in SLAM and will be discussed in following chapters. For now we will continue obviating the data association problem.

Much of what is necessary for implementing a basic EKF-based SLAM algorithm has been described in previous sections. In this section we summarize all the blocs necessaries for implementing this basic EKF-based SLAM algorithm.

3.4.1 SLAM State Vector and its Covariance

Now the current estimates of the locations of the robot and the Map features are stored in the system SLAM state vector $\hat{\mathrm{x}}$, and the uncertainty of the estimates in the covariance matrix P:

$$\hat{\mathrm{x}} = \begin{bmatrix} \hat{\mathrm{x}}_v \\ \hat{\mathrm{y}}_1 \\ \vdots \\ \hat{\mathrm{y}}_n \end{bmatrix} \tag{3.54}$$

$$\mathrm{P} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \cdots & \mathrm{P}_{xy_n} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \cdots & \mathrm{P}_{y_1y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{P}_{y_nx} & \mathrm{P}_{y_ny_1} & \cdots & \mathrm{P}_{y_ny_n} \end{bmatrix} \tag{3.55}$$

P is symmetric, with dimensions equals to the system state. $\hat{\mathrm{x}}$ and P will change in dimension as features are added or deleted from the map. $\hat{\mathrm{x}}_{\mathrm{v}}$ is the robot position estimate, and $\hat{\mathrm{y}}_{\mathrm{i}}$ the estimated location of the i^{th} feature, all in the world frame W.

For simulations of the current chapter, we defines $\hat{\mathrm{x}}_{\mathrm{v}}$ and $\hat{\mathrm{y}}_{\mathrm{i}}$ as :

$$\hat{\mathrm{x}}_v = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix} \quad , \qquad \hat{\mathrm{y}}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.56}$$

3.4.2 Filter Initialization

Commonly, when the algorithm starts, there is not previous knowledge of the vehicle position or any map feature, being the robot coordinates are $\mathrm{x_v}$=0, $\mathrm{y_v}$=0, θ_v=0 and therefore the world coordinate frame is initially aligned with the robot's coordinate frame. The covariance matrix has all entries equal to zero:

$$\hat{\mathrm{x}}_v = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad , \quad \mathrm{P} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.57}$$

3.4.3 Moving and Making Predictions

A robot motion model (like the presented in section 3.2.1) is used in order to propagate the position estimations and their uncertainty, this model is discretised into steps of time interval Δt, with an incrementing label k affixed to each. After each step, a new state and covariance are produced:

$$\hat{\mathrm{x}}(k) = \begin{bmatrix} f\left(\hat{\mathrm{x}}_v(k), u(k)\right) \\ \hat{\mathrm{y}}_1(k) \\ \vdots \\ \hat{\mathrm{y}}_n(k) \end{bmatrix} \tag{3.58}$$

$$P(k+1) = \nabla F_x P(k) \nabla F_x{}^{\intercal} + \nabla F_u Q \nabla F_u{}^{\intercal} \tag{3.59}$$

For the experiments of this chapter f_{v} is defined in equation (3.20), and Q in (3.27). We are assuming a static map where the features remain static; hence note that the system state part corresponding to the map does not change in the prediction step.

The Jacobians ∇F_x and ∇F_v are defined by:

$$\nabla F_x = \begin{bmatrix} \frac{\partial f_v}{\partial x_v} & 0 \\ 0 & \mathrm{I}_{n\times n} \end{bmatrix} \quad , \; \nabla F_u = \begin{bmatrix} \frac{\partial f_v}{\partial u} & 0 \\ 0 & 0_{n\times n} \end{bmatrix} \tag{3.60}$$

Where n is equal to the dimension corresponding to the map features in the system state.

3.4.4 Predicting a Measurement and Searching

The way to measure a particular feature i is determined by its measurement model h_i, and the measurement noise R. Analogous to process noise, measurement noise

takes account of the things which we do not model in the feature measurement model. Whenever we wish to measure a particular feature, the value of the measurement can be predicted by substituting current estimates $\hat{x}_v$ and $\hat{y}_i$ into the expressions for h_i.

$$z_i = h_i(\hat{x}_i, \hat{y}_i) \tag{3.61}$$

In the SLAM case the Jacobian measurement model ∇H include both parts: the derivatives respect to vehicle state ∇H_x and the derivatives respect to the feature state ∇H_{yi}.

$$\nabla H = \left[\underbrace{\nabla H_x}_{vehicle\ state} \quad \underbrace{\cdots 0 \cdots}_{other\ features} \quad \underbrace{\nabla H_{y_i}}_{observed\ feature} \quad \underbrace{\cdots 0 \cdots}_{other\ features} \right] \tag{3.62}$$

For the examples showed in this chapter, we have been obviating the data association problem, that consist in establish a unique relationship between an already mapped feature and the incoming measurements, nevertheless for real SLAM implementations searching techniques can improve the robustness in the data association process. In this way, regions of measurability can be defined for each feature, and aid robustness by only allowing match attempts from positions where the chances are good [89].

The innovation covariance S_i is how much the actual measurement z_i is expected to deviate from this prediction:

$$\begin{aligned} S_i = \frac{\partial h_i}{\partial x_v} P_{xx} \frac{\partial h_i}{\partial x_v}^{\intercal} + \frac{\partial h_i}{\partial x_v} P_{xy_i} \frac{\partial h_i}{\partial y_i}^{\intercal} + \frac{\partial h_i}{\partial y_i} P_{y_i x} \frac{\partial h_i}{\partial x_v}^{\intercal} \\ + \frac{\partial h_i}{\partial y_i} P_{y_i y_i} \frac{\partial h_i}{\partial y_i}^{\intercal} + R \end{aligned} \tag{3.63}$$

Calculating S_i before making measurements allows us to form a search region in measurement space for each feature at a chosen number of standard deviations.

3.4.5 Updating the State Vector

To update the map based on a set of measurements z_i, we perform an EKF update step. For measurements h_i, the Kalman gain W can be calculated and state updated as follows using the system state and covariance matrix estimated in the Prediction Step (Section 3.4.3):

$$W = P(k+1) \frac{\partial h}{\partial x}^{\intercal} S^{-1} \tag{3.64}$$

$$\hat{x}(k) = \hat{x}(k+1) + W(z_i - h_i) \tag{3.65}$$

$$\hat{P}(k) = \hat{P}(k+1) - WSW^{\mathrm{T}} \tag{3.66}$$

If does not exist measurements at the step k then the system state and covariance matrix are updated with the prediction step estimations, $\hat{\mathrm{x}}(k)=\hat{\mathrm{x}}(k+1)$ and $P(k)=P(k+1)$.

3.4.6 Adding a new Feature

When an unknown feature is first observed, a measurement $\mathrm{z_i}$ is obtained from its position relative to the robot along with a measurement noise R. If the measurement function $h_i(\hat{\mathrm{x}}_\mathrm{v},\hat{\mathrm{y}}_\mathrm{i})$ could be converted to $\mathrm{y_i}(\hat{\mathrm{x}}_\mathrm{v},\hat{\mathrm{y}}_\mathrm{i})$, then assuming that two features are known, this new feature $\hat{\mathrm{y}}_\mathrm{i}$ can be initialized in the state $\hat{\mathrm{x}}$ and its covariance matrix P:

$$\hat{\mathrm{x}} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \end{bmatrix} \quad \hat{\mathrm{x}}_{\mathrm{new}} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \\ \mathrm{y_i} \end{bmatrix} \tag{3.67}$$

From equations (3.47) and (3.48) we have that:

$$P = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} \end{bmatrix}$$

$$P_{new} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \mathrm{P}_{xx}\nabla G_x{}^{\mathrm{T}} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \mathrm{P}_{y_1x}\nabla G_x{}^{\mathrm{T}} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \mathrm{P}_{y_2x}\nabla G_x{}^{\mathrm{T}} \\ \nabla G_x\mathrm{P}_{xx} & \nabla G_x\mathrm{P}_{xy_1} & \nabla G_x\mathrm{P}_{xy_2} & \nabla G_x\mathrm{P}_{xx}\nabla G_x{}^{\mathrm{T}} + \nabla G_z\mathrm{R}\nabla G_z{}^{\mathrm{T}} \end{bmatrix} \tag{3.68}$$

It should be noted that after several initializations, the bias introduced to the system could be substantial due to the implicit linearization process.

3.4.7 Removing a Feature

In real SLAM implementation, is very common that the old-weak features are removed from the system state. Removing features is much easier than adding them. For deleting features from state vector and covariance matrix, the rows and columns which contain it, have to be removed. An example in a system where the second of three features want to be removed is:

$$\begin{bmatrix} x_v \\ y_1 \\ y_2 \\ \mathrm{y_3} \end{bmatrix} \rightarrow \begin{bmatrix} x_v \\ y_1 \\ \mathrm{y_3} \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \mathrm{P}_{xy_3} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \mathrm{P}_{y_1y_3} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \mathrm{P}_{y_2y_3} \\ \mathrm{P}_{y_3x} & \mathrm{P}_{y_3y_1} & \mathrm{P}_{y_3y_2} & \mathrm{P}_{y_3y_3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_3} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_3} \\ \mathrm{P}_{y_3x} & \mathrm{P}_{y_3y_1} & \mathrm{P}_{y_3y_3} \end{bmatrix} \tag{3.69}$$

3.4.8 The role of Correlations

The covariance matrix P as a structure that it can be partitioned in vehicle P_{vv} and map P_{mm} diagonals blocks.

$$\mathrm{P} = \begin{bmatrix} \mathrm{P}_{vv} & \mathrm{P}_{vm} \\ \mathrm{P}_{vm} & \mathrm{P}_{mm} \end{bmatrix} \tag{3.70}$$

Where P_{vm} are the correlations between map and vehicle. It's pretty clear that there should be correlations between map and vehicle as they are so interrelated. From the moment of initialization the feature´s location is a function of vehicle location and so errors in vehicle locations will also appear as errors in feature location. Correlations cause adjustments in one state to ripple into adjustments in others states.

Every observations of a feature affect the estimate of every other feature in the map. In the same way, the observation of a previously mapped feature can affect the estimate of the current vehicle location. This is a very important behavior in SLAM, because inevitably, when the vehicle goes far away from its initial position, there always will be a drift in the vehicle and map estimations, on the other hand this drift could be minimized if the vehicle returns to a previously mapped site and is able of redetect an old map feature.

In the simulations presented in the next section it can be clearly appreciated the effect of correlations in covariance matrix.

3.4.9 Simulations

Figure 3.8 shows a SLAM simulation for a differential drive robot (section 3.2.1) equipped with a range and bearing sensor (3.38). The robot follows a predefined cycled trajectory, emulating the corridors of a building.

In simulation, ground truth (G.T.) of robot's trajectory and map are compared against their estimates along with ellipses illustrating the uncertainties in estimations.

At the beginning of the trajectory the robot have no previous knowledge of its environment. In graphic (**a**) the first landmark is detected and is added as a new feature in system state and covariance matrix. We have supposed a sensor with 90 degrees of working area and a short measurement range, in order to emulate the "in-corridors" behavior. In graphic (**b**) three features have been added, also note that estimated robot location have begun to deviating from the real one. In this case we have supposed a

relative pessimistic odometry in the simulation. In graphic (**d**) the robot is close to finish its first lap. Note how the robot's uncertainty has grown; also note the gradual growth in the feature's uncertainties, due to the feature´s location is a function of vehicle location. At graphic (**e**) the first mapped feature has been redetected.

In this simulation we have obviated the data association problem; therefore we always know when a new measurement corresponds to a previously mapped feature. The first mapped feature has a lower uncertainty (respect to the last mapped features) and acts like a "strong" landmark, and consequently when it is re-measured; "pull" all the others features. This is an important behavior in SLAM, and is due to the correlations in the full covariance matrix.

Note how the estimated feature's locations have been closer from their ground truth and the estimated robot's location has been improved close to its real one. The problem of redetect previously mapped sites in order to minimize both map and robot drift is major challenge in SLAM and is called "the closing cycle problem". Finally, in graphic (**f**) the robot has complete several laps. Note how the map and trajectory correspond properly with the real ones, and the uncertainties have been minimized.

3.5 Conclusions

In this chapter we presented an introduction to a general EKF-based SLAM algorithm. Many mathematical bases, which will be used in following chapters, were introduced. In that sense, an uncomplicated case of SLAM was formulated; assuming odometry as the control input, 2D representation of both Map and robot's location, Euclidean parameterization for features points, Data association problem obviated and availability of range and bearing measurements. In following chapters, this SLAM formulation, will evolve in order to be suitable to more challenging and realistic scenarios.

In the MATLAB simulations, can be appreciated some common and important behaviors in SLAM; Using measurements to external landmarks helps to reduce the "dead reckoning drift", but when the robot moves far away enough to be incapable of measuring the first mapped landmarks, always will be a drift in estimates. In that sense, errors in vehicle locations will also appear as errors in feature location, and therefore will be a divergence between estimates of both map and robot's location, and the ground truth, as the robot moves far away from its initial position. On the other hand, due to the correlations in the covariance matrix, if the robot is capable of recognizing previous mapped features then both map and robot drift could be minimized.

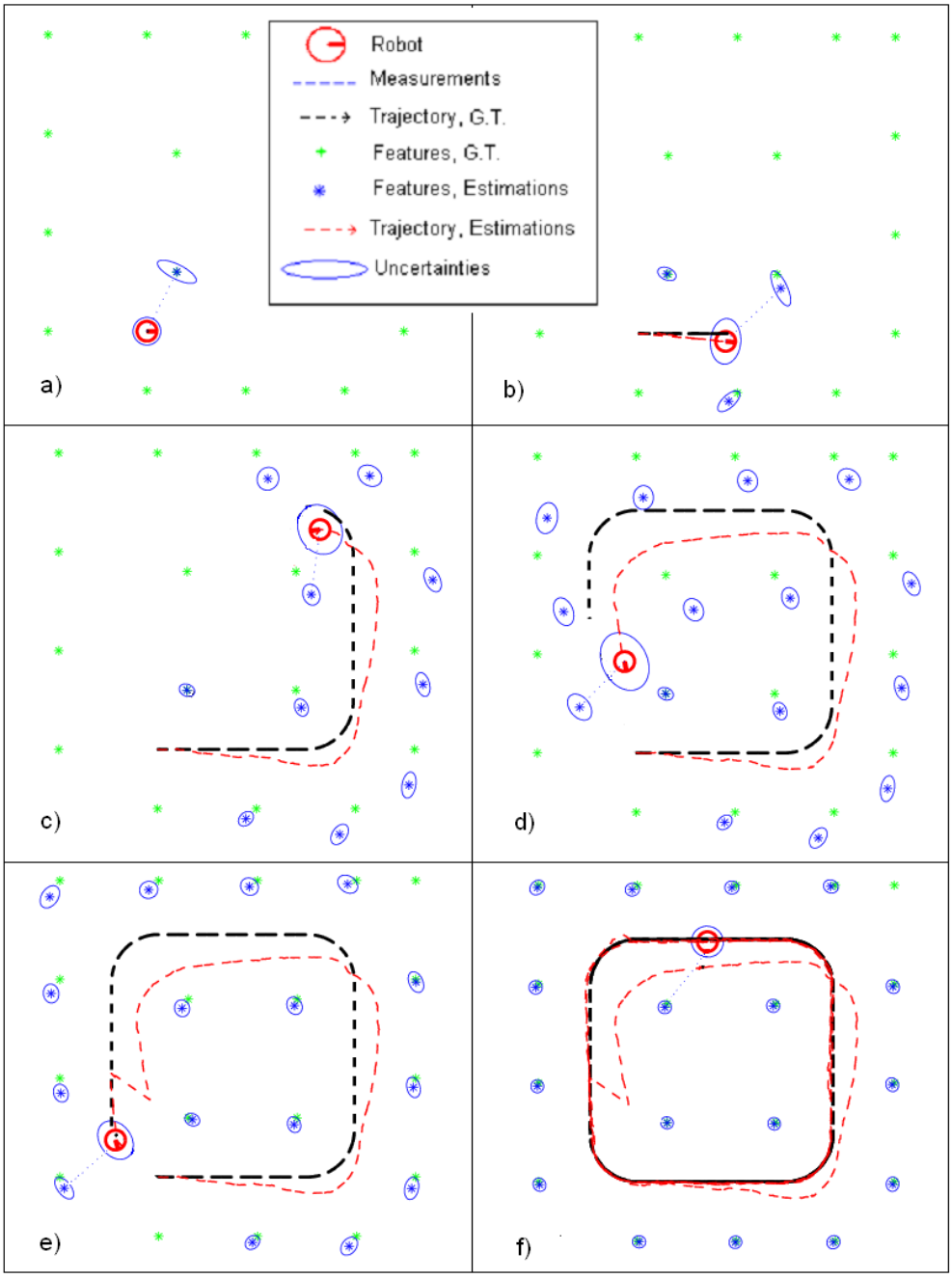


Figure 3.8 SLAM simulation of a Mobile Robot in a cycled trajectory (emulating an indoor corridor).

Chapter 4

Bearing-Only SLAM

SLAM is one of the most active research fields in robotics, with excellent results obtained during the last years, but until recently it was mainly restricted to the use of sonar and laser range-finder sensors. In this chapter we explore the use of bearing sensors in SLAM. The SLAM which is mainly bearing sensor based is commonly called Bearing-Only SLAM. In counterpart to the range-finder sensors, commonly used in SLAM, which provides range and angular information respect to the landmarks, bearing sensors only offers angular information respect to the spatial position of the landmark. On the other hand, cameras, by far the most popular bearing sensor, can provide, in addition, appearance sensing which it is a huge advantage for addressing the data association problem. Moreover Bearing-Only SLAM is not limited to the use of cameras as the main sensor, in that sense, other sensorial capabilities, as the auditory sense, can be explored in the context of Bearing-Only SLAM.

Due to the bearing sensors provides only angular spatial-information then depth information cannot be retrieved directly from a single measurement. The application of bearing sensors to the SLAM problem involves the use of special techniques in order to initialize the landmarks as new map's features. On the other hand, as we shall see later, in several scenarios, the SLAM could be benefit from the use of bearing sensors.

First in Section 4.1 an introduction to the Bearing-Only problematic is given, mainly pointing to the Features Initialization problematic which represent the main challenge in Bearing-Only SLAM, the viability of this kind of methods is also revised. In section 4.2 the most relevant related work is presented, a brief taxonomy together with a summary of methods is also given.

In Section 4.3 one of the most important contributions of this thesis is presented: the *Delayed Inverse-Depth Feature Initialization*, which is a novel method for adding new features to the map in Bearing-Only SLAM systems. In this chapter the method is

introduced assuming a 2D context with availability of odometry. First the theoretical and experimental bases of the method are given, later the method is widely described and finally several simulations are presented in order to demonstrate its performance and viability. A comparison with the *Un-delayed Inverse-Depth Feature Initialization* is also presented.

Using our general *Bearing-Only SLAM with Delayed Feature Inverse-Depth Feature Initialization* algorithm, in section 4.4, we present a Sound-based SLAM system (called SSLAM) which uses "Sound Sources" as map's features. The main contribution of the SSLAM is demonstrating the viability on the inclusion of the hearing sense in SLAM. Experimental results with real data, obtained from the sensors of a small mobile robot, are presented In order to show the performance of the SSLAM method.

Finally in Section 4.5 the conclusions of this chapter are presented.

4.1 Introduction

Bearing-only SLAM is a partially observable SLAM problem, in which the sensor, used for perceiving the robot´s environment, provides only-angular information respect to the landmarks, and therefore does not give enough information to compute the full state of a landmark from a single observation. The counterpart of this problem is the range-only SLAM: [90], [91], [92], [93], [94], common in sonar-based systems. In these kind of partially observable SLAM systems, since a single observation is not enough to estimate all the parameters of a landmark, multiple observations are taken from multiple locations in order to estimate the state of the landmark.

4.1.1 Feature Initialization Problematic

If a measurement function $h_i(\hat{x}_v,\hat{y}_i)$ is invertible to $y_i(\hat{x}_v,\hat{y}_i)$, we can initialize the feature states as was described in Section 3.4.6. If h_i is not invertible, it means that a single measurement does not give enough information to pinpoint the feature location (for instance, a single measurement of a point feature from a bearing-sensor only defines a ray on which it lies).

Let us consider again the simplest feature possible: a point feature such that for the i^{th} feature:

$$\hat{y}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.1}$$

And a sensor capable of doing measurements $z(k)$ of range and bearing $\boldsymbol{r}$ and θ:

$$z(k) = \begin{bmatrix} r \\ \theta \end{bmatrix} \tag{4.2}$$

The measurement equation h_i is:

$$h_i(x_v, y_i) = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_v(k))^2 + (y_i - y_v(k))^2} \\ atan2\left(\frac{y_i - y_v(k)}{x_i - x_v(k)}\right) - \theta_v \end{bmatrix} \quad (4.3)$$

Then for initializing a new feature $\hat{y}_i$ in the map is necessary to invert $h_i(\hat{x}_v,\hat{y}_i)$ to $y_i(\hat{x}_v,\hat{y}_i)$. From inverting equation (4.3) we obtain:

$$\hat{y}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} r\cos(\theta + \theta_v) + x_v \\ r\sin(\theta + \theta_v) + y_v \end{bmatrix} \quad (4.4)$$

Being $\hat{x}_v$ the vehicle pose (the joint of position and orientation) defined as:

$$\hat{x}_v = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix} \quad (4.5)$$

For the bearing-only case we would have a sensor capable of providing angular measurements $z(k)$ such as:

$$z(k) = [\theta] \quad (4.6)$$

Being the measurement equation h_i:

$$h_i(\hat{x}_v, \hat{y}_i) = [\theta] = atan2(y_i - y_v(k) \quad , \quad x_i - x_v(k)) - \theta_v \quad (4.7)$$

For inspection, is clear that we cannot invert equation (4.7) in order to obtain a feature state $\hat{y}_i = [x_i, y_i]$. In this case we can only define a partially initialized feature $\hat{y}_{pi}$, with a different geometrical type (e.g. a line feature to represent the ray we know a point must lie on). *atan2* function is used for obtaining an angle expressed in $-2\pi < \theta < 2\pi$.

The Jacobians of the measurement model are:

$$\nabla H_x = \begin{bmatrix} \frac{y_i - y_v(k)}{r^2} & -\frac{x_i - x_v(k)}{r^2} & -1 \end{bmatrix} \quad (4.8)$$

$$\nabla H_{y_i} = \begin{bmatrix} -\frac{y_i - y_v(k)}{r^2} & \frac{x_i - x_v(k)}{r^2} \end{bmatrix} \quad (4.9)$$

For equation (4.7) we can define a unitary vector pointing the direction on which the feature lies:

$$\hat{y}_{pi} = [\cos(\theta + \theta_v), \sin(\theta + \theta_v)\ ] \quad (4.10)$$

It´s seen that the problem with bearing-only initialization is that a single measurement is insufficient to determine the location of the feature, and at least two bearing measurements θ_i and θ_j, from two different vehicle poses x_{vi} and x_{vj} are required. The location of the feature then is calculated as the intersection of two lines as shown in Figure 4.1.

$$\hat{\mathrm{y}}_i = \mathrm{g}(\hat{\mathrm{x}}_{v_i}, \hat{\mathrm{x}}_{v_j}, \theta_i, \theta_j) \tag{4.11}$$

$$\hat{\mathrm{y}}_i = \begin{bmatrix} \dfrac{x_{v_i} s_i c_j - x_{v_j} s_j c_i + \left(y_{v_j} - y_{v_i}\right) c_i c_j}{s_i c_j - s_j c_i} \\ \dfrac{y_{v_j} s_i c_j - y_{v_i} s_j c_i + \left(x_{v_i} - x_{v_j}\right) s_i s_j}{s_i c_j - s_j c_i} \end{bmatrix} \tag{4.12}$$

Where

$$\begin{aligned} s_i &= \sin(\theta_{v_i} + \theta_i) \\ c_i &= \cos(\theta_{v_i} + \theta_i) \end{aligned} \tag{4.13}$$

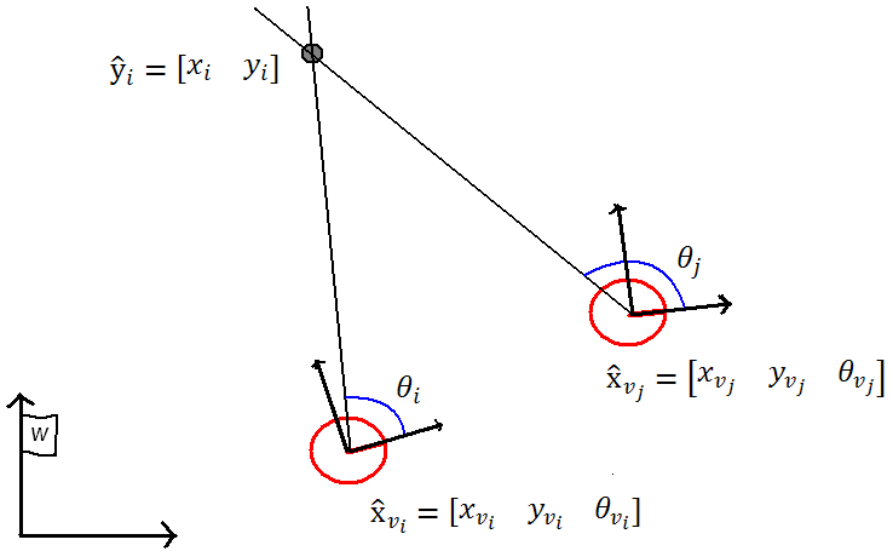


Figure 4.1 Feature initialization via the intersection of two bearing measurements.

If the bearing and pose estimates were perfectly known, the feature location would be trivial from Equation (4.12) but, as they are uncertain, the estimated feature location may be ill-conditioned depending on several factors: (i) the uncertainty of the pose estimates, (ii) the uncertainty of bearing measurements, and (iii) the base-line afforded by the two vehicle locations. Furthermore, unless the correlations between the two vehicle pose estimates are included in the initialization estimate, the estimated location may be inconsistent (i.e., the estimated uncertainty is less than its true uncertainty).

Feature initialization, is a challenging problem in bearing-only SLAM, and attracts most of the research attention in this kind of systems.

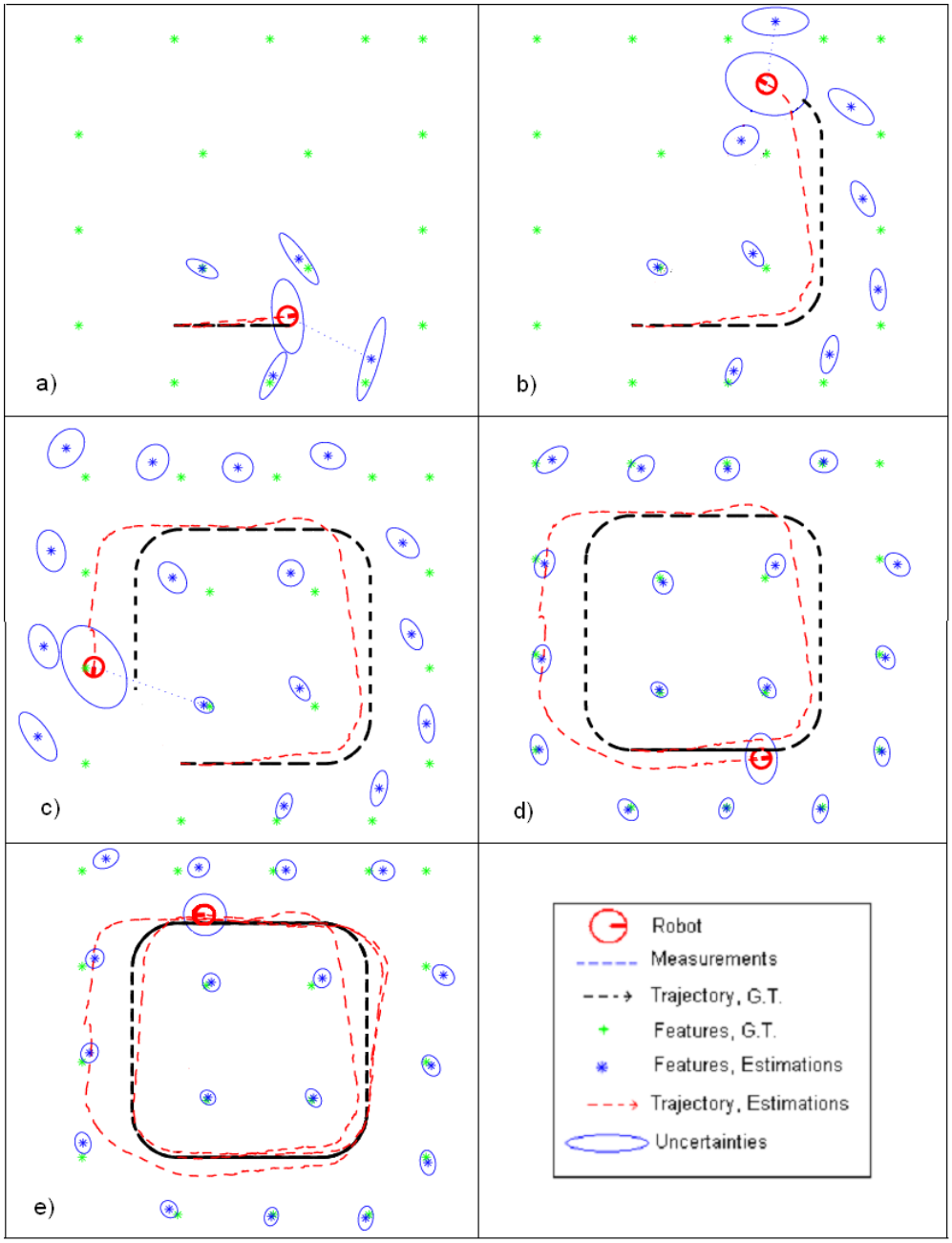


Figure 4.2 Bearing-Only SLAM simulation. In this experiment, the feature initialization problem has been obviated.

4.1.2 Viability of Bearing-Only SLAM

We have seen that Bearing-Only SLAM is a partially observable SLAM problem, and due to this fact, features initialization is a difficult challenge. But what happen if we have solved the features initialization problem, how does affect the partial observability when the features have been initialized?

Figure 4.2 shows a simulation of a differential drive robot equipped with a bearing-only sensor, for the same simulated environment used in experiments of Section 3.4.9. An important fact, of this simulation, is that feature initialization problem has been obviated. When a new feature must be initialized, in the system state and covariance matrix, we use a range and bearing sensor as was used in section 3.4.9. After a feature is initialized, we assume that the sensor can only provide bearing information respect to this feature. Therefore, in the Kalman update step, the measurement equation (3.11) is used.

If we compare Figure 3.8 and Figure 4.2, it can be note that propagation of both, robot and features uncertainties is bigger in the case of the bearing-only SLAM, this fact can be explained due to lower information provided by the bearing-only sensor, respect to the information provided by the range and bearing sensor. Nevertheless the bearing-only SLAM algorithm was also capable of successfully relocalizing the robot and minimizing map's drift, after a loop was detected.

Based in this simulation we can conclude that the bearing-only SLAM is a feasible case of SLAM if the feature initialization problem is addressed. In [95], a vast analysis in observability of Bearing-only SLAM systems can be founded.

4.2 Related Work

Most part of the literature related to the Bearing-only SLAM is associated with the camera-based SLAM systems, since the cameras are by far the most popular bearing sensor. This fact has led to a huge interaction between the robotics and computer vision societies for addressing the Bearing-only SLAM problem.

4.2.1 Bearing-Only SLAM and SFM methods

Bearing-only SLAM is similar to what in the computer vision society is referred as the structure-from-motion problem (SFM). The major difference is that the SFM methods relies on a global optimization process and commonly run off-line and consider time consuming batch processing of the entire set of images acquired in the sequence while SLAM requires incremental and computationally tractable approaches suitable for on-line and real-time processing. Furthermore, the SFM methods do not assume feedback from information sources as odometry that are commonly used in SLAM. Nevertheless some of recent work in SFM has focused in the real-time operation like [96] that shows real-time

results for a small environments using random sample consensus (RANSAC) technique for obtaining good starting point for the bundle adjustment estimation. Bundle adjustment is a batch technique that is widely used in SFM methods under perspective projection [97]. Other approach for real-time SFM is [98]

Other SFM methods take elements from the SLAM like [99] which focuses on an incremental real-time solution to the SFM problem, in this method the initial 3D coordinates are obtained by triangulation on local part of the image sequence, which limits opportunity of initializing distant points. In [100] Links between these non linear minimization algorithms and the EKF used in SLAM are studied and some attempts have also been made to recursively compute SFM using Kalman filters [101]

Another big difference of the SFM methods respect to the SLAM methods is that SFM methods do not make an implicit treatment of estimation uncertainties, which is fundamental from the robotics point of view.

4.2.2 Feature Initialization Approaches for Bearing-Only SLAM

Bearing-Only SLAM has received most attention in the current decade. Therefore many of the related approaches are actually very recent. In this section we listing (in chronologic order) some of the relevant work on Bearing-Only SLAM, focusing on the techniques used for address the feature initialization problem.

In [102] Deans proposes a combination of a global optimization BA, (Bundle Adjustment) for feature initialization and Kalman Filter for state estimation. The BA is run on a limited number of camera poses and the associated feature observations. The complexity of the initialization step is greater than Kalman Filter but theoretically gives more accurate results. Providing an appropriate “condition” measure is used to signal when to perform feature initialization. The claim of this paper is that good results are possible for bearing-only SLAM within the conventional EKF framework. In this method, due to the limitation on the baseline on which features can be initialized and depending on the camera motion and the landmark location, some features cannot be initialized.

Strelow proposes in [103] a method similar to [102] but mixing camera and inertial sensors measurements in an Iterated Extended Kalman Filter (IEKF), in that sense, image measurements can counteract the error accumulated when inertial readings are integrated, and can be used to distinguish between the effects of acceleration, gravity and bias in accelerometer measurements. Conversely inertial data can resolve the ambiguities in motion estimated by a camera that sees a degenerate scene; such as one containing too few features, features infinitely far away and also make motion estimation more robust to miss-tracked image features.

In [104] Bailey proposes a variant of constrained initialization for bearing-only SLAM, where past vehicle pose estimates are retained in the SLAM state so that feature initialization can be deferred until their estimates become well-conditioned. In that sense

the past poses of the robot are stacked in the map, together with associated measures, until base-line is sufficient to permit a Gaussian initialization. This method explicitly computes the intersection of the 3D lines defined by corner point's observations, and thus the estimation is computed using observations from two robots poses. The criteria used for determining if the estimation is well-conditioned (Gaussian) is the Kullback distance. The complexity of the sampling method proposed to evaluate this distance is quite high. Once initialized, the batch observation is used to refine and correct the whole map.

Also there has been made effort in using others estimation techniques (apart to the EKF) in Bearing-Only SLAM like the Particle Filters (PF) based methods.

In a series of papers, Kwork employs Particle Filters: In [105] Variations to standard PF are proposed to remedy the sample impoverishment problem in bearing-only slam. In [106] Initial sate of features are approximated using a sum of Gaussians, which defines a set of hypothesis for the position of the landmark, and includes them all inside the map from the beginning. On successive observation, sequential radio test (SRT) based on likelihoods is used to prune bad hypothesis, and the one with maximum likelihood is used to correct the map. The way these hypotheses are initialized is not detailed, and convergence and consistency issues are not discussed.

In [107] Kwork extend the algorithm using a Gaussian Sum Filter, with an approach similar to the proposed in [108] for bearing-only tracking. The proposed multi-hypotheses method in [107] is perhaps the first un-delayed feature initialization method. This means that an approximation to the feature depths is used for initializing the feature in the map at the first frame that it was seen. The main drawback of this approach is that the number of required filters can grow exponentially, and therefore computational load grows exponentially with number of landmarks.

Some of the most notably advances on Bearing-Only SLAM have been presented by Davison [109] [110], who shows the feasibility of real-time SLAM with a single camera, using the well-established EKF estimation framework, The system takes a top-down-Bayesian estimation approach, searching for landmarks in images regions constrained by estimate uncertainty instead of performing extensive bottom-up image processing and feature matching. In this work a Bayesian partial-initialization scheme for incorporating new-landmarks are proposed where a separate Particle Filter is used for estimating the feature depth which is not correlated with the rest of the map. In that sense it maintains a set of depth hypotheses uniformly distributed along the viewing ray of a new landmark, with a particle filter in one dimension. Each new observation is used to update the distribution of possible depths, until the range variance is small enough to consider a Gaussian estimate, at which point the estimate is added to the map as three-dimensional entity. Until this initialization occurs, the ray estimate is maintained in the system`s single Extended Kalman Filter.

The Davison approach derives in an effective real-time implementation which gives notable results. Nevertheless a main drawback of this approach is that the initial

distribution of particles has to cover all possible depth values for a landmark, this fact makes it difficult to use when the number of detected features is large or there are far features in the scene. As a result, its application in large environments is not straightforward, as it would require a huge number of particles. Figure 4.3 illustrates features initialization technique proposed by Davison.

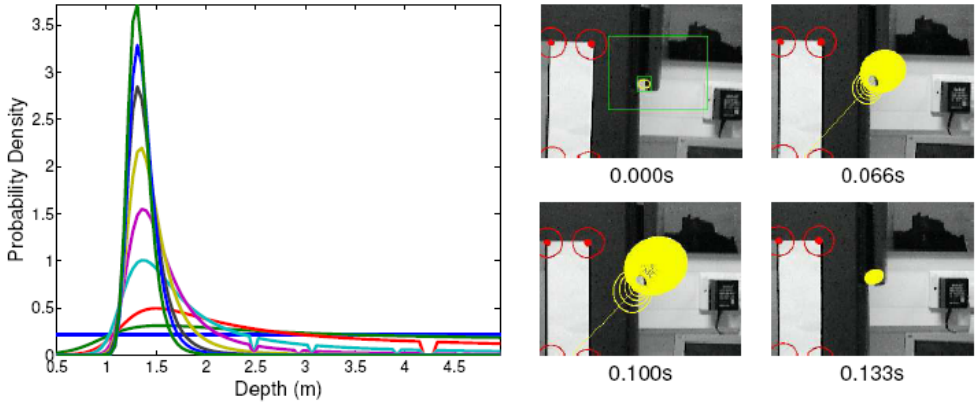


Figure 4.3 Davison features initialization. Frame-by-frame evolution of the probability density over feature depth represented by a particle set. 100 equally-weighted particles are initially spread evenly along the range 0.5m to 5.0m; with each subsequent image measurement the distribution becomes more closely Gaussian. (Graphics taken from [109])

Jensfelt in [111] presented a method where the idea is to let the SLAM estimation lag behind N frames and use these N frames to (i) determine which points are good landmarks and (ii) find an estimate of their 3D location, This way the landmarks can be initialized with an estimate of the depth immediately in the SLAM process. In this case a buffer of a constant number of images is maintained and the filter is using output of this buffer: here the filter itself is delayed. Feature points are tracked in this buffer and 3D point estimates are computed by triangulation. Most of the focus in this work is on the management of the features to archive real-time performance in extraction, matching and loop detection. A limitation of this method resides on the baseline with which the features can be initialized, depending on the camera motion and the landmark location, some features cannot be initialized.

In [112] a framework for vision-based SLAM is presented using a structure-from-motion (SFM) approach from multiple views. It is mentioned that for cases when the robot performs only translational motion along the optical axis, the 3D triangulation is significantly uncertain, due to very little or no disparity between matched features, similar to [111] the reconstruction is performed using multiple images.

In [113] Sola present a method based on Federate Kalman Filtering technique. Where initial Probability Distribution Function (PDF) for the features, is defined using a geometric sum of Gaussians. The method is an approximation of the Gaussian Sum Filter (GSF), which was used in [107], that permits un-delayed initialization with simply an additive growth of the problem size; At the first observation, the robot only knows the optical ray on which the landmark is located, this ray, with associated covariance, define a conic PDF for its position. A minimal representation of this PDF is introduced as a geometric series of Gaussians that maintain several depth hypotheses as Gaussian volumes for each initialized feature spread in a geometric sum - a development of the particle method of [109] but taking advantage to some extent of the inverse depth concept- then they are included in one single EKF-SLAM map, as with all approximations, As the estimation proceeds the hypotheses are pruned and an approximation to the GSF is proposed to keep the computational overhead low. (This representation has the risks of inconsistency and divergence) For this task a simple criterion is defined for pruning the less likely members of the ray. A drawback of this approach is that does not cope with features at very large depths.

Figure 4.4 Sola and Lemaire feature initialization. From left to right: the sum of Gaussians is initialized in the robot frame; some Gaussians are pruned based on their likelihood after additional observations of the feature; when a single hypotheses remains, the feature is projected into the map frame; and finally past observations are used to update the feature estimate. (Graphics taken from [114]).

Parallel to [113] in [114] Sola and Lemaire presents a method similar to the exposed in [113] but feature are initialized with a delayed method; The initial feature representation is initialized with an probability density function which is approximated with a particular weighted sum of Gaussians. This initial state is expressed in the robot frame, and not in the global map frame, so that it is decorrelated from the stochastic map, until it is declared as a landmark and added to the map. Many features can enter the initial estimation process at low computational cost, and the delay can be used to select

the best features. The initialization process is similar to [109], but distributes depth hypotheses uniformly in inverse depth along the ray. To this distribution corresponds to constant density of hypotheses when they are projected into the image. As new measurements are made, repeatedly prune unlikely hypotheses until only one remains. A new feature is initialized using the survivor hypothesis as starting point. This method has the same drawback than [113] in relation with far features. Figure 4.4 illustrates the initialization approach proposed by Sola and Lemaire.

Recent work as [115] or [1] have shown that the use of an inverse depth parameterization for Bearing-Only SLAM can improve the linearity of the measurement equation even for small changes in the camera position yielding small changes in the parallax angle, this fact allows a Gaussian distribution to cover uncertainty in depth which spans a depth range from nearby to infinity.

In [115] a FastSLAM [73] based approach is proposed by Eade. Here the landmark is not immediately added to the map when it is detected for the first time, instead, it enters a “partially initialized “state, and is stored using the inverse-depth parameterization. The position of each new partially initialized feature added to the map is parameterized with three coordinates representing its direction and inverse depth relative to the camera pose at the first observation. Estimates of these coordinates are refined within a set of Kalman filters for each particle of the map. While in this state, new observation of this feature is used to update, its depth estimate, and the current estimate of the camera pose using the epipolar constraint. Once the inverse depth estimation has collapsed, the feature is converted to a fully initialized standard Euclidean representation. An unscented transformation is applied to transform the landmark to the Cartesian representation.

This method is not a pure un-delayed method, but has the advantage that the observations are directly used to update the camera pose estimate; While Retain the difference partially and fully-initialized features, the method go further and are able to use measurements of partially initialized features with unknown depth to improve estimates of camera orientation via special epipolar update step. This approach for features initialization appears appropriate within a FastSLAM implementation, but lacks for a more general framework.

In [1] Montiel presented an Un-delayed method with inverse depth parameterization, within a Standard EKF based SLAM framework. Due to the inverse depth parameterization the estimation process benefits from the quasi-linearity of the observation function under the condition that the motion of the camera along the depth axis is small relatively the depth of the point. In this method transitions from partially to fully initialized features need not to be explicitly tackled, making it suitable for direct use in EKF framework for sparse mapping. In this approach the features are initialized in the first frame observed, (un-delayed initialization) with an initial fixed depth and uncertainty,

determined heuristically to cover ranges from nearby to infinity, so distant points can be coded.

Due to the clarity and scalability of this method, the approach presented by Montiel in [1] is a good option for monocular-SLAM implementation. These facts motive us to utilize this kind of parameterization for our work and consequently [1] is discussed in more detail in following sections.

4.2.3 Summary of Methods

We propose a simple taxonomy for classifying the Bearing-Only algorithms based on the techniques used for address the feature initialization problem and the kind of method used for the stochastic estimation of the main SLAM process. Table 4.1 shows a summary of relevant Bearing-Only methods:

Approach	Delayed / Un-delayed	Initial representation	Estimation
Deans 2000 [102]	Delayed	Bundle adjustment	EKF
Strelow 2003 [103]	Delayed	Triangulation	IEKF
Bailey 2003 [104]	Delayed	Triangulation	EKF
Davison 2003 [109]	Delayed	Multi-Hypotheses	EKF
Kwok 2004 [106]	Delayed	Multi-Hypotheses	PF
Kwok 2005 [107]	Un-delayed	Multi-Hypotheses	PF
Sola 2005 [113]	Un-delayed	Multi-Hypotheses	EKF
Lemaire 2005 [114]	Delayed	Multi-Hypotheses	EKF
Jensfelt 2006 [111]	Delayed	Triangulation	EKF
Eade 2006 [115]	Delayed	Single Hypotheses	FastSLAM
Montiel 2006 [1]	Un-delayed	Single Hypotheses	EKF

Table 4.1 Summary of methods.

Delayed & Un-delayed methods:

For *delayed* methods, a feature observed at time t is added to the map at subsequent time step $t + k$. This delay allows the parallax between observations of this landmark to grow, and the triangulation operation to become well conditioned.

On the other hand, un-delayed methods take advantage of the feature observation to localize the robot at time t. But the update of the stochastic map has to be computed carefully.

Initial feature representation:

Global optimization methods are based in batch processing, they are often used in SFM algorithms, and look for and optimal solution using all the available measurements. For its application to SLAM they have the same drawbacks that the SFM.

Triangulation methods explicitly compute the intersection of the lines defined by the observations, these methods requires defining a condition for considering the computation well-conditioned.

Single hypotheses methods define the initial feature representation with a single probability distribution function (PDF). Usually this methods are easy to compute, but requires attention respect to the linearity in the measurement equation.

Multi-hypotheses methods define the initial feature representation with a multiple probability distribution function (PDF). Usually these methods are robust respect to measurement equation used, nut their complexity is high.

Estimation techniques:

These are the techniques used for estimating both the vehicle and landmarks location. Most of the algorithms are based on the Extended Kalman Filter (EKF), but several are also based on the Particle Filters (PF) methods or variations.

4.3 2D Delayed Inverse Depth Feature Initialization

We have seen that the features initialization problem is fundamental for Bearing-Only SLAM, due to bearing sensors does not give enough information to compute the full state of a landmark from a single observation. On the other hand, if the features initialization problem is satisfactory solved then a Bearing-Only sensor is a viable alternative for implementing SLAM systems. Therefore, a considerable amount of the research on Bearing-Only SLAM has focused on developing techniques for addressing the feature initialization problem.

In this section a novel method for Bearing-Only SLAM, called *Delayed Inverse Depth Feature Initialization*, is proposed. It is introduced in a 2D context, assuming the availability of odometry.

4.3.1 Parallax Angle

Parallax, more accurately *motion parallax,* is the change of angular position of two observations of a single object relative to each other as seen by an observer, caused by the motion of the observer. By observing parallax, measuring angles and using geometry, one can determine the distance to various objects. Distance measurement by parallax is a special case of the principle of triangulation, which states that one can solve for all the sides and angles in a network of triangles if, in addition to all the angles in the

network, the length of a least one side has been measured. Thus the careful measurement of the length of one baseline can fix the scale of an entire triangulation network. In parallax, the triangle is extremely long and narrow, and by measuring both its shortest side (the motion of the observer) and the, small top angles (the other two being close to 90 degrees) the length of the long sides (in practice considered to be equal) can be determined.

In SLAM a Bearing-Only sensor cannot retrieve depth information in a single measurement. To infer the depth of a feature, the sensor must observe it repeatedly as it translates though the environment, each time capturing a "ray" from the feature to its center. The angle between the captured rays is the feature´s parallax, allowing depth estimation.

4.3.2 Inverse Depth Parameterization

Recently, the inverse depth parameterization has shown to be good alternative for the Bearing-Only SLAM problem, in a scheme of Extended Kalman Filtering for the estimation of the stochastic map and vehicle pose.

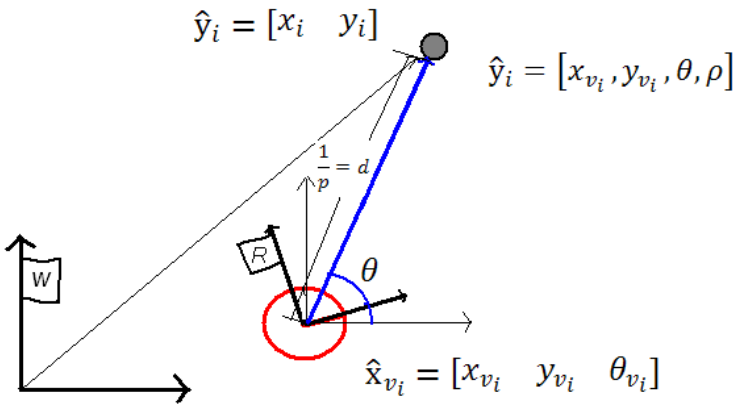


Figure 4.5 Change features parameterization from Euclidean coordinates to Inverse-Depth Polar coordinates.

We are using the EKF to estimate the state. The more linear the measurement equation is, the better performance is expected from the Kalman filter. In recent works, [1] and [115] have shown that the use of an inverse depth parameterization for Bearing-Only SLAM can improve the linearity of the measurement equation even for small changes in sensor position yielding small changes in the parallax angle, this fact allow a Gaussian distribution to cover uncertainty in depth which spans a depth range from nearby to

infinity. Montiel in [1] presented on of the most promising approaches for Bearing-Only SLAM. In this approach based in a simple parameterization, far features can be included in the map, (due to the linearity in the measurement equation) in a context of standard EKF framework for sparse mapping. This method is clear and scalable; hence we use this parameterization for our work.

So far we have been representing the map features in a Euclidean manner, thus for a 2D point:

$$\hat{y}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.14}$$

The same $\hat{y}_i$ point can be represented in polar coordinates:

$$\hat{y}_i = [x_{ref}, y_{ref}, \theta, d] \tag{4.15}$$

Where x_{ref} and y_{ref} represent a reference point, expressed in the global coordinate frame. In our case x_{ref} and y_{ref} are represented by the location of the vehicle x_{vi} and y_{vi}. θ represents the direction in global coordinates of the ray d. (See Figure 4.5).

The really key of the inverse depth parameterization is representing the point depth along the ray d coded in its inverse:

$$\rho = \frac{1}{d} \tag{4.16}$$

Therefore, our new feature representation is defined by:

$$\hat{y}_i = [x_i, y_i, \theta, \rho] \tag{4.17}$$

Figure 4.5 shows both, Euclidean and Inverse Depth parameterization or the map features. Note that the new feature parameterization also describes in a more natural manner a feature measurement, since the feature is represented using the vehicle parameters x_{vi} and y_{vi}. For simplicity note that in equation (4.17) we remove the subscript *v* related to the vehicle location.

4.3.3 Improving Linearity with Inverse Depth Parameterization

Some feature initialization methods for Bearing-Only SLAM employ multi-hypotheses based techniques, like Particle Filters [109] or Sum of Gaussians and variations [105], [106], [113], [114]. This, mainly due to the multi-hypotheses techniques are better suited for highly non-linear problems, in contrast the EKF, which is based in a first order linearization, suffer when uncertainty cannot be represented by Gaussian noise.

On the other hand, inverse depth parameterization improves the linearity in the measurement process. Figure 3.1 show a simulation for a point reconstruction from noisy bearing measurements at different parallax, using both, the Euclidean and the Inverse Depth parameterization.

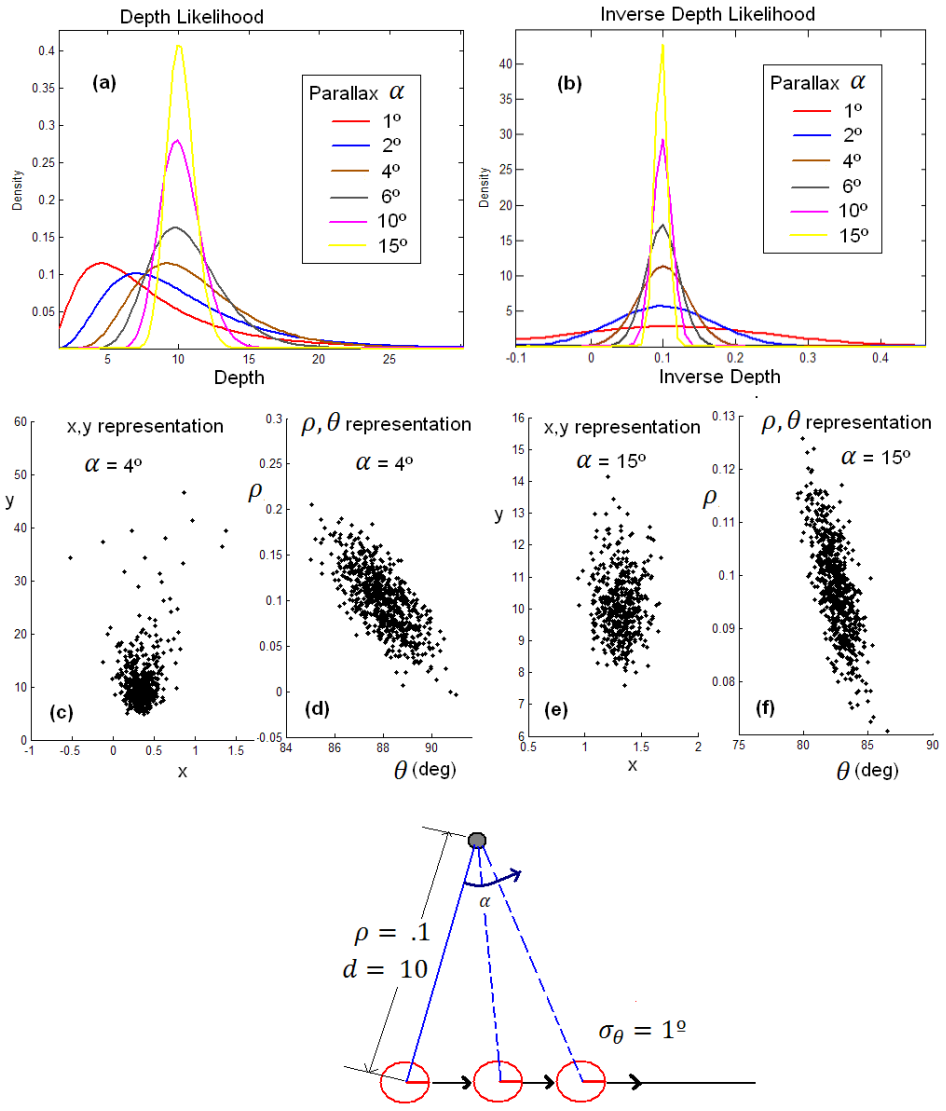


Figure 4.6 Simulation of a point reconstruction from observations with different parallax. The location of the vehicle is known. A Gaussian error σ_θ=1º (degrees) is introduced in bearings. Upper graphics show the evolution of the likelihood for depth and inverse depth as the parallax in the observation grows: In (a), the estimates of depth likelihood converge to a Gaussian-like shape, but the initial estimates are highly non-Gaussian, with heavy tails. In contrast, likelihoods of inverse depth (b) (abscissa in inverse meters) are nearly Gaussian, even for low parallax. Middle graphics illustrates the estimated location of the point, coded as Cartesian XY (c,e) and ρ,θ (d,f), for two different parallax. When parallax is equal to 4º note how reconstruction is Gaussian for ρ,θ parameterization (d) and non-Gaussian for the Cartesian XY parameterization (c).

In the simulation it can be clearly appreciated how the uncertainty can be represented Gaussian using the inverse depth parameterization over whole parallax range, whereas the Euclidean representation converges to a Gaussian-like shape only to the final estimates.

For the Euclidean representation, the parallax needed for the likelihood converges to a Gaussian-like shape, depends on the sensor noise, and thus noisier is the sensor more parallax is needed for convergence.

Moreover, as the far features do not exhibit parallax, then they cannot be coded in Euclidean parameterization. On the other hand, the inverse depth parameterization allows using linear techniques for the estimation process, thus allowing the direct use of Kalman Filtering techniques for estimating the features depth, even for far features.

4.3.4 Measurement Equation

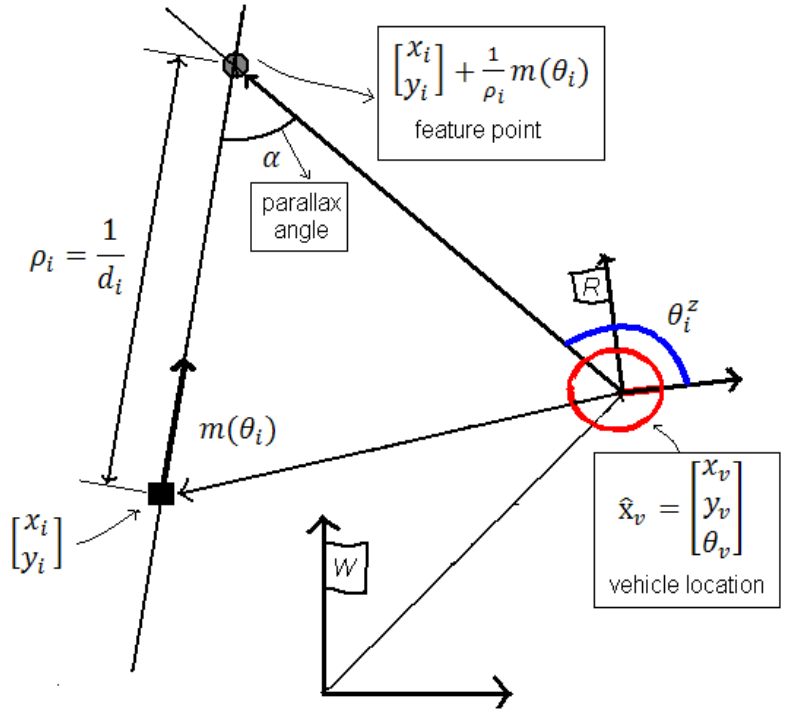


Figure 4.7 Feature parameterization and measurement equation.

The different locations of the vehicle, along with the location of the already mapped features, are used to predict the feature bearing θ^z_i (angle describing the direction of the feature respect to the vehicle, in the vehicle coordinate frame). The measurement equation $h_i(\hat{x}_v, \hat{y}_i)$ is used to predict the feature bearing.

In section 4.1.1 we presented a simple measurement equation expressed in Euclidean coordinates.

$$h_i(\hat{x}_v, \hat{y}_i) = atan2(y_i - y_v \quad , \quad x_i - x_v) - \theta_v \tag{4.18}$$

Now considering the change in the features parameterization (Note that we have included an index *i* for θ and ρ):

$$[x_i, y_i] \rightarrow [x_i, y_i, \theta_i, \rho_i] \tag{4.19}$$

Where the feature 4-dimension state vector (Equation (4.17)) models the 2D point located at (Figure 4.7):

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i) \tag{4.20}$$

Being x_i and y_i the vehicle center coordinates when the feature was observed the first time; and θ_i represent azimuth (respect to the world reference W) for the directional vector $m(\theta_i)$. The point depth d_i is coded by its inverse $\rho_i = 1/d_i$.

We can define a new measurement equation $h_i(\hat{x}_v, \hat{y}_i)$ for the inverse depth parameterization:

$$h_i = atan2\left(\frac{1}{\rho}\sin\theta_i + y_i - y_v, \frac{1}{\rho}\cos\theta_i + x_i - x_v\right) - \theta_v \tag{4.21}$$

Where x_v, y_v and θ_v are taken from the vehicle state vector $\hat{x}$ (Equation (4.5).

The new Jacobians for the observation model h_i are:

$$\nabla H_x = \begin{bmatrix} \frac{s}{c^2 u} & \frac{-1}{cu} & -1 \end{bmatrix} \tag{4.22}$$

And :

$$\nabla H_y = \begin{bmatrix} \frac{-s}{c^2 u} & \frac{1}{cu} & \frac{\frac{\cos\theta_i}{\rho c} + \frac{s\sin\theta_i}{\rho c^2}}{cu} & \frac{\frac{\sin\theta_i}{\rho^2 c} + \frac{s\cos\theta_i}{\rho^2 c^2}}{cu} \end{bmatrix} \tag{4.23}$$

Where:

$$\begin{aligned} u &= \left(1 + \frac{s^2}{c^2}\right) \quad \text{and} \\ c &= \frac{1}{\rho}\cos\theta_i + x_i - x_v \\ s &= \frac{1}{\rho}\sin\theta_i + y_i - y_v \end{aligned} \tag{4.24}$$

Note that the Jacobians has become a little more complex due to new parameterization.

4.3.5 2D Un-delayed Feature Initialization

In the un-delayed method presented in [1], transition from partially to fully initialized features need not to be explicitly tackled; this means that the feature is added to the map in its final representation since the first frame that was observed. Possibly the main benefit of this approach is that the feature provides information to the system since the beginning.

The initialization includes both the feature sate initial values and the covariance assignment. Despite the initial uncertainty regions covers a huge range depth [d_{min}, ∞] because the low linearization errors, due to the inverse depth parameterization, the uncertainty is successfully coded as Gaussian: once initialized, the feature is processed with the standard EKF prediction-update loop.

Using the inverse depth parameterization, while the feature is observed at low parallax, the feature will be used mainly to determine the vehicle orientation but the feature depth will be kept quite uncertain, including in its uncertainty region the even infinity; if the robot translation is able to produce a parallax big enough then the feature depth estimation will be improved.

In [1] the method is defined for a camera 3D context where no odometry information is available. In this section we adapt the equations defined in [1] for representing our current 2D vehicle context.

The initial location for the observed feature is defined as:

$$\hat{y}_{new} = \left[\hat{x}_i, \hat{y}_i, \hat{\theta}_i, \hat{\rho}_i\right]^{\mathrm{T}} \tag{4.25}$$

Where:

$$\begin{bmatrix}\hat{x}_i \\ \hat{y}_i\end{bmatrix} = \begin{bmatrix}x_v \\ y_v\end{bmatrix} \tag{4.26}$$

Are taken directly from the current vehicle state $\hat{\mathrm{x}}_{\mathrm{v}}$. and

$$\hat{\theta}_i = \theta_v + z_\theta \tag{4.27}$$

Is simply the addition of the current vehicle orientation θ_{v}, taken from the vehicle state $\hat{\mathbf{x}}_{\mathbf{v}}$, and the initial bearing measurement z_θ.

The covariance for $\hat{x_i}$, $\hat{y_i}$ and $\hat{\theta}_{\mathrm{i}}$ is derived from the measurement error covariance $\mathrm{R_j}$ and the state covariance estimate $\mathrm{P_k}$.

The initial value for $\hat{\rho}_{\mathrm{o}}$ is derived heuristically to cover in its 95% acceptance region a working space from infinity to a predefined close distance $\mathrm{d_{min}}$ expressed as inverse depth:

$$\left[\frac{1}{d_{min}}, 0\right] \quad \text{so:}$$
$$\hat{\rho}_0 = \frac{\rho_{min}}{2} \qquad \sigma_\rho = \frac{\rho_{min}}{4} \quad \rho_{min} = \frac{1}{d_{min}} \tag{4.28}$$

In [1] d_{min} = 1, $\hat{\rho}_o$ = 0.5, σ_ρ = 0.25.

The new system state $\hat{x}$ is conformed simply adding the new feature $\hat{y}_i$ to the final of the vector state:

$$\hat{x} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \end{bmatrix} \quad \hat{x}_{new} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \\ \hat{y}_i \end{bmatrix} \tag{4.29}$$

The state covariance after feature initialization is defined by:

$$P_{new} = \nabla Y \begin{pmatrix} P_k & 0 & 0 \\ 0 & R_j & 0 \\ 0 & 0 & \sigma_\rho^2 \end{pmatrix} \nabla Y^T \tag{4.30}$$

Where

$$R_j = {\sigma_z}^2 \tag{4.31}$$

Is simply the standard deviation of the bearing sensor and:

$$\nabla Y = \begin{pmatrix} I & & 0 \\ \frac{\partial y}{\partial x_v}, 0, \ldots, 0, & & \frac{\partial y}{\partial R_j} \end{pmatrix} \tag{4.32}$$

For the current vechicle-2D case the derivatives are simply:

$$\frac{\partial y}{\partial x_v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial y}{\partial h_{R_j}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.33}$$

Figure 4.8 shows a simulation for the Un-delayed Inverse Depth Feature Initialization. Graphics a,b,c and d illustrates a single initialization: When the feature is observed at first time, then it is initialized immediately in the system state, (a) with a huge uncertainty that codes a depth from the vehicle to the infinity (blue ellipses defines an uncertainty of 3σ). Note that the feature location has been initialized very near to the center of the vehicle. As the vehicle move to the right (b, c and d) the feature parallax increases, and the new incoming bearing measurements improves the feature estimation. Note how the estimate feature´s location is getting closer to its ground truth and the uncertainty is minimized. Graphics e,f,g and h illustrates the same vehicle´s trajectory but in this case the robot is capable of "seeing" four features.

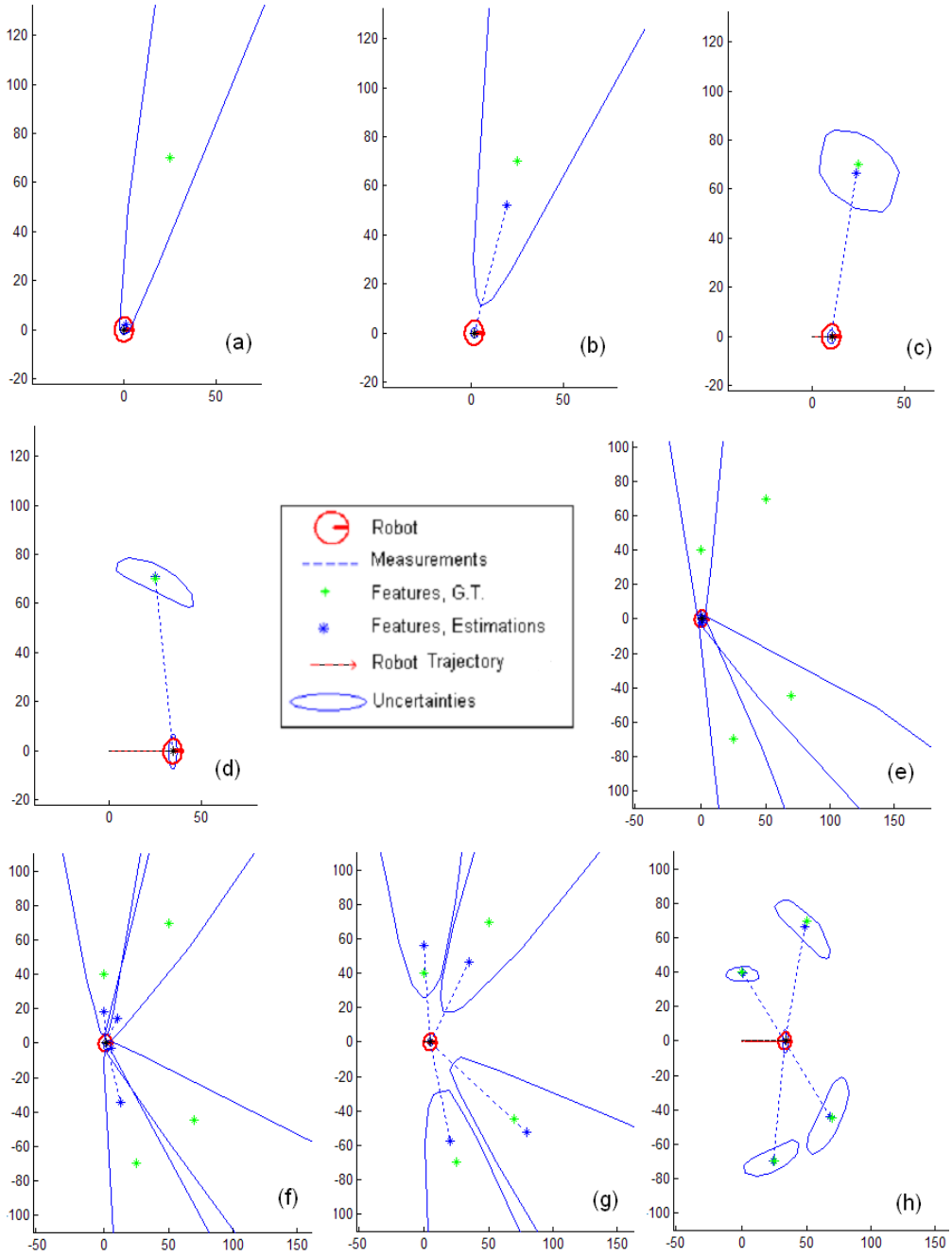


Figure 4.8 Un-delayed Feature Initialization: Initializing a single feature (graphics a,b,c and d), Initializing four features (graphics e,f,g and h).

At the end of the trajectory (h), note that the uncertainty of the closer feature is lower respect to the others. The accuracy of the feature estimation depends on its distance to the sensor. Closer features can be estimated better than the far ones.

4.3.6 Drawbacks for the Un-delayed Initialization

When the Un-delayed feature initialization is used, it often happens that the inverse depth becomes negative after a Kalman update. In this case the next predicted observation for this landmark is 180º off the observation and is completely inconsistent. It causes the divergence of the Kalman filter. This difficulty is not mentioned in the original paper [1].

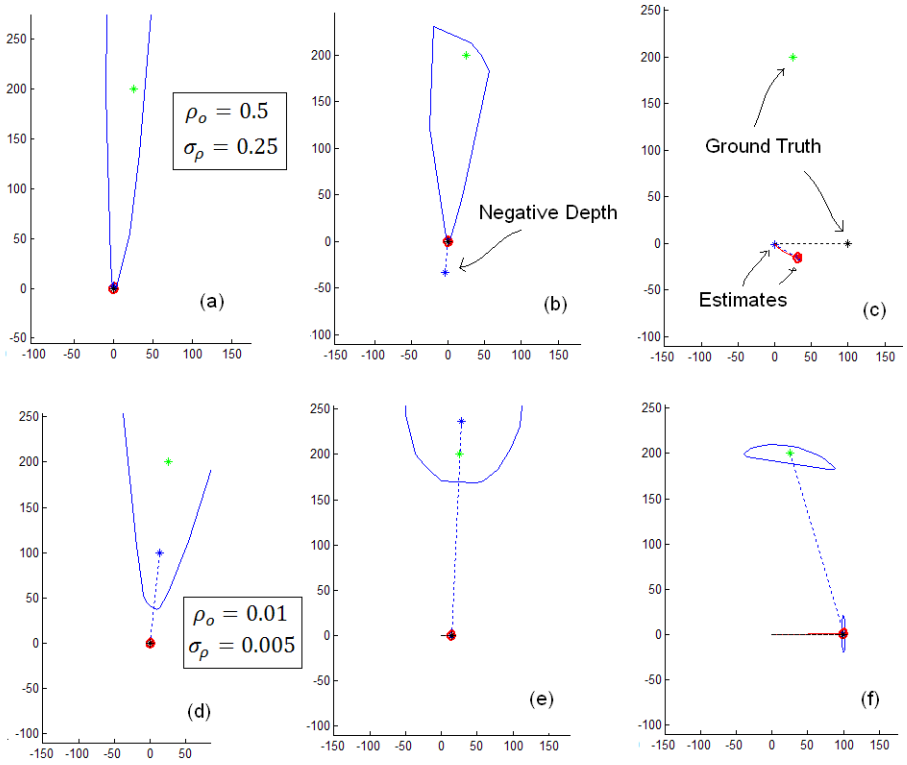


Figure 4.9 Two cases for a single Un-delayed Feature Initialization: Using ρ_o=0.5,σ_ρ=0.25 as the initial parameters (a,b and c graphics), and using ρ_o=0.01,σ_ρ=0.005 (d,e and f graphics).

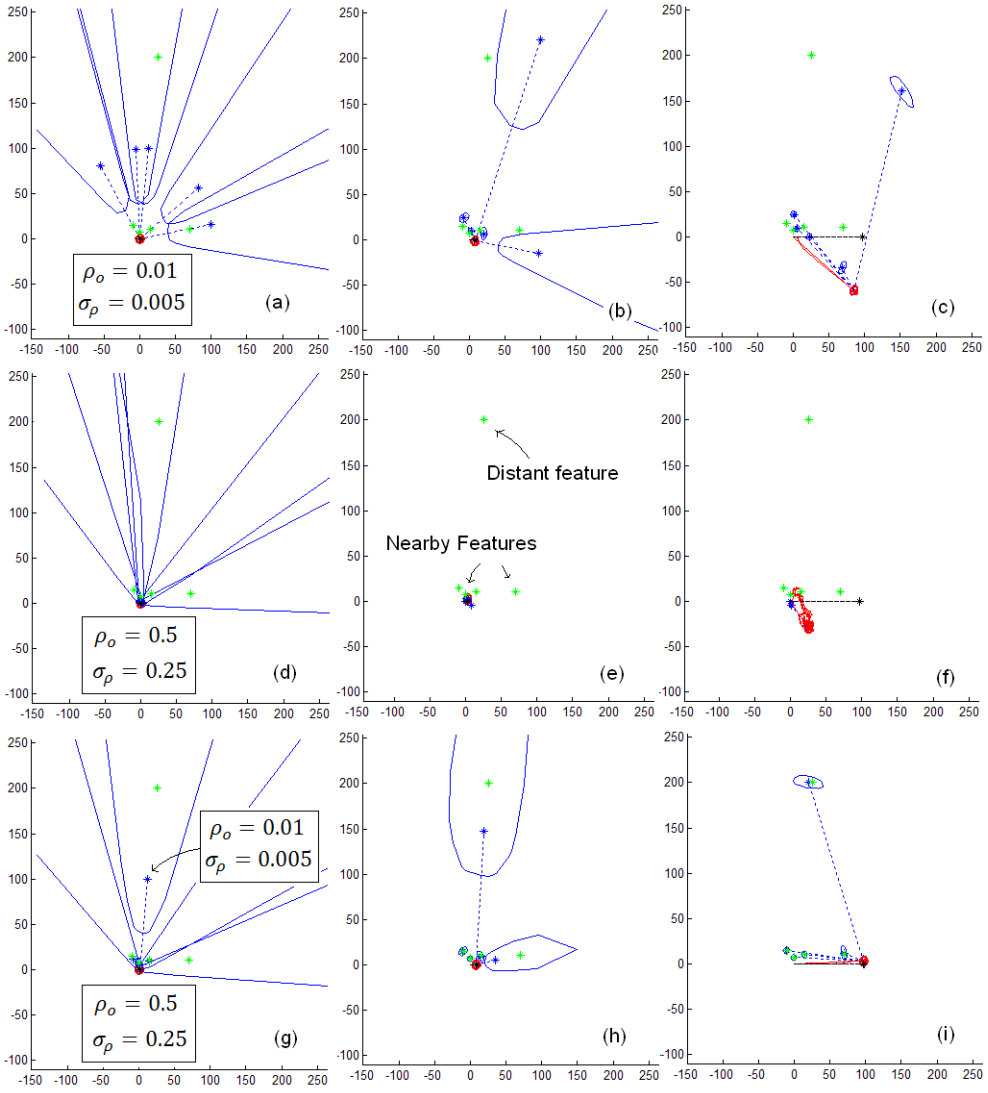


Figure 4.10 Un-delayed initialization for 5 features, (1 distant and 4 close); In the first sequence (upper graphics a,b and c) the initial parameter have been set for initializing the features at the middle between the vehicle and the feature ground truth ρ_o=0.01,σ_ρ=0.005 (a), at the end of the trajectory (c) there are a huge drift in the estimations. When the parameters are set for initializing the features near to the vehicle ρ_o=0.5,σ_ρ=0.25 (middle graphics c,d and f), the problems of divergence still remains (f). On the other hand, when we use different parameters for the distant and nearby features (Lower graphics g,h and i), then the filter convergence is improved (i).

The origin of the problem is to update the depth of the landmark with an observation which does not carry any information on this dimension. Then the observation noise predominates in the update of the depth: the landmark can get behind of the sensor.

Figure 4.9 (upper graphics a,b and c) shows a case where the features depth become negative after some Kalman update steps (b). In this case the default values for the initial parameters were used. At the end of the trajectory (c) note that neither the feature location nor the robot location converged to the real ones, due to the divergence of the Kalman Filter. For this simulation the 58% of the times there was a divergence.

On the other hand, if the initial parameters (ρ_o and σ_ρ) are tuned then percentage of divergence decreases, In the lower graphics (d,e and f) the initial depth has been set in order to initializing the feature at the middle of the distance between the robot and the landmark (d), at the end of the trajectory (f), both robot and landmark location have been successfully estimated. Making this change in the initial values for the simulation, practically a 100% of effectiveness was archived.

But what happens, if there are some distant features and others nearby to the vehicle; in order to validate if as result of modifying the initial parameters ρ_o and σ_ρ the robustness of the filter is increased, we perform another experiment (Figure 4.10). In this case, additional to the single landmark used in the previous experiment, we add some nearby features.

In upper graphics (a,b and c) when the initial parameters (ρ_o=0.01,σ_ρ=0.005) were set in order to initialize the features at the middle of the distance between the vehicle and the more distant feature, and thus far respect to the nearby features (a) we found in almost every cases a huge drift in the estimates (c), in this case it can be appreciated that initializing the features far to the vehicle in order to avoid negative depths fails if there are some closest landmarks. On the other hand if the original values (ρ_o=0.5,σ_ρ=0.25) are used, (middle graphics d,e and f) then the convergence issue remains (approx 50%) (f), due to the influence of the distant feature. In this case if the distant feature is removed from the map (Not illustrated) then a percentage of convergence of 80% is archived.

In the last series of simulations (lower graphics g,h and i) we combine both initial values: ρ_o=0.01,σ_ρ=0.005 for the distant landmark and ρ_o=0.5,σ_ρ=0.25 for the all the nearby ones. In this case an effectiveness of 90% was archived for the algorithm.

From all the previous experiments is clear that values for the initial parameters ρ_o and σ_ρ play an important role in the robustness of the method. In this context, we improve the results of the method making a supervised tuning for the initial parameters. Nevertheless, it is clear for a real method application that supervised tuning is impractical.

4.3.7 2D-Delayed Feature Initialization

In last section, it was seen that making a supervised tuning of initial parameters, improve the robustness of the Un-delayed Inverse Depth feature initialization. On the other hand the supervised tuning violates the principle of autonomy for an SLAM algorithm.

In this context, if the robustness of initialization process wants to be improved, then an expected idea is to gather some previous information about the feature depth before to be initialized in the map. For a bearing sensor, gather information about feature depth, can't be done in a single measurement, in that sense, delayed approaches obtain some depth information about the feature priors its initialization in the map.

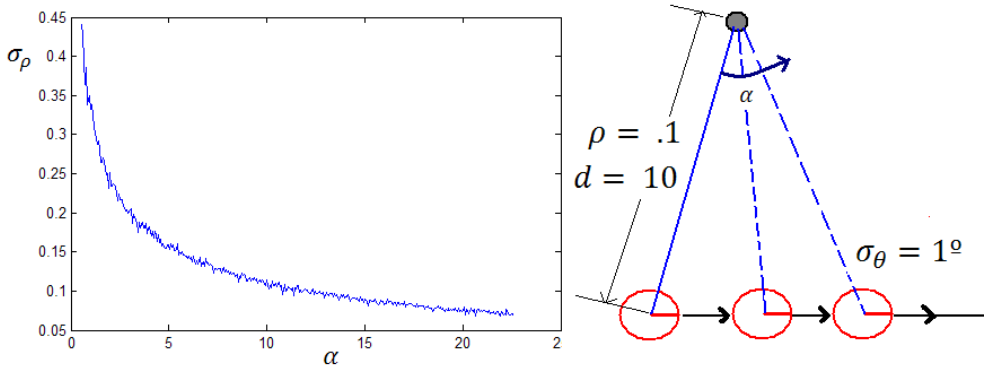


Figure 4.11 Simulation for the decrementing of uncertainty in feature depth σ_ρ respect with the increase of the parallax angle. Note that a few degrees parallax is enough to reduce the uncertainty in the estimation.

In an Un-delayed approach when a feature is added to the map when it was first observed, the feature depth is modeled for a huge uncertainty, in that sense the new feature does not provide information in the depth dimension to system, however at this stage the feature provides information about the sensor orientation. A benefit of the Un-delayed approach is that the feature provides information to the system since the beginning. On the other hand in Figure 4.11 it can be observed that a few parallax degrees are enough for reducing significantly the depth uncertainty, of course the parallax depend on the feature distance and the movement of the vehicle. For nearby indoor features, only a few centimeters of movement will be sufficient, distant features may require many meters or even kilometers of motion before parallax is observed.

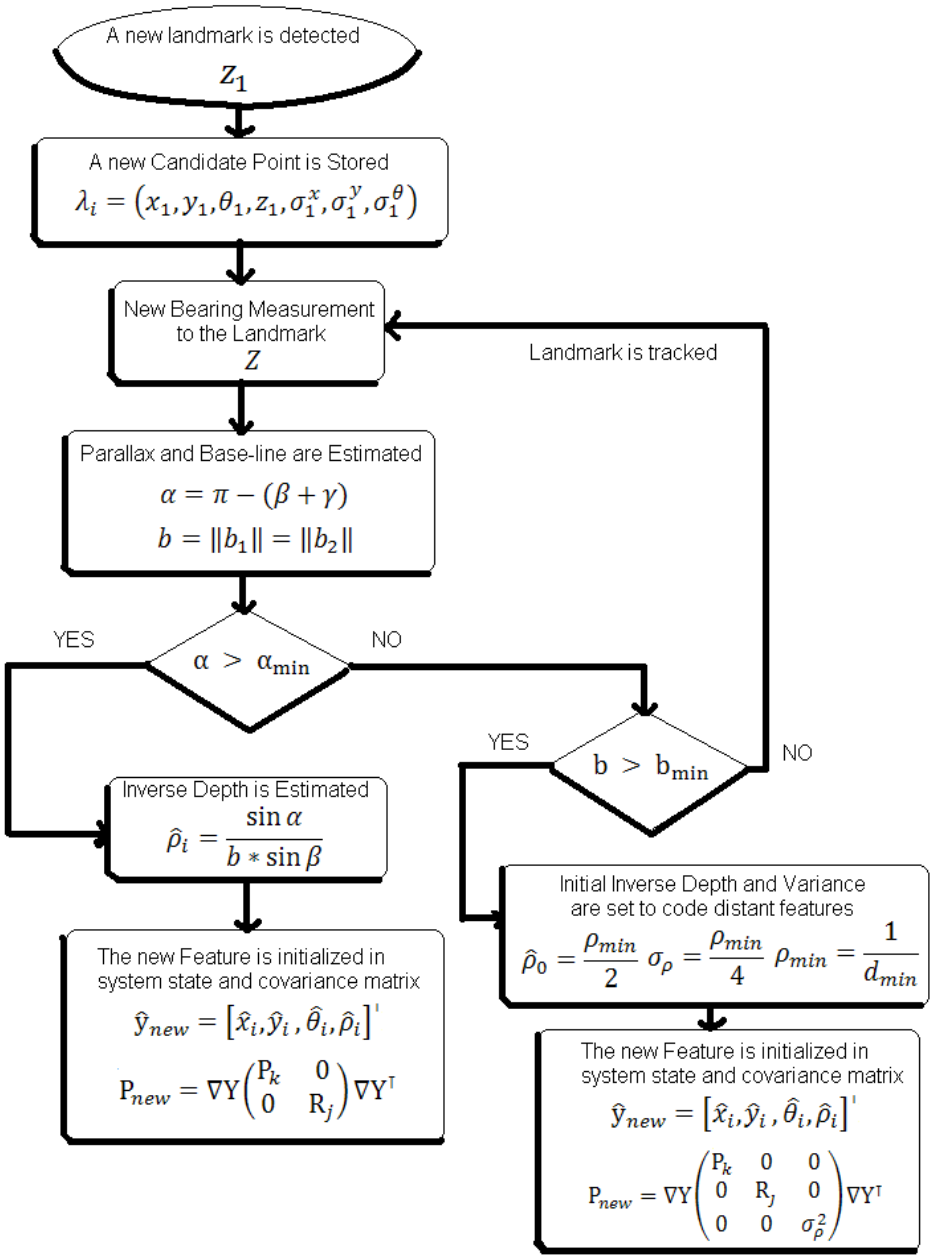


Figure 4.12 Delayed Feature Initialization Flow chart.

For many applications we think useful to wait until the vehicle movement produces some parallax in the features in order to gather depth information for improving the robustness of the inverse depth initialization. In this section we describe our proposed features initialization for bearing-only SLAM called *Delayed Inverse Depth Feature Initialization*.

The key idea of the proposed method consist in (i) estimating dynamically an initial feature depth ρ_o close to the real one, and (ii) incorporating directly the uncertainty related to the estimation process of ρ_o in the system covariance matrix when the feature is initialized.

A general description is: when a feature is first observed, the bearing measurement and some part of the system state $\hat{\mathrm{x}}$ and covariance matrix P are stored, later, while the vehicle moves (producing parallax), the stored information together with the current system state and bearing measurement is used to approximated the parallax produced in the feature, this process is made until the parallax is bigger than some threshold , if this condition is archived then an estimation of the feature depth and its uncertainty is made. Finally the feature is initialized in the map using the estimated depth and uncertainty. Figure 4.12 show a flow chat for our proposed Delayed Initialization method. Hereinafter will be explained each one of the chart modules.

In [104], the idea of storing some part of the system state is used, but in this case for each new measurement some part of the state are retained, on the other hand, our approach only requires to store a minimum number of parameters when the feature is first observed. Moreover in [104] due to the non-linearity in the measurement equation, (Euclidean parameterization is used) a criteria (Kullback distance) is used for determining if the estimation is well-conditioned (Gaussian). The complexity of the sampling method proposed to evaluate this distance is quite high making computationally expensive. As was seen in Section 4.3.3 the inverse depth parameterization highly improves the linearity in the measurement equation, therefore our approach does not need a measure for evaluating if the depth estimation is well-conditioned (Gaussian). In that sense our approach uses a simple approximation of the parallax in order to consider a new feature initialization.

4.3.8 Candidate Points

When a feature is detected the first time k, some part of the current state $\hat{\mathrm{x}}$ and covariance matrix P together with the sensor measurement are stored, this data λ is conformed by:

$$\lambda_i = \left(x_1, y_1, \theta_1, z_1, \sigma_1^x, \sigma_1^y, \sigma_1^\theta\right) \tag{4.34}$$

We call λ_i as candidate points. The values x_1, y_1 and θ_1 represent the current robot position; $\sigma_1^{\,x}$, $\sigma_1^{\,y}$ and $\sigma_1^{\,\theta}$ represent their associated variances taken from the state

covariance matrix P_k and z_1 is the first bearing measurement to the landmark in the vehicle frame R. In subsequent instants *k*, the feature is tracked until a minimum parallax threshold α_{min} is reached.

4.3.9 Cases where there is not Parallax

Some common drawback attached to the delayed methods is that very distant features cannot be added to the map, because a condition (used for ensure that some depth information has been obtained from the feature) has to be reached in order to initialize it, in that sense for features at the "infinity" will be impossible to obtain depth information and therefore the condition will be never reached. In our case a feature at the "infinity" will never produce a minimum parallax α_{min}. A nice attribute of the Un-delayed initialization [1] is that features at infinity are included and maintained in the map with a huge depth uncertainty. Very distant features are useful because provides orientation information.

Some delayed methods can not include very distant features because Euclidean parameterization cannot code Gaussian a high uncertainty in the depth dimension. On the other hand Due to the inverse depth parameterization it is possible to code Gaussian acceptance regions for features from nearby to infinity. Moreover, in the "delayed" period it is possible to expect that a feature is very distant, if after a long movement there is not parallax in the feature.

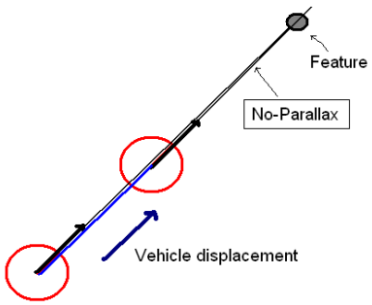


Figure 4.13 The vehicle movement will not produce parallax in the feature if it points toward the direction of the landmark.

In our approach, if very distant features wants to be included, then if after a minimum base-line b_{min} (de base-line b defines the translation from the location when feature was first seen and the current vehicle location) a minimum parallax α_{min} has not been reached, then it is assumed that feature is located distant from the vehicle, in this case the feature is initialized in a similar manner that the un-delayed initialization with a predefined ρ_o and σ_ρ tuned for initializing the feature far from the vehicle.

There is an especial case when a non-distant feature does not produce parallax: If the vehicle is moving exactly in the direction of the feature, this feature will never produce parallax (Figure 4.13). This case could be a shortcoming for the un-delayed method because there are not previous assumptions neither vehicle movement nor feature locations when the feature is initialized. In this case the observation noise could predominate over the update of the depth, producing negative depths. In our experiments it is assumed that the vehicle movement will produce some parallax for initializing the features, nevertheless in our delayed context is possible to comparing the direction of the vehicle since the feature was first observed respect to the direction of the candidate point λ, in this case if it is determined that the vehicle moves toward the landmark then the candidate point λ could not be considered as a new feature map.

4.3.10 Estimating Parallax

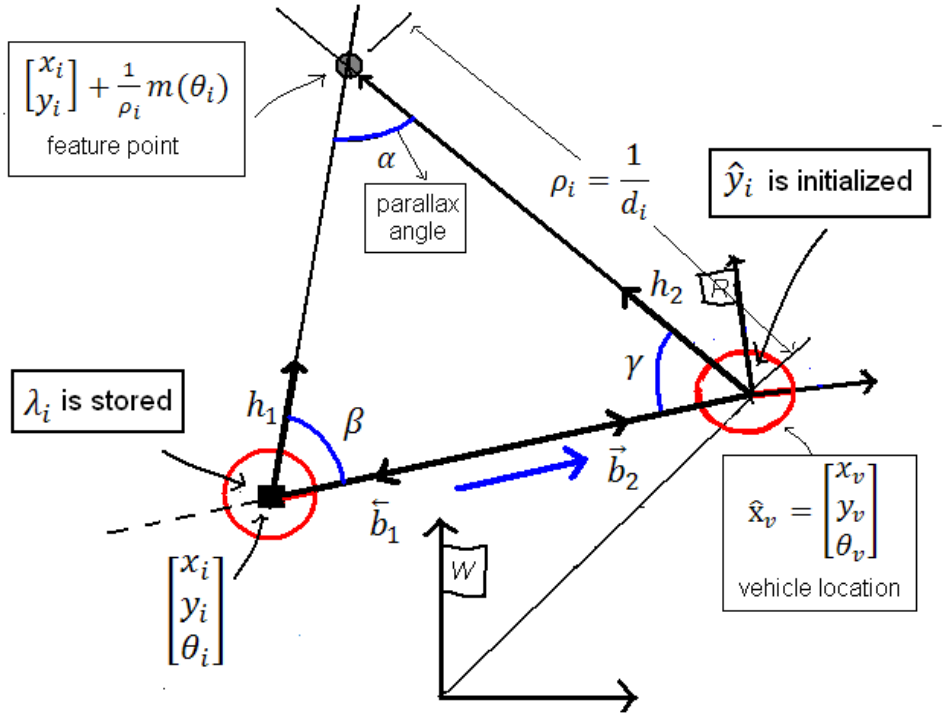


Figure 4.14 Delayed feature initialization.

To infer the depth of a feature, the robot must observe it as it freely moves through its environment, each time estimating the angle (bearing) from the landmark to its center. In our approach if a candidate point λ_i shows a minimum parallax α_{min} then it will initialized as a new feature $\hat{y}_i$.

At this stage, the uncertainty of the measurements is not considered, and the parallax α is approximated using (i) the base line b, (ii) λ_i using its associated data (x_1, y_1, θ_1,z_1), and (iii) the current state and bearing measurement (x_v, y_v, θ_v,z).

For each candidate point λ_i, every time a new bearing measurement z is available, then the parallax angle α can be estimated by (Figure 4.14):

$$\alpha = \pi - (\beta + \gamma) \tag{4.35}$$

The angle β is determined by the directional unitary vector h_1 and the vector b_1 defining the base-line b in the direction of the robot trajectory by:

$$\beta = cos^{-1}\left(\frac{h_1 \cdot b_1}{\|h_1\|\|b_1\|}\right) \tag{4.36}$$

Where $(h_1 \cdot b_1)$ is the dot product between h_1 and b_1. The directional vector h_1, expressed in the absolute frame W, points from the robot location to the direction when the landmark was observed for the first time, and is computed using the data stored in λ_i denoting the bearing z_i:

$$h_1 = \begin{bmatrix}\cos(\theta_1 + z_1)\\ \sin(\theta_1 + z_1)\end{bmatrix} \tag{4.37}$$

b_1 is the vector representing the robot base-line between the robot center position x_1, y_1 stored in λ_i where the point was first detected and the current robot center x_v, y_v:

$$b_1 = [(x_v - x_1), (y_v - y_1)] \tag{4.38}$$

The angle γ is determined in a similar way as β but using the directional unitary vector h_2 and the vector b_2 defining the base line in the opposite direction of the camera trajectory by:

$$\gamma = cos^{-1}\left(\frac{h_2 \cdot b_2}{\|h_2\|\|b_2\|}\right) \tag{4.39}$$

The directional vector h_2 expressed in the absolute frame W, is computed in a similar way as equation (4.37) but using the current robot position $\hat{x}_v$ and the current measurement z. b_2 is equal to b_1 but pointing to the opposite direction:

$$h_2 = \begin{bmatrix}\cos(\theta_v + z)\\ \sin(\theta_v + z)\end{bmatrix} \tag{4.40}$$

$$b_2 = [(x_1 - x_v), (y_1 - y_v)] \tag{4.41}$$

The base-line b is the module of b_2 or b_1:

$$b = \|b_1\| = \|b_2\| \tag{4.42}$$

4.3.11 Feature Initialization in state and covariance matrix

If $\alpha > \alpha_{min}$ then λ_i is initialized as a new feature map $\hat{y}_i$. The threshold α_{min} can be established depending on the acuity of the bearing sensor. In Figure 4.11 it can be appreciated that depth uncertainty is reasonably well minimized when $\alpha = 10^o$.

The new feature $\hat{y}_{new}$ is determined by:

$$\hat{y}_{new} = \left[\hat{x}_i, \hat{y}_i, \hat{\theta}_i, \hat{\rho}_i\right]^T \tag{4.43}$$

Where the three first elements are determined in the same manner than the un-delayed initialization method:

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \end{bmatrix} \tag{4.44}$$

Are taken directly from the current vehicle state $\hat{\mathbf{x}}_\mathbf{v}$. and

$$\hat{\theta}_i = \theta_v + z \tag{4.45}$$

Is simply the addition of the current vehicle orientation θ_v, taken from the vehicle state $\hat{\mathbf{x}}_\mathbf{v}$, and the bearing measurement z.

In our delayed approach a dynamical estimation of the inverse depth $\hat{\rho_i}$ is made prior to be added to the map instead of the initial fixed depth $\hat{\rho_o}$ used in the Un-delayed method (4.28). $\hat{\rho_i}$ is derived from the sine law:

$$\hat{\rho}_i = \frac{\sin\alpha}{b * \sin\beta} \tag{4.46}$$

The new system state $\hat{x}$ is conformed simply adding the new feature $\hat{y}_{new}$ to the final of the vector state:

$$\hat{x} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \end{bmatrix} \quad \hat{x}_{new} = \begin{bmatrix} x_v \\ y_1 \\ y_2 \\ \hat{y}_{new} \end{bmatrix} \tag{4.47}$$

The variance σ_ρ for the inverse depth ρ is derived now from the initialization process, instead of a variance predefined heuristically as it was made in the un-delayed method, therefore the covariance for the new feature $\hat{y}_{new}$ is derived from the error diagonal covariance matrix R_j measurement and the state covariance matrix P.

$$R_j = \begin{bmatrix} \sigma_1^x & 0 & 0 & 0 & 0 \\ 0 & \sigma_1^y & 0 & 0 & 0 \\ 0 & 0 & \sigma_1^\theta & 0 & 0 \\ 0 & 0 & 0 & {\sigma_z}^2 & 0 \\ 0 & 0 & 0 & 0 & {\sigma_z}^2 \end{bmatrix} \tag{4.48}$$

Note that R_j is now conformed by the error variance of the bearing sensor σ_z (one for each bearing estimation z_1 and z) and the variances stored in λ_i (σ_1^x, σ_1^y and σ_1^θ).

The new state covariance matrix, after initialization, is:

$$\mathrm{P}_{new} = \nabla \mathrm{Y} \begin{pmatrix} \mathrm{P}_k & 0 \\ 0 & \mathrm{R}_j \end{pmatrix} \nabla \mathrm{Y}^\intercal \tag{4.49}$$

Note that a difference from the Un-delayed method (4.30), there is not an implicit initial uncertainty in depth σ_ρ (4.28). In our method the complete covariance for the new feature is estimated by the initialization process.

4.3.12 Jacobian ∇Y

The Jacobian $\nabla \mathrm{Y}$ for the initialization process is:

$$\nabla \mathrm{Y} = \begin{pmatrix} \mathrm{I} & & 0 \\ \dfrac{\partial y}{\partial \hat{x}_v}, & 0, \ldots, 0, & \dfrac{\partial y}{\partial R_j} \end{pmatrix} \tag{4.50}$$

Where I is an identity matrix with the same dimension of $\mathrm{P_k}$, $\partial y / \partial x_\mathrm{v}$ are the derivatives of the initializations equations with respect to the vehicle state $\hat{\mathrm{x}}_\mathrm{v}$ and $\partial y / \partial R_\mathrm{j}$ the derivatives respect to the parameters of the covariance matrix R_j.

$$\frac{\partial y}{\partial x_v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ & \dfrac{\partial \rho}{\partial \hat{x}_v} & \end{bmatrix} \tag{4.51}$$

$$\frac{\partial y}{\partial R_j} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ & & \dfrac{\partial \rho}{\partial R_j} & & \end{bmatrix} \tag{4.52}$$

Note that derivatives of the initialization $\hat{\mathrm{y}}_\mathrm{new}$ (4.43) respect to the three first parameters (x_i, y_i and θ_i) are very simple. On the other hand, the derivatives respect to the inverse depth parameter ρ are complex due to the several functions used to estimate it, in this case the chain rule is used in order to make simplest the expressions.

$$\frac{\partial \rho}{\partial \hat{x}_v} = \left[\frac{\partial \rho}{\partial x_v}, \ \frac{\partial \rho}{\partial y_v}, \ \frac{\partial \rho}{\partial \theta_v}\right] = \frac{\partial \rho}{\partial \beta}\frac{\partial \beta}{\hat{x}_v} + \frac{\partial \rho}{\partial \gamma}\frac{\partial \gamma}{\hat{x}_v} + \frac{\partial \rho}{\partial b}\frac{\partial b}{\hat{x}_v} \tag{4.53}$$

$$\begin{aligned}\frac{\partial \rho}{\partial R_j} &= \left[\frac{\partial \rho}{\partial x_1}, \frac{\partial \rho}{\partial y_1}, \frac{\partial \rho}{\partial \theta_1}, \frac{\partial \rho}{\partial z_1}, \frac{\partial \rho}{\partial z_2}\right] \\ &= \frac{\partial \rho}{\partial \beta}\frac{\partial \beta}{R_j} + \frac{\partial \rho}{\partial \gamma}\frac{\partial \gamma}{R_j} + \frac{\partial \rho}{\partial b}\frac{\partial b}{R_j}\end{aligned} \tag{4.54}$$

Where:

$$\frac{\partial \rho}{\partial \beta} = -\frac{\sin \gamma}{b} - \frac{\cos^2 \beta \sin \gamma}{b \sin \beta} \tag{4.55}$$

$$\frac{\partial \rho}{\partial \gamma} = \frac{\cos(\beta + \gamma)}{b \sin \beta} \tag{4.56}$$

$$\frac{\partial \rho}{\partial b} = -\frac{\sin(\beta + \gamma)}{b^2 \sin \beta} \tag{4.57}$$

And:

$$\frac{\partial \beta}{\hat{x}_v} = \left[\frac{(y_1 - y_v)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sc)^{\frac{1}{2}}}, \frac{(x_v - x_1)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sc)^{\frac{1}{2}}}, \ 0\right] \tag{4.58}$$

$$\frac{\partial \beta}{R_j} = \left[\frac{(y_v - y_1)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sc)^{\frac{1}{2}}}, \ \frac{(x_1 - x_v)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sc)^{\frac{1}{2}}}, \ -\frac{c^{\frac{1}{2}}s^{\frac{1}{2}}}{(sc)^{\frac{1}{2}}}, \ -\frac{c^{\frac{1}{2}}s^{\frac{1}{2}}}{(sc)^{\frac{1}{2}}}, \ 0\right] \tag{4.59}$$

$$\frac{\partial \gamma}{\hat{x}_v} = \left[\frac{(y_v - y_1)q^{\frac{1}{2}}}{s^{\frac{1}{2}}(sq)^{\frac{1}{2}}}, \ \frac{(x_1 - x_v)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sq)^{\frac{1}{2}}}, \ \frac{q^{\frac{1}{2}}s^{\frac{1}{2}}}{(sq)^{\frac{1}{2}}}\right] \tag{4.60}$$

$$\frac{\partial \gamma}{R_j} = \left[\frac{(y_1 - y_v)q^{\frac{1}{2}}}{s^{\frac{1}{2}}(sq)^{\frac{1}{2}}}, \ \frac{(x_v - x_1)c^{\frac{1}{2}}}{s^{\frac{1}{2}}(sq)^{\frac{1}{2}}}, 0, 0, \frac{q^{\frac{1}{2}}s^{\frac{1}{2}}}{(sq)^{\frac{1}{2}}}\right] \tag{4.61}$$

$$\frac{\partial b}{\hat{x}_v} = \left[\frac{x_v - x_1}{s^{\frac{1}{2}}}, \quad \frac{y_v - y_1}{s^{\frac{1}{2}}}, \quad 0\right] \tag{4.62}$$

$$\frac{\partial b}{R_j} = \left[\frac{x_1 - x_v}{s^{\frac{1}{2}}}, \ \frac{y_1 - y_v}{s^{\frac{1}{2}}}, \quad 0, \quad 0, \quad 0\right] \tag{4.63}$$

Being:

$$s = {x_v}^2 - 2x_v x_1 + {x_1}^2 + {y_v}^2 - 2y_v y_1 + {y_1}^2 \quad (4.64)$$

$$c = \cos(z_1 + \theta_1)^2 + \sin(z_1 + \theta_1)^2 \quad (4.65)$$

$$q = \cos(\theta_v + z)^2 + \sin(\theta_v + z)^2 \quad (4.66)$$

4.3.13 Simulations Results

The proposed Delayed Inverse Depth Feature Initialization was tested performing several simulations. Figure 4.15 shows the initialization for a single feature; when the landmark is first observed (a), a candidate point λ_i is stored composed of parameters taken from the system state and covariance matrix and the current bearing measurement z_1. While the robot moves making new bearing measurements z, the parallax α is estimated repeatedly (b,c and d). If the parallax $\alpha > \alpha_{min}$ (d) then a new feature $\hat{y}_i$ is initialized in the system state and covariance matrix (e). In this simulation a $\alpha_{min} = 20^o$ was used for visual clarity but lower threshold can be used, in subsequent simulations $\alpha_{min} = 10^o$ is used. In (e) note that the feature has been initialized very near respect its ground truth position and its 3σ uncertainty covers a smallest area respect to the huge initial uncertainty coded by the Un-delayed approach. At the end of the vehicle trajectory (f) the estimated feature location has been improved and its uncertainty minimized.

Figure 4.17 shows three different cases: In the first sequence (plots a,b and c) four near features are initialized. In the second sequence (plots d,e and f) one landmark has been set very far respect to the vehicle trajectory, in this case when the vehicle has moved 100 units, the landmark ,due to its remoteness, does not produce enough parallax for being initialized as a new feature in the map using the α_{min} condition (e), therefore the feature is initialized with an initial fixed depth $\hat{\rho}_o$ and fixed uncertainty σ_ρ when the base-line is bigger than a b_{min} = 100 units (f).

Also note at the end of the trajectory, that the uncertainty of the distant feature remains high. In the third sequence (plots h,I and j) the vehicle moves in a s-like trajectory through twenty five random-location landmarks, in (j) not all the uncertainties of the initialized features has been fully minimized due to the trajectory followed by the vehicle. But the estimated map and trajectory are very good.

Figure 4.18 shows a simulation of a differential drive robot equipped with a bearing-only sensor, in the same simulated environment used in experiments of Section 3.4.9. and Section 4.1.2, but in the current case using the proposed Delayed Inverse Depth Feature Initialization.

The sequence (plots a,b,c,d,e and f)shows the typical SLAM behavior; In (c) The robot location and map estimates shows a drift respect to their ground truth. In (d) the vehicle has return near to its initial position re-detecting previous mapped features and

thus minimizing the drift in vehicle and map estimates. Also note in (d) when the robot has completed its first lap that some feature`s uncertainties remain high. At the end of the simulation (f) the map and vehicle location have been estimated consistently.

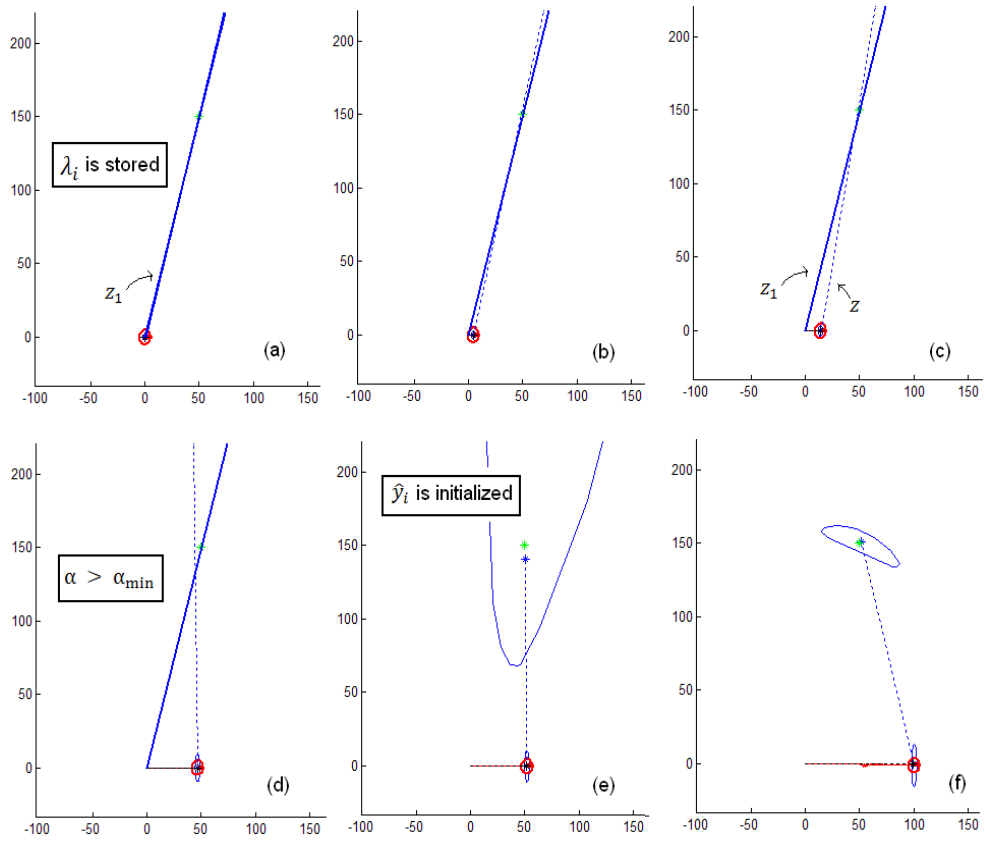


Figure 4.15 Delayed Inverse Depth Feature Initialization: Note that the feature has been near to its real position and its uncertainty (blue ellipses) covers a smallest region respect to the huge uncertainty used in the Un-delayed approach.

4.3.14 Comparing Un-delayed and Delayed Methods

For comparing the performance of both Un-delayed and Delayed methods, we have realized the same experiments showed Section 4.3.6 but now using our proposed method. Figure 4.16 illustrates the results of two simulations, in sequence (a,b and c) a single landmark has been set relatively far from the initial vehicle location. In section 4.3.6 it was seen that if initial feature depth was set near to the vehicle location, then could happen that the feature depth become negative after some Kalman update steps, causing

the filter divergence. On the other hand, since the Delayed Method realizes depths estimations priors to the state and covariance initialization phase, the feature is initialized very near to its ground truth with lower associated uncertainty (b). At the end of the simulation (c) the landmark position was estimated correctly. Using the Delayed method, and effectiveness of 100% was achieved for this experiment, against the 42% achieved by the Un-delayed method (using the default initial parameters).

For the second sequence (d,e and f) three very near landmarks were added, in order to test both initialization methods in a context with very near and very distant landmarks. With the Un-delayed method an effectiveness of 50% was achieved using the default initial parameters and 90% making a differentiated tuning for the distant and near features. On the other hand with the Delayed method a success of 100% was achieved. Even if the distant feature is removed from the experiment (not illustrated) an effectiveness of 100% is achieved compared with the 80% for the Un-delayed method.

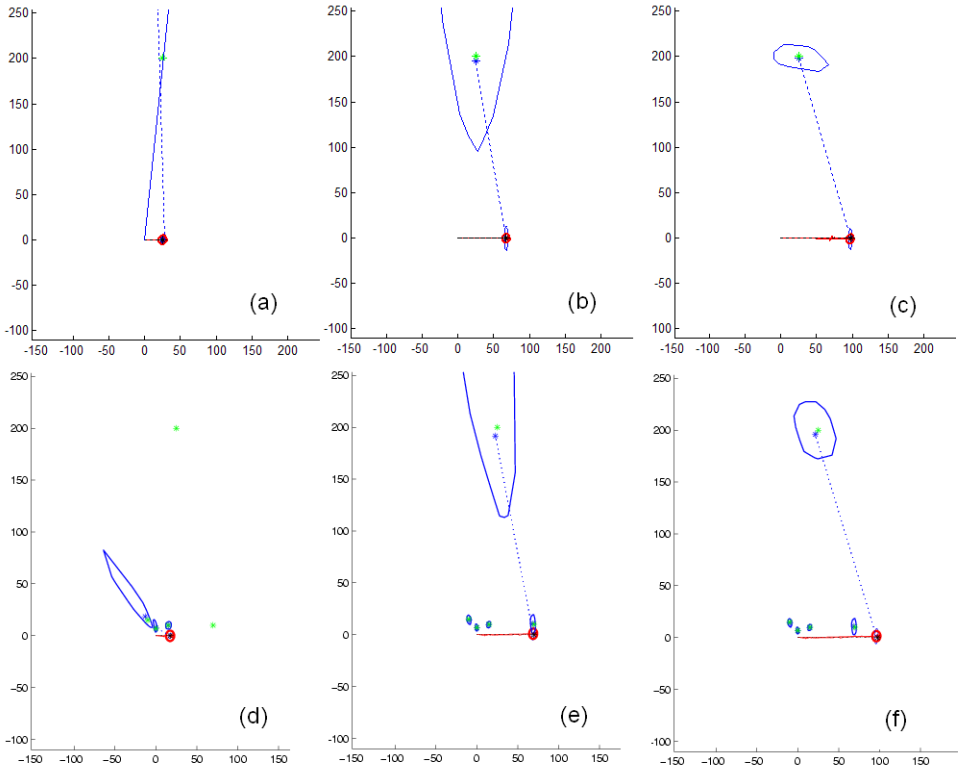


Figure 4.16 Similar experiments were performed with Un-delayed and Delayed Inverse depth Feature Initialization. The experimental result show that the percentage of effectiveness was clearly increased using the Delayed approach.

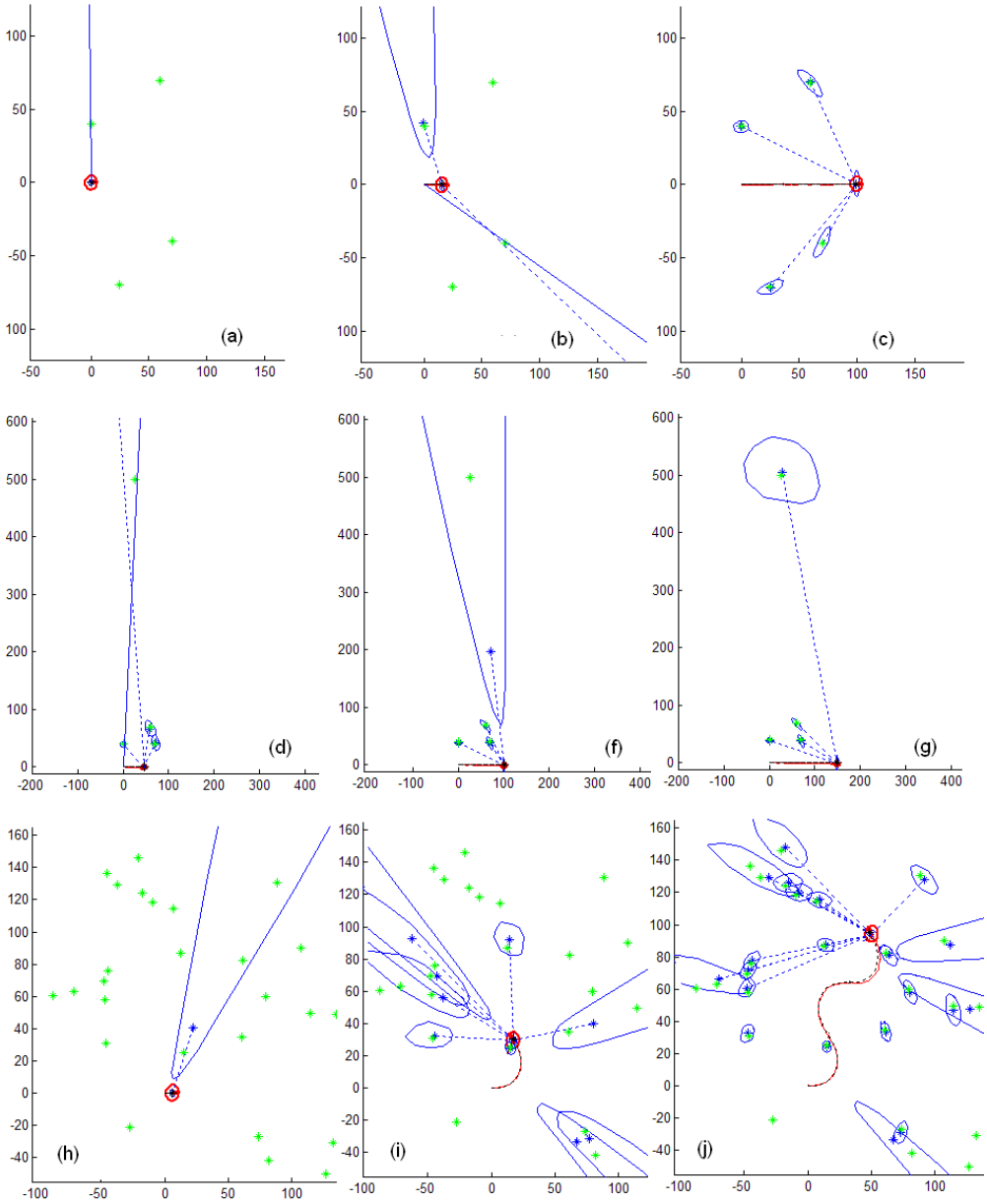


Figure 4.17 Delayed Inverse Depth Feature Initialization. Three cases: (i) Four near features (a,b,c). (ii) Three near features and a single very distant feature (d,f,g), in this case the distant feature has not produced enough parallax and therefore has been initialized using fixed initial inverse depth and uncertainty when the vehicle has reached a minimum base-line. (iii) Twenty five random features.

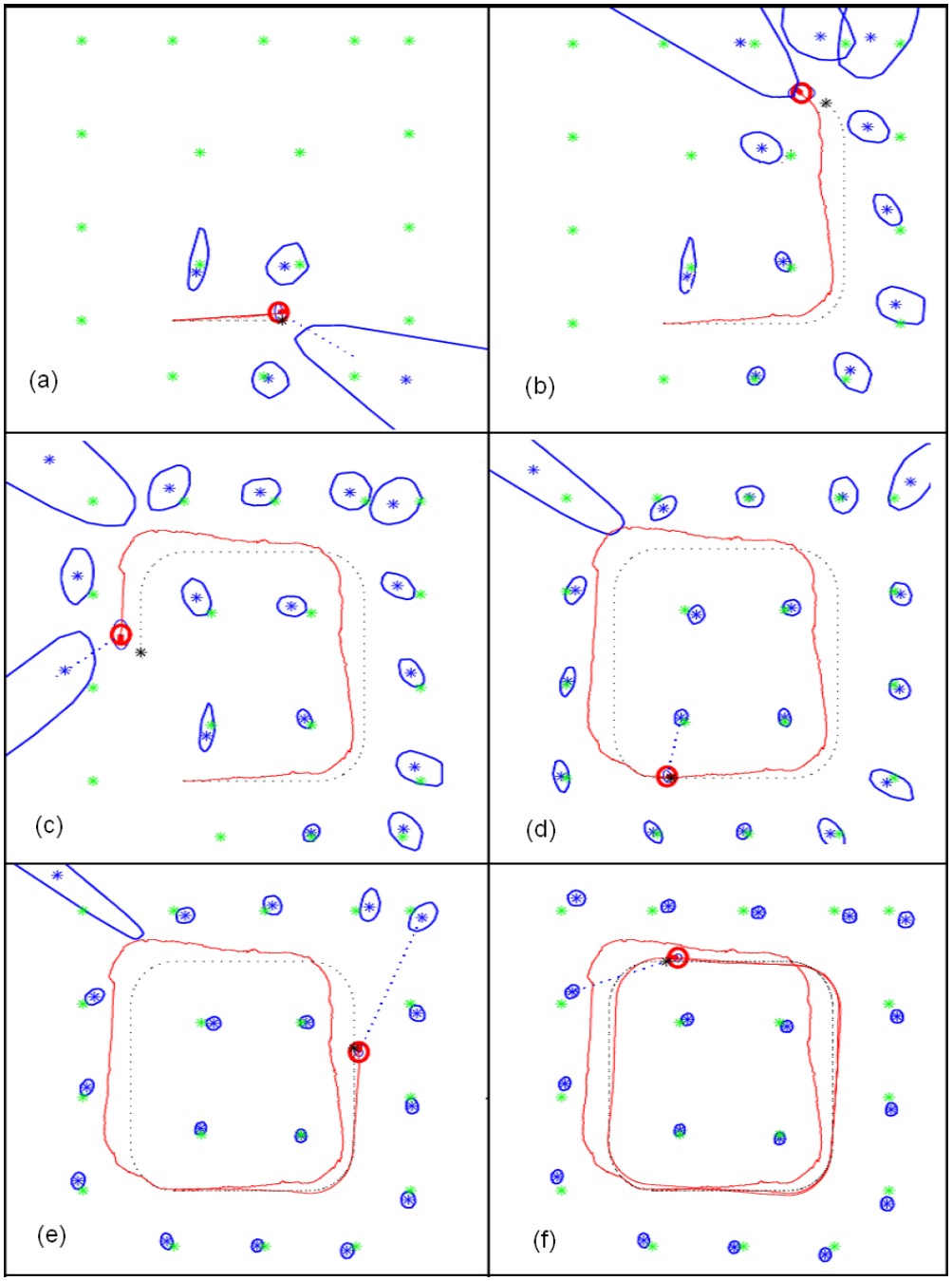


Figure 4.18 Bearing Only SLAM with Delayed Inverse Depth Feature Initialization. In a simulated corridor.

4.4 SSLAM: Sound-based SLAM

Usually the Bearing-Only SLAM has been associated with Vision-based SLAM systems, perhaps because Cameras are by far the most popular bearing sensor used in robotics. On the other hand the use of the hearing sense in localization and mapping has been much less explored. In this case, sound sources (artificial or natural) are used as landmarks for being included in the robot's map in order to localize it along the time.

This section focuses on the inclusion of the hearing sense in SLAM, attempting to localize (without a priori information of the sound source location) both, the robot position and the sound source, along the time while the robot is moving freely in its environment.

In this context, a real application for Bearing-Only SLAM with Delayed Inverse Depth Feature Initialization is described. A small differential drive robot capable of sense bearing information respect to an external sound source with modest angular acuity ($\pm 10°$) is considered.

4.4.1 Sound-based Localization Approaches

Most robot sound localization approaches focused on improving the sensor capabilities (robustness, acuity, etc.) [116], [117], [118], [119], [120], [121]; in these cases the word "localization" refers to the estimation of the sound source respect to the robot in a robot coordinate frame in order to perform navigation; in that sense, a typical experiment is that a robot autonomously follows a sound source. In our case "localization" refers to estimating the position of both, robot and sound source along the time in a world coordinate frame.

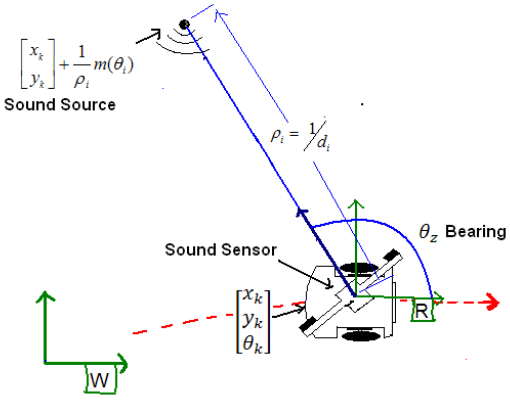


Figure 4.19 SSLAM: Mobile robot equipped with a Sound Sensor and parameterization for the Sound Source.

Others methods, [122], combine audio and vision for localizing speakers. In [123], there is an attempt to localize a mobile robot using several structured sound sources emitting distinctive codes in a similar approach to GPS system, where the position of each sound source is known a priori precisely. This is in contrast with a SLAM approach where landmarks locations are not know a priori.

4.4.2 Sound Sensor

In this work the term "Sound Sensor" refers to a robot sub-system capable of providing directional acoustic sensing (bearing information) independently on the number of transducers (2, 3 or a ring of microphones), the acoustic technique (interaural time difference (ITD), interaural phase difference (IID), etc) and the kind of sound source (natural or artificial).

A sound sensor has some similarities with others bearing sensor as monocular vision since in both cases depth information cannot be directly retrieved. A big distant object can appear in an image with the same size of a small close object and a loud distant sound source can be heard it similar to a weak close one. In that context some of the general principles applied to bearing-SLAM methods can be straightforward applied in sound-based SLAM (SSLAM).

4.4.3 Sound-based SLAM Algorithm

A differential drive mobile robot with 2-dof equipped with a sound sensor is considered. The cinematic model of the robot is the same that has been used so far (introduced in Section 3.2.1). To adapt our proposed Bearing-only SLAM algorithm (with Delayed Inverse Depth Feature Initialization) in order to implement a Sound-based SLAM system we consider a Sound Source as a feature point. Also it is assumed that the Sound Sensor is capable of tracking and measuring (bearing) the Sound Source. Figure 4.19 illustrates the mobile robot equipped with a sound sensor and the SSLAM parameterization (similar to the presented in Section 4.3)

4.4.4 Simulations

Figure 4.20 shows the simulation of the algorithm for a circular trajectory, a Gaussian noise with variance of 0.01 is added to the odometry (typical in encoders based systems). For the Sound Sensor, a variance $\sigma = 5°$ is considered. In the simulation it is supposed that the sound sensor is able of tracking the sound source over the 360°; in the case of having several sound sources it is supposed that the sensor can do a single measurement at a time and the separation of sound sources (data association problem) is obviated.

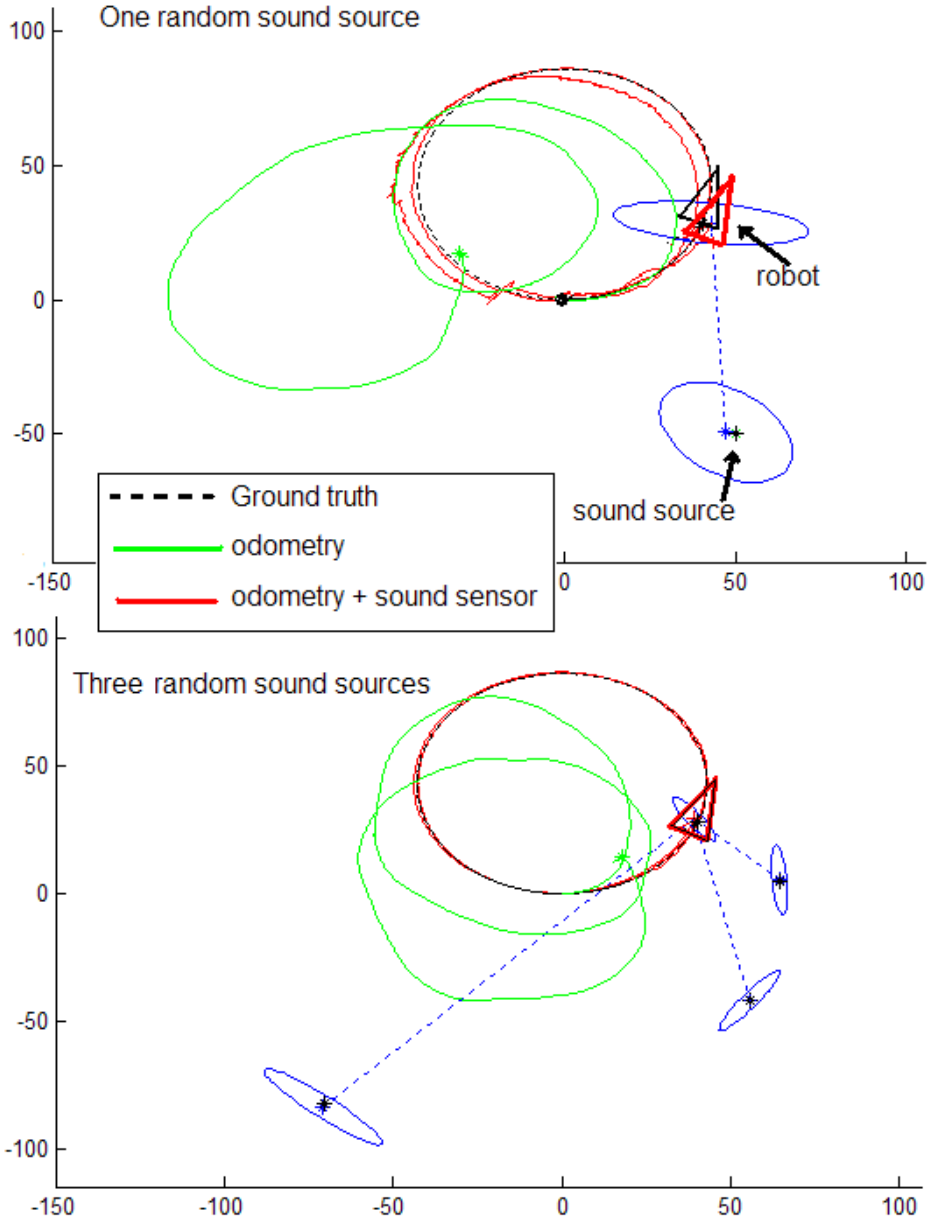


Figure 4.20 Simulations of the algorithm with initial random Sound Sources positions: Upper plot shows an example with a single sound source. Lower plot shows an example with three Sound Sources. Blue ellipses represent 2σ robot and Sound Source uncertainties. Note that with three sources the trajectory and its positions are more precisely recovered whereas the uncertainties are lower.

In the case of a single sound source the correct location initializations of the feature (Sound Source) in the system plays a very important role; if the feature is initialized far of its real position the filter could even diverge, if the single feature is initialized properly the trajectory can be recovered acceptably.

In the case of several Sound Sources the robustness of the algorithm increases, with lower dependence on initialization location; robot position and Sources positions uncertainties are lower due to an increase of available information in the system.

In both cases note how the estimated trajectory using only robot's encoders (odometry) diverges rapidly from the ground truth, in contrast with the trajectory estimated by the fusion (made in the SLAM algorithm) of both information sources; odometry and bearing (respect to the Sound Source). From these simulations, using our proposed Bearing-Only algorithm, we can conclude that if a mobile robot equipped with a Sound Sensor is capable of tracking the angular direction of a single or several Sound Sources then it is possible to estimate its position over the time reasonably well (correcting the odometry error).

4.4.5 Robot Implementation

Observing the simulation's results it is clear that the algorithm works better with more than one sound source, nevertheless in a real situation it is obvious that to track and to discriminate among several sound sources is a very challenging effort.

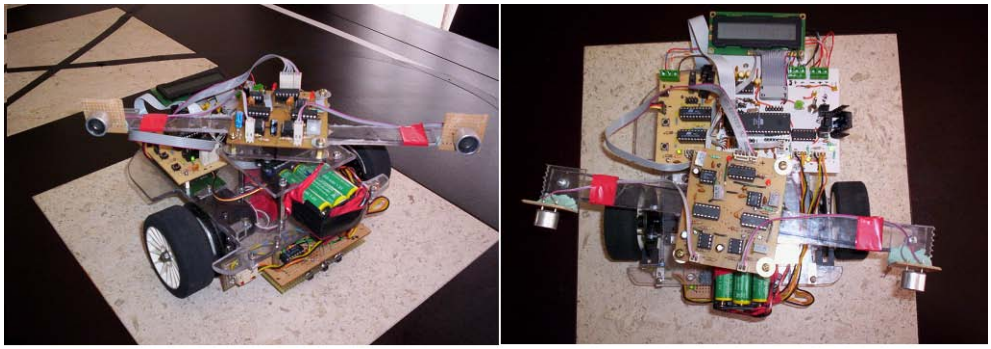


Figure 4.21 Pictures of the robot used for testing the algorithm; the robot was implemented specifically for the experiments of this chapter. Note the ultrasonic receivers at the ends of the rotating head.

On the other hand, observing simulation's results, it is surprising to note that a single sound source is enough for correct the odometry error in an acceptable manner, this fact has motivated us to testing the algorithm in a real robot equipped with a sound sensor of uncomplicated implementation.

Figure 4.21 shows the robot implemented for the experiments; it is equipped with optical encoders for odometry, sensors for tracking floor lines, a radio modem in order to establish communication with a PC and a 360° rotating head capable of tracking the sound direction of a single ultrasonic source.

For implementing the sound sensor (Figure 4.22), a sound-servo configuration is used. In order to avoid the sound source separation issue, a structured ultrasonic source is used. This sound source emits eight 40 kHz pulses every 40 milliseconds. Two ultrasonic transceivers are mounted at the ends of a 360° rotating head with a separation of 25cm between them. The head is driven by a modified servo for continuous rotation.

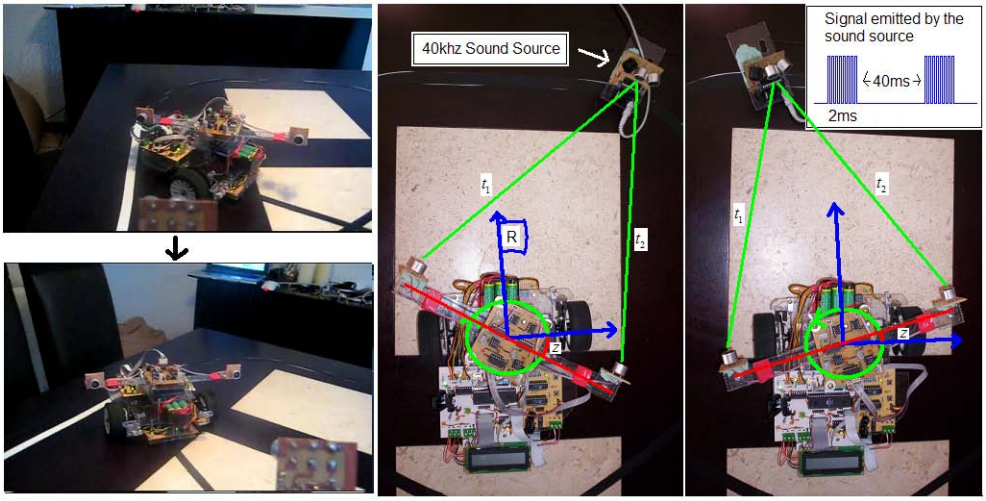


Figure 4.22 Interaural time difference (ITD) is used as feedback signal in a 360° servo configuration for implementing the robot's Sound Sensor. Every time (40ms) that a ultrasonic bean is received, a PI controller drives the servo looking for a ITD ($||t_1\text{-}t_2||$) close to zero (right pictures). If the robot remains static and the Sound Source is moving, then the rotating head will follow the direction of the Sound Source (left pictures).

A microcontroller measures the interaural time difference (ITD) in the ultrasonic receivers, the ITD is used as a feedback in a PI controller that drives the servo looking for an ITD close to zero.

The rotation head is connected by gears to a continuous rotation potentiometer for generating (with a voltage divisor) a voltage proportional to the bearing of sound source. The potentiometer has a wedge of 30° where is not possible to obtain measurements, therefore the Sound Sensor has a working range of 330°. Finally this voltage is converted to a binary value by an ADC (analog to digital converter). The Sound Sensor has acuity around $\pm 10°$.

4.4.6 Experimental Results with Real Data

For the experiments with real data, when the robot is moving, each time that the control program (running in a PC) requests data, the current encoders' counts (both wheels) and the ADC value (representing the bearing to the sound source) are sent by radio modem to the PC.

For every request these three values are stored in a table, when the robot stops of moving, the data table is used as the input for a SSLAM MatLab implementation, here it is important to say that is straightforward to implement the algorithm in a real time version.

Several experiments were realized in order to test the performance of the SSLAM algorithm. Figure 4.23 shows two of the experimental setup used: In one type of experiments, the robot tracked floor lines in order to make a good comparison between the estimated trajectories and the ground truth. Note the test paths and the location of the Sound Source (left picture). In other type of experiments the robot was manually driven by a joystick (right picture). The idea is move the robot from a starting zone to another one (observe the blue circles) and then, return the robot near to its initial position. After that a comparison is made between the estimated and the real location of the robot.

Figure 4.24 illustrates a robot's lap in an eight-shape path. The frames were extracted from a video taken of the experiment. Note how the rotating head follows de direction of the Sound Source, independently of the robot's direction.

Figure 4.25 shows the results obtained with the data captured from the robot's sensors, when the robot followed two predefined paths using its line-tracking sensor: an oval path (a) and an eight-shape path (b). In plot (a) note how after 6 laps (around 24m of travel) the trajectory is reasonably well recovered with the SSLAM algorithm (red trajectory) compared with the estimates using only the odometry (green).

In (b) a more challenging trajectory is followed by the robot, note how the large drift showed by the robot's odometry is well bounded in the trajectory estimated by the SSLAM algorithm. The non-operative wedge of 30º in the robot's head has little effect in the estimated trajectories; maybe in plot (a) can be slightly appreciated, but the trajectory is rapidly recovered by the algorithm when the head enters in the fully operative area of 330º.

Finally in Figure 4.26 the robot was manually driven by a joystick from a starting location to a second location and then driven back to the starting location several times. Commonly turns are the mainly source of odometry errors. In this experiment, it can be clearly appreciated how the trajectory estimated by the odometry differs from the real one due to the several (and closed) turns. Nevertheless the robot location is consistently estimated with the SSLAM method over the whole trajectory.

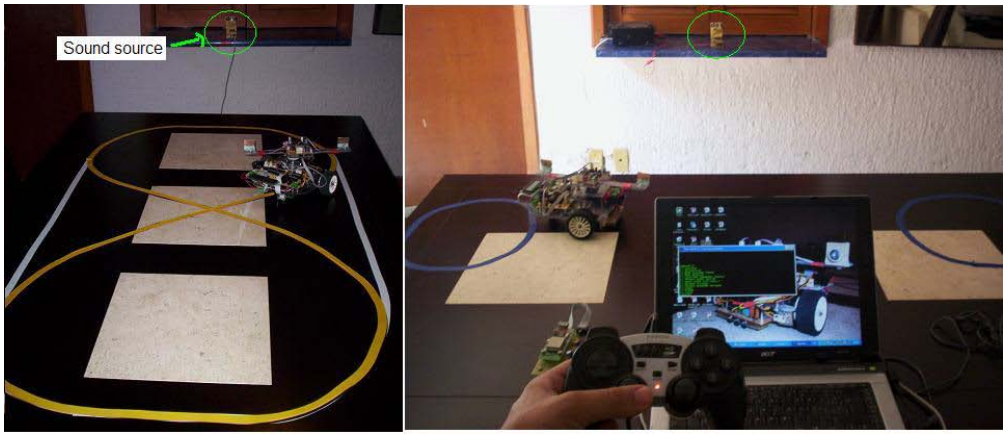


Figure 4.23 Experimental Setup: Robot tracking a predefined path using its line-tracking sensor (left). Robot manually driven by a joystick (right). Note the Sound Source at the middle-upper of the pictures.

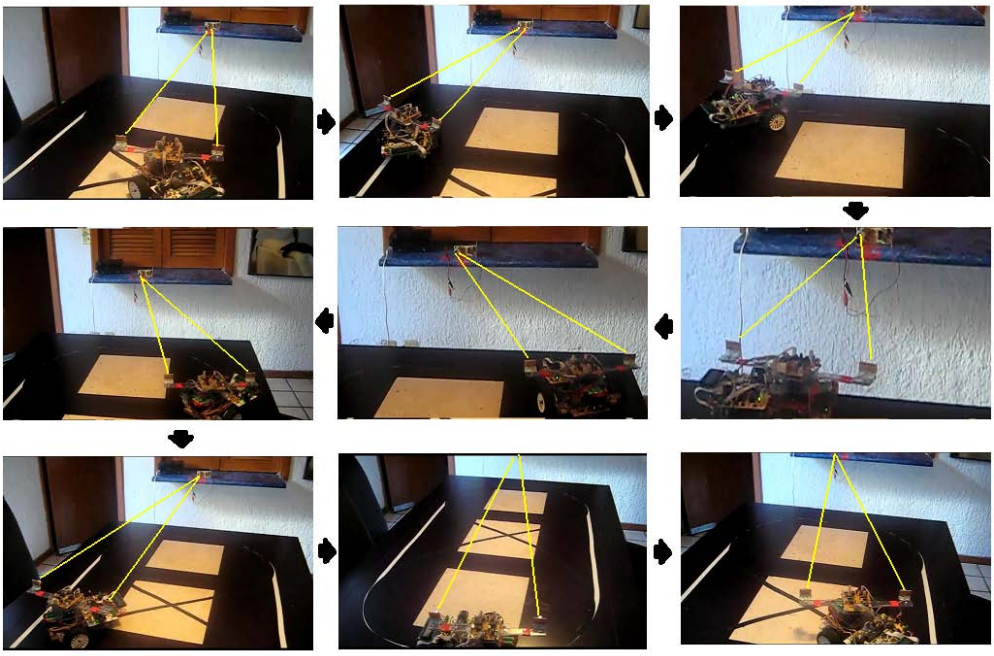


Figure 4.24 Sequence of frames (taken from a video) illustrating a robot's lap in an eight-shape predefined path. The yellow rays are only for point up the movement of the robot's rotating head following the direction of the Sound Source.

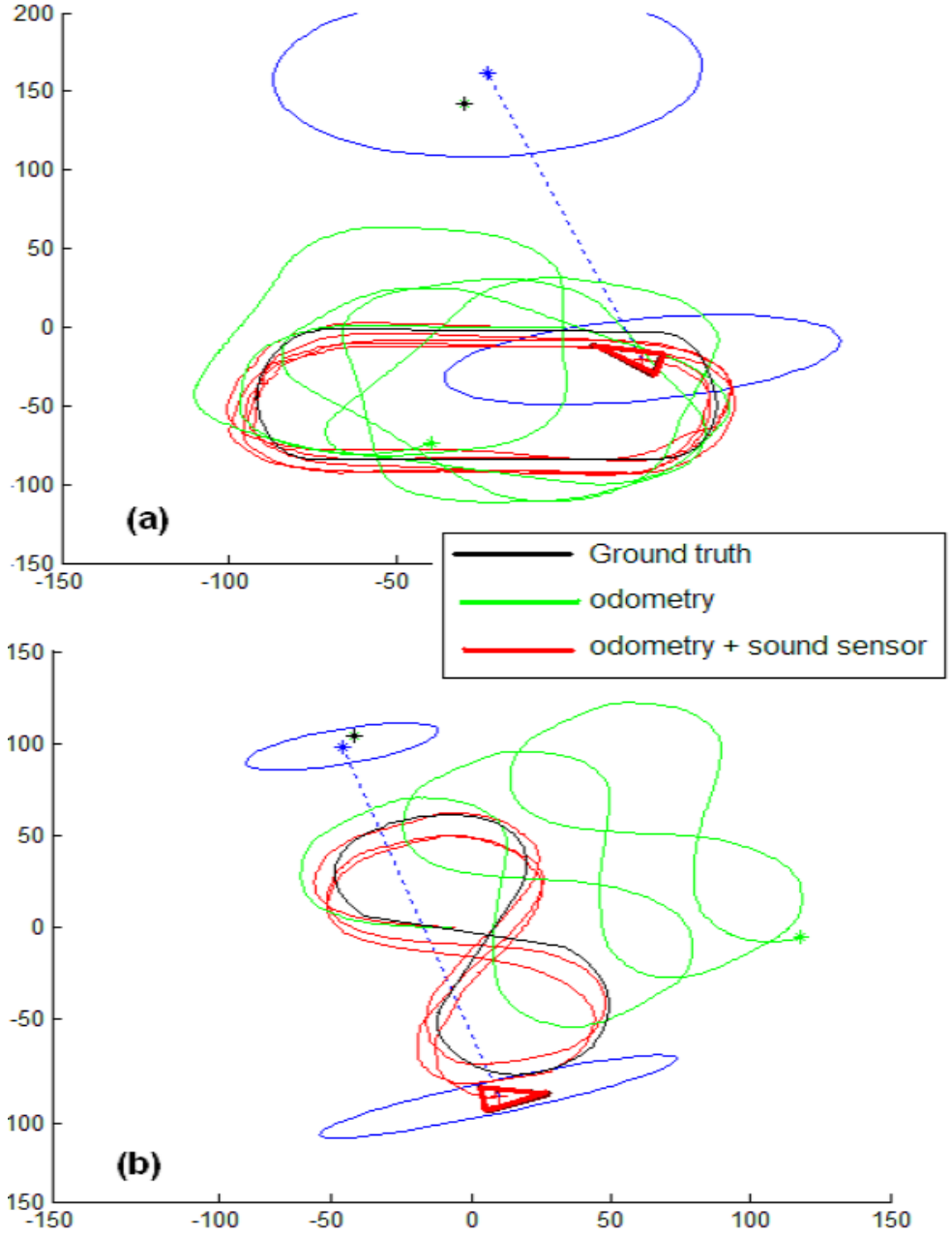


Figure 4.25 Plots (a,b) show the estimates for two different predefined trajectories followed by the robot: an oval (a) and an eight-shape (b), in both cases note how the estimated trajectories with the sound based SLAM (red) are reasonably recovered after several laps, on the other hand note the big error propagation in the estimated

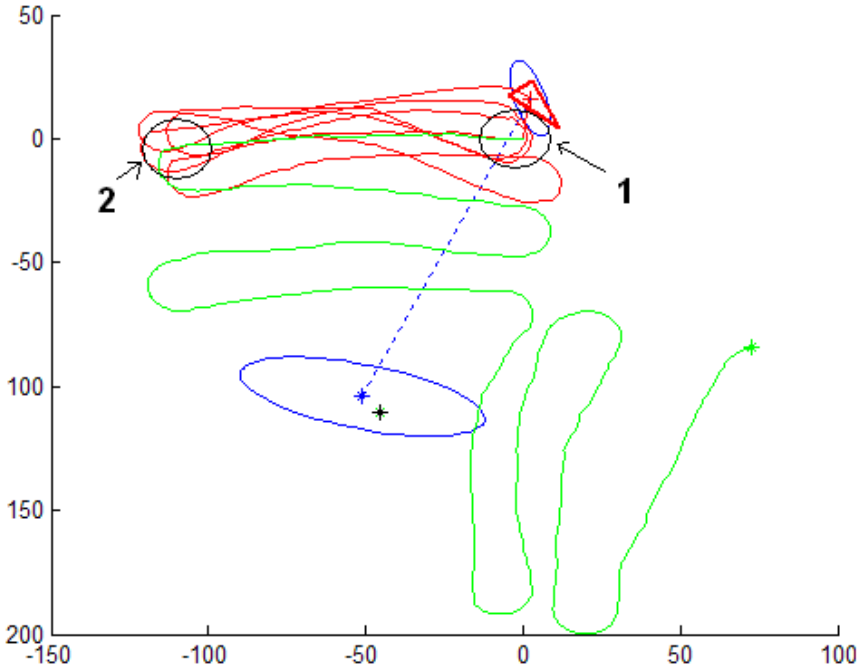


Figure 4.26 In this experiment, the robot was manually driven by a joystick from the location 1 to the location 2 (illustrated by black circles) and then back from the location 2 to the location 1 several times. Note the huge error in the odometry estimates, compared with the consistently SSLAM estimated trajectory.

4.5 Conclusions

In this chapter we have analyzed the Bearing-Only SLAM problematic. It has been seen that the main challenge is related with the process of initializing new features in the map. In Bearing-Only SLAM extra treatment of the features has to be done previous to added it to the stochastic map since bearing-sensors doesn't provides spatial information related to features depth.

Several techniques have been proposed in literature for initializing features in Bearing-Only SLAM. Inverse Depth feature Parameterization increases the linearity of the measurement equation. In that sense, Un-delayed Inverse Depth parameterization method has show to be an excellent option for Bearing-Only SLAM, because a standard EKF-SLAM framework can be straightforward applied to address the Bearing-Only SLAM problem. Nevertheless this kind of feature initialization methods has some drawbacks: Mainly due to the initial features depth and variance are fixed, it often happens, in cases

where there are near and distant landmarks, that estimated depth cannot converge to the real ones causing that map and vehicle location are poorly estimated. It also often happens that after some Kalman update steps, the features depths become negatives causing even the divergence of the filter. On the other hand, experimentally we found that making a manual tuning of the initial parameters then the percentage of success increasing significantly. The last fact, motive us to propose a novel feature initialization technique called Delayed Inverse Depth Feature Initialization, where initial parameters are dynamically estimated priors to add a landmark as a new feature in the stochastic map.

In this chapter we introduce our Delayed Inverse Depth Feature Initialization method for Bearing-Only SLAM for a 3DOF (2D) context and assuming the availability of odometry. Several simulations were presented in order to illustrate its performance. Simulations results show that our proposed Delayed method is more robust than the Un-delayed method for different scenarios.

In order to test our Delayed Inverse Depth Feature Initialization method with real data, a novel approach, called SSLAM, has been also presented. This algorithm performs sound-based simultaneous localization and mapping. A robot subsystem (called, Sound Sensor), capable of providing bearing information respect to one or several sound sources, is closely related (from the algorithmic point of view) with others bearing sensors (e.g. camera), in that sense a general Bearing-Only SLAM framework (like the proposed in this chapter) can be straightforward modified for working in a sound-based context.

The algorithm simulations show that several sound sources improve the robustness and effectiveness of the method; nevertheless a real sound sensor capable of tracking and separating several sound sources requires a challenging implementation. In that sense, a small mobile robot, capable of tracking a single sound source, is implemented in order to test SSLAM with real data sensor.

The experimental results show how the method is able to estimate the sound source position without prior knowledge of the environment; this information is subsequently used for estimating reasonably well the robot's trajectory. It can be appreciated how the error propagation of the encoders-based odometry is bounded.

These experimental results are very promising since the method is tested in a robot with very limited capabilities. The method could be applied in a robot equipped with a more complex sound sensor capable of measuring several natural audio sources.

Chapter 5

The Data Association Problem

So far we have avoided in all the simulations, or at least simplified, in the case of the SSLAM, the data association problem. In that sense, we have been assuming that we perfectly know which landmark correspond to each measurement. Nevertheless, in practice, the data association is a very challenging problem. On the other hand we have seen that Bearing-Only SLAM is a viable alternative for address the SLAM problem if the initialization of new map's features is accounted for. In this context Cameras are by far the most popular bearing sensor, among other things because provides a huge amount of information, useful for addressing the data association problem.

In this chapter several techniques are described for addressing the data association problem in a SLAM context. All the approaches presented in this chapter are vision-based techniques due to cameras are the more powerful bearing sensor for performing SLAM. One aim of this chapter is to be a "bridge" between the chapter 4, where the Bearing-Only SLAM general problematic is introduced, and the chapter 5, where the Monocular SLAM is presented as a sub-class of Bearing-Only SLAM based on monocular cameras.

First in section 5.1, a small survive on the data association problem based on local features appearance is given. Some of the related work and methods, representing the state of the art, are also briefly reviewed.

In section 5.2 a novel image feature descriptor called ICAD is presented. The ICAD descriptors are based on ICA (independent component analysis). Experimental results are presented in order to show the performance of the method.

In a context of camera-based SLAM, a video-stream is available. In Section 5.3 a new framework for capturing the variability of image features descriptors based on statistical methods is presented, in order to make the features descriptors more robust to changes in illumination or point of view. Experimental results comparing methods and descriptors are presented.

Some of the techniques available in the literature have shown to be an excellent option for addressing the data association problem. Nevertheless their direct use in SLAM applications is not straightforward. In Section 5.4 a simple but effective framework is presented for using state of the art image features descriptors in camera-based SLAM systems.

Finally in section 5.5 the conclusions of the chapter are given.

5.1 Introduction

Possibly the one of the hardest problem in robotic is the *correspondence problem*, also known as the *data association problem* [124]. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world. For example two instances of the correspondence problem in robot mapping are closing cycles in large cyclic environment and the kidnapped problem. When a mobile platform moves through its environment a single video camera can be used in order to build a map of its surroundings and to determine its position (absolute or relative). Because in computer vision a great amount of information is available, this information can be used for solve or at least to reduce the challenging correspondence problem.

For visual recognition (camera-based data association), two major classes of techniques can be identified:

Model-based. These kinds of techniques employ geometric models of the target objects for feature extraction or alignment [125].

Appearance-based. These kinds of techniques are related to the appearance of objects in an image. That is, they are related directly to the pixels of an image. They are also called model-free methods since they extract features or extract information from images without explicit model shape properties of the target objects.

In computer vision, sparse image statistics called *features* are used in order to create a model that is rich enough to represent the environment and sparse yet to be stored efficiently. This chapter presents some appearance-based feature extraction techniques that are relevant to SLAM. Two key requirements must be met for a vision-based feature extraction technique to have SLAM relevance. First, the method must operate in real time. Mobile robots move through their environment, and so the processing simply cannot be an off-line operation. Second, the method must be robust to the real-world conditions outside a laboratory. This means that carefully controlled illumination assumptions and carefully painted objects are unacceptable requirements.

Through the following descriptions, we keep in mind that vision-based interpretation is primarily about the challenge of *reducing information*. A sonar unit produces perhaps fifty bits of information per second. By contrast, a CCD camera can

output 240 million bits per second! The sonar produces a tiny amount of information from which we hope to draw broader conclusions. But the CCD chip produces too much information haphazardly. For example, we may intend to measures the color of a landmark. The CCD camera does not simply report its color, but also measures the general illumination of the environment, the direction of illumination, the defocusing caused by optics, the side effects imposed by nearby objects with different colors, and so on. Therefore the problem of visual feature detection is largely one of removing the majority of irrelevant information in an image so that the remaining information unambiguously describes specific features in the environment.

We divide vision-based feature detection method into two classes based on their spatial extent:

Spatially localized features. Are those features found in sub-regions of one or more images, corresponding to specific locations in the physical world.

Whole-image features. Are those features that are functions of the entire image or set of images, corresponding to a large visually connected area in the physical world.

In this chapter we focuses on spatially localized features because are congruent with a general SLAM scenario: In the computer vision community many algorithms assume that the object of interest occupies only a sub-region of the image, and therefore the features being sought are localized spatially within images of the scene. Local image-processing techniques find features that are local to a subset of pixels, and such local features map to specific locations in the physical world. This makes them particularly applicable to geometric models of the robot's environments.

The data association is fundamental part of the SLAM process, since wrong associations will produce incorrect maps.

5.1.1 Visual Features Representation

Features can be the result of a general neighborhood operation (feature extractor or feature detector) applied to the image, specific structures in the image itself, ranging from simple structures such as points or edges to more complex structures such as objects. Other examples of features are related to motion in image sequences, to shapes defined in terms of curves or boundaries between different image regions, or to properties of such a region. The feature concept is very general and the choice of features in a particular computer vision system may be highly dependent on the specific problem at hand. The most popular local feature extractor used by the mobile robotics community is the edge detector.

When features are defined in terms of local neighborhood operations applied to an image, a procedure commonly referred to as *feature extraction*, one can distinguish between feature detection approaches that produce local decisions whether there is a feature of a given type at a given image point or not, and those who produce non-binary

data as result. The distinction becomes relevant when the resulting detected features are relatively sparse. Although local decisions are made, the output from a feature detection step does not need to be a binary image. The result is often represented in terms sets of (connected or unconnected) coordinates of the image points where features have been detected, sometimes with sub-pixel accuracy.

When feature extraction is done without local decision making, the result is often referred to as a *feature image*. Consequently, a feature image can be seen as an image in the sense that it is a function of the same spatial (or temporal) variables as the original image, but where the pixel values hold information about image features instead of intensity or color. This means that a feature image can be processed in a similar way as an ordinary image generated by an image sensor. Feature images are also often computed as integrated step in algorithms for feature detection.

A specific image feature, defined in terms of a specific structure in the image data, can often be represented in different ways. For example, an edge can be represented as a Boolean variable in each image point that describes whether an edge is present at that point. Alternatively, we can instead use a representation which provides a certainty measure instead of a Boolean statement of the edge's existence and combine this with information about the orientation of the edge. Similarly, the color of a specific region can either be represented in terms of the average color (three scalars) or a color histogram (three functions).

When a computer vision system or computer vision algorithm is designed, the choice of feature representation can be a critical issue. In some cases, a higher level of detail in the description of a feature may be necessary for solving the problem, but this comes at the cost of having to deal with more data and mote demanding processing. An instance of a feature representation can be referred to as a (feature) descriptor.

5.1.2 Features Descriptors

A *descriptor* can be viewed like a distinctive representation of the feature and its variations among the time in a compact way respect the original data without lose of its statistical meaning. In this scenario, the correspondence problem is represented for matching image features descriptors in a wide base line. We understand "wide base line" as a big difference in time and camera point of view, between learning and recognition phases. To address the whole problem descriptors can be very helpful because they can provide distinctive signatures of different locations in space. Furthermore, descriptors have to be as much invariant as possible to changes in scale, rotation, illumination or projection (point of view) and algorithms must be efficient and robust to a number of environmental variations such as lighting, shades, and occlusions among others.

Reliable image features are a crucial component of any visual recognition system. Despite much progress, research is still needed in this area. Elementary features and

descriptors suffice for a few applications, but their lack of robustness and invariance puts a heavy burden on the learning method and the training data, ultimately limiting the performance that can be achieved. More sophisticated descriptors allow better inter-class separation and hence simpler learning methods, potentially enabling generalization from just a few examples and avoiding the need for large, carefully engineered training databases. The feature and descriptor families that we advocate typically share several basic properties:

5.1.3 Locality and redundancy:

For resistance to variable intra-class geometry, occlusions, changes of viewpoint and background, and individual feature extraction failures, descriptors should have relatively small spatial support and there should be many of them in each image. Schemes based on collections of image patches or fragments are more robust and better adapted to object-level queries than global whole-image descriptors. A typical scheme thus selects an appropriate set of image fragments, calculates robust appearance descriptors over each of these, and uses the resulting collection of descriptors as a characterization of the image or object (a ``bag-of-features'' approach – see below).

5.1.4 Photometric and geometric invariance:

Features and descriptors must be sufficiently invariant to changes of illumination and image quantization and to variations of local image geometry induced by changes of viewpoint, viewing distance, image sampling and by local intra-class variability. In practice, for local features geometric invariance is usually approximated by invariance to Euclidean, similarity or affine transforms of the local image.

5.1.5 Repeatability and salience:

Fragments are not very useful unless they can be extracted reliably and found again in other images. Rather than using dense sets of fragments, we often focus on local descriptors based at particularly salient points – ``keypoints'' or ``points of interest''. This gives a sparser and thus potentially more efficient representation, and one that can be constructed automatically in a preprocessing step. To be useful, such points must be accurately relocalizable in other images, with respect to both position and scale.

5.1.6 Informativeness:

Notwithstanding the above forms of robustness, descriptors must also be informative in the sense that they are rich sources of information about image content that can easily be exploited in scene characterization and object recognition tasks. Images

contain a lot of variety so high dimensional descriptions are required. The useful information should also be manifest, not hidden in fine details or obscure high-order correlations. In particular, image formation is essentially a spatial process, so relative position information needs to be made explicit, e.g. using local feature or context style descriptors rather than global moments or Fourier descriptors.

Features and descriptors with some or all of these properties have become popular choices for visual correspondence and recognition, particularly when large changes of viewpoint may occur (wide base-line).

5.1.7 Data Association Process based on Features Descriptors

Two steps can be distinguished on the utilization of visual features: The detection of interest points (features) and the description of the selected points. The first step involves the selection of suitable points in the images that can be used as landmarks. The points should be detected at different distances and viewing angles, since they will be observed by the robot from different poses. In a second step the features are described by a feature vector which is composed using local image information. The descriptor is used to solve the data association problem: when the robot observes a landmark in the environment, it must decide whether the observation corresponds to a previously seen landmark or a new one.

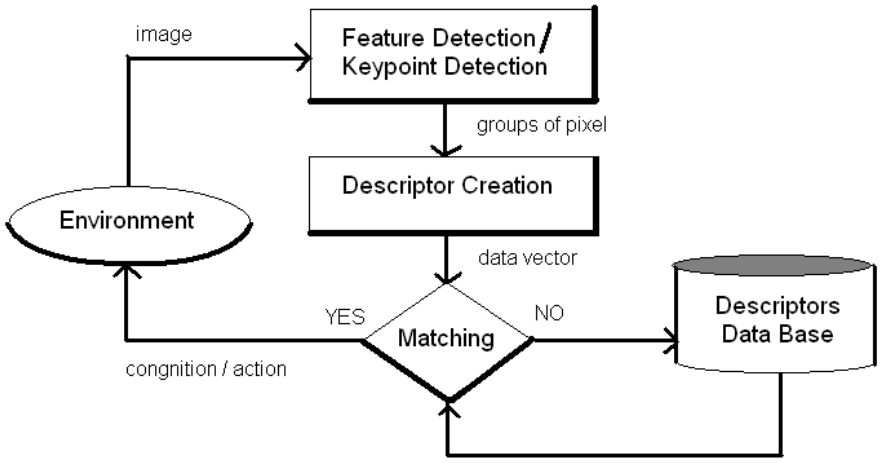


Figure 5.1Typical scheme for data association process based on features descriptors.

A typical descriptor consists on a vector of *n* elements. Common values for *n* are 64, 128 or 256. Figure 5.1 show a typical scheme for data association process based on

features descriptors, from the computer vision point of view. When a new image is obtained from the environment by a vision-based sensor, features (also called Keypoints) are extracted. Is important to note that all vision-based sensors supply image with such a significant amount of noise that a first preprocessing step that usually consist of "cleaning" the image before launching any feature extraction algorithm. A common output for the feature extraction algorithm is a group of pixel centered in a point of interest of the image (keypoint). After that, the Descriptors are obtained from the detected features. For know if the current detected feature (related to a spatial location) correspond to a spatial location of the environment previously detected then their Descriptor is matched against a data-base of previously stored feature descriptors. The data-base of descriptors can be viewed as a space of n dimensions where each Descriptor represents a point in such space.

Then for comparing the Descriptor against the data-base, it is looking for the close "point" (or points) in the descriptor-space. Nearest Neighbor techniques are usually employed for determined the most closed point (or points) in a space. If the matching process succeeds then it is assumed that the vision-based sensor is "viewing" some feature previously mapped and some action is carried out. If the matching process fails the nit is assumed that the Descriptor corresponds to a new feature of the environment and it is stored in the data-base as a new Descriptor.

5.1.8 Related Work

Interest points detected in the images are not very stable, and the matching between different views becomes difficult. In consequence, the problem of selecting a suitable interest point detector and descriptor for visual-based SLAM is still open. Some examples of interest point detectors are: Harris [126], Harris-Laplace [127], SUSAN [128], and Shi-Tomasi [129]. In [130] some interest point detectors are evaluated for their use in visual-based SLAM. In this work the Harris corner detector [126] was found to be the most suitable point detector for vision-based SLAM.

Some common descriptors are:

Gray level patch: This method describes each feature using the gray level values at a sub-region around the interest point. This method has been used in [109] as a descriptor of Harris points in visual-based SLAM framework.

Orientation Histograms: The orientation histograms are computed from the gradient image, which represents the gray value variations in the x and y direction. In [131] orientation histograms are applied for navigation tasks.

SIFT: The Scale-Invariant Feature Transform (SIFT) detects distinctive key points in images and computes a descriptor for them. The algorithm, developed by Lowe, was initially used for object recognition task [132]. SIFT features are located at maxima and minima of a difference of Gaussian functions applied in scale space. They are computed by

building an image pyramid with re-sampling between each level. Next, the descriptors are computed based on orientation histograms at a 4-by-4 sub-region around the interest point, resulting in a 128 dimensional vector. SIFT descriptors provides invariance to image translation, scaling, rotation and partial invariance to illumination changes and view point changes. SIFT features have been used in robotics applications as visual landmarks for localization and SLAM task: [133], [134], [135].

SURF: Speeded Up Robust Features (SURF) is a scale and rotation invariant descriptor presented by Bay *et al.* [136]. The detection process is based on the Hessian matrix. SURF descriptors are based on sums of 2D Haar wavelet responses, calculated in a 4-by-4 sub-region around each interest point. The standard SURF descriptor has a dimension of 64 and the extended version (e-SURF) of 128. The u-SURF version is not invariant to rotation and has a dimension of 64.

Other approaches like [137], [138] or [139] have been presented for address the problem of image features representation for tracking, recognition or reconstruction. In general, those methods searching of extrema in image scale space for obtain good candidates locations for detection. In [140] an approximate version of Kernel Principal Component Analysis (KPCA) was used to estimate features descriptors.

5.2 ICA Descriptors

If features Descriptors want to be used in robotics applications like SLAM, where real time operation is fundamental, then the computational cost of the data association process based on feature descriptors becomes critical. In general SIFT descriptors have shown to be the more robust option for reliable data association problem using local feature descriptors [141]. Nevertheless the computational cost of the SIFT descriptors are quite high and therefore their direct application to several scenarios (including, SLAM) is not straightforward. More recently SURF descriptors also have shown to be an excellent option for addressing the data association problem because demonstrates to have a similar performance than SIFT but a lower computational cost [142]. Even so, for applications like SLAM, where are required the computation of several tasks apart from the data association, SURF descriptors maintain a considerable computational cost. On the other hand Gray level patches are an option very fast to computing but only useful for matching features frame to frame (small base-line tracking), where there are small changes on illumination or point of view.

In this section we present an intermediate option for addressing the data association problem, called *ICA Descriptors* (ICAD) which are based on ICA (independent component analysis) technique [143]. In general ICA descriptors presents an intermediate performance in, computational cost and robustness, respect to SIFT and SURF descriptors on the one hand and the Gray level patches on the other: ICA

descriptors are more robust than Gray level patches but less than SIFT or SURF. ICA descriptors also have a lower computational cost than SIFT or SURF but a little more expensive than the Gray level patches.

5.2.1 PCA and ICA.

Principal Component Analysis (PCA) [144] is a standard statistical tool used to find the orthogonal directions corresponding to the highest variance. It is equivalent to a decorrelation of the data using second-order information. The basic idea in PCA is to find the components $y_1, y_2, \ldots, y_n$, that explain the maximum amount of variance, by n linearly transformed components. Then the principal components are given by:

$$y_i = w_i^T X \tag{5.1}$$

Where

$$X = [x_1, x_2, \ldots, x_m]^T \tag{5.2}$$

And x_i is an observed data vector, and w_i is a basis vector (an eigenvector of the sample covariance matrix $E\{XX^T\}$) . It can be written, in matrix form, as:

$$Y = WX \tag{5.3}$$

Where

$$Y = [y_1, y_2, \ldots, y_n]^T \tag{5.4}$$

And y_i is a principal component vector. (See [144] for more details about PCA).

The ICA attempts to go one step further than PCA, by finding the orthogonal matrix V which transforms the data Y into Z having $Z_1, Z_2, \ldots, Z_m$ statistically independent. The ICA is thus more general than PCA in trying not only to decorrelate the data, but also to find decomposition, transforming the input into independent components.

The simplest ICA model, the noise-free linear ICA model, seems to be sufficient for most applications. This model is as follows: ICA of observed random data X consist of estimating the generative model:

$$X = AS \tag{5.5}$$

Where

$$X = [x_1, x_2, \ldots, x_n]^T \tag{5.6}$$

$$S = [s_1, s_2, \ldots, s_n]^T \tag{5.7}$$

And x_i is an observed random vector and s_i is a latent component, and A is the constant *mixing* matrix. The transform we seek is $B=VW$, then

$$Z = BX = BAS = CS \tag{5.8}$$

If an orthogonal matrix V that transforms the mixed signals X into Z with independent components could be found, and assuming that at least one independent source s_k is normally distributed, then $Z=CS$ with C being a non-mixing matrix. The ICA algorithm attempts to find the matrix B which ensures that Z is independent.

Many ICA algorithms are available. A computationally efficient ICA algorithm, called the FastICA algorithm, is chosen to be used in our work. See [143] and [145] respectively, for more details about ICA and FastICA.

5.2.2 ICA Applied to window-based Image Features

The assumption of implicit Gaussian sources in PCA makes it inadequate when true sources are non-Gaussian. In particular, it has been empirically observed that many natural signals, like natural images are better described as linear combinations of sources with long tailed distributions. Ica provides a better probabilistic model of data in an n-dimensional space. It is sensitive to high-order statistics in the data not only to the covariance matrix. ICA can yield either an orthogonal or a non orthogonal basis, changing the relative distance between data point, this change in metrics may be useful for classification algorithms, like nearest neighbor, for instance that make decisions based on relative distances between points.

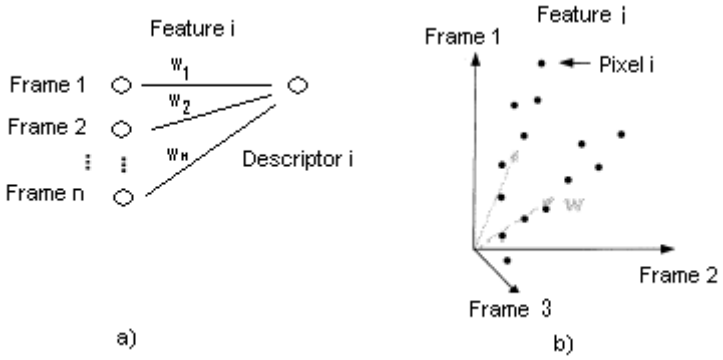


Figure 5.2 a) To apply ICA to an image a matrix is formed where each row is a feature i tracked in frame n. b) ICA finds a weight vector w in the directions of statistical dependencies among the pixel locations.

Our goal in this work is to find descriptors to represent image features. If we consider a feature as a window of p-by-p pixels in a frame, we can organize each feature as a long vector with as many dimensions as number of pixels in the feature. ICA can be

applied to this data organizing each vector into a matrix X where each row is the same image feature for different frames (Figure 5.2 left). In this approach, image features are random variables and pixels are trials (Figure 5.2 right), it makes sense to talk about independence of features or functions of features.

Two features i and j are independents if when moving across pixels, it is not possible to predict the value taken by the pixel on feature j based on the values taken by the same pixel on feature i.

5.2.3 Method Description

In the scenario of mobile robot mapping and localization, we pretend to recognize at high speed different features along a sequence of images, based on a two-phases algorithm: *database generation* (learning, off-line process) and *recognition* (matching, on-line process).

In the learning phase, we employ a standard small base-line tracker for detecting and tracking candidate features, in this work the Lucas Kanade tracker (KLT) [146], [147] is used. But any small base-line tracker can be used.

When the KLT locates a feature (feature i at frame f), a p-by-p pixels window around the feature center is stored as a vector u_{fi} of length p-by-p, with a distinctive label; in the following frames the feature tracked and repeating the above process, storing with the same label. We can choose, as a parameter, first, the number of frames that the features has to persist in the KLT in order to estimate the feature descriptor and second, the number of features to treat for each frame. After that, vectors with the same label (same feature i) are regrouped in a matrix U_{i} .

$$U_i = [u_{1i}, \dots, u_{ni}]^T \tag{5.9}$$

Where n is the number of frames where the feature has been tracked.

Then for each matrix U the FastICA is applied as it were shown in the previous section along with dimensional reduction selecting the largest eigenvalue to be retained. At the output of the FastICA we obtain a descriptor q_{i} with dimension equal to the feature windows size. The descriptors are stored in a database with unique label for each feature.

In the recognition phase features are detected but not tracked by the KLT for each incoming frame. Then for each feature detected a windows is obtained in the same way than the learning stage and sorted in a vector v_{i} the FastICA is directly applied to this vector without dimensional reduction producing a descriptor r_{i}. A fast approximate k-nearest neighbor algorithm [148] is applied to the database in order to look for the 2-nearest neighbor descriptors q_{i1} and q_{i2}.

Be

$$k_1 = d(r_i, q_{i1}) \quad k_2 = d(r_i, q_{i2}) \tag{5.10}$$

(d is the Euclidean distance), $k_1 \leq k_2$, we define α

$$\alpha = \frac{k_1}{k_2} \tag{5.11}$$

To be a factor used by our algorithm, as a threshold, for considering a good match between the candidate descriptor r_i and its corresponding nearest descriptor q_{i1} and q_{i2} in the database. When α tends to 0 means a great distance between candidates and, empirically, the results are better.

5.2.4 Experimental Results.

We have implemented a C++ version of our method that runs on a PC 2GHz Pentium IV processor, 512MB RAM. A non-expensive USB Webcam with a maximum resolution of 640-by-480 pixels and 30 fps has been used.

We performed a variety of experiments in order to examine the validity of our proposal; In the learning phase, a video sequence of a rigid environment desktop scene was recorded moving the camera slowly and continuously in order to obtain a change of some degrees in the 3D point of view and rotation of the camera. Later, twenty ICA descriptors were created from this video sequence using windows of 12-by-12 pixels and stored in a descriptor database as it has been described in the previous section. Figure 5.3 shows the relationship between the sizes of the window used to create the ICA descriptors and the false positive in our experiments.

Together with these 20 descriptors, the database contains another 1000 descriptors corresponding to other video sequences. The objective of these experiments consists of observing the response of our system when a set of descriptors coming from an online video sequence (close to the learning sequence, as it is explained below) will be matched with the descriptors database.

The first experiment has been to observe the response to the change of the p-by-p windows size used to create the descriptors in images of 320-by-240 pixels. Figure 5.3 shows percentage of false positive versus different sizes of p.

A second experiment has been to observe the response of the system in front the changes in parameter α in the following four cases (Figure 5.5):

a) Little change in 3D point of view with respect to the position in the learning phase and little change in the illumination (Figure 5.4, upper-left).
b) Little change in 3D point of view and medium change in illumination (Figure 5.4 upper-right).
c) Approximately a change of 20 degrees in the 3D point of view and medium change in illumination (Figure 5.4 lower-left).
d) Approximately a change of 10 degrees in the 3D point of view, 5 degrees in rotation and medium change in illumination (Figure 5.4 lower-right).

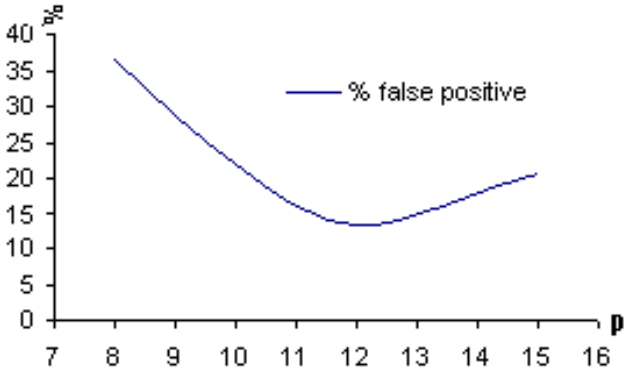


Figure 5.3 Relationship between the sizes of the window used to create the ICA descriptors and the false positive, Note the local minimum at p=12.

Figure 5.4 Descriptors created in the learning phase (central image). Examples of descriptors matched in the recognition phase: (upper-left image) *case a*: little change in 3D point of view and little change in illumination; (upper-right image) *case b*: Little change in 3D point of view and medium change in illumination; (lower-left image) *case c*: a change of 20 degrees in the 3D point of view and medium change in illumination; (lower-right image) *case d*: a change of 10 degrees in 3D point of view, 5 degrees in rotation and medium change in illumination.

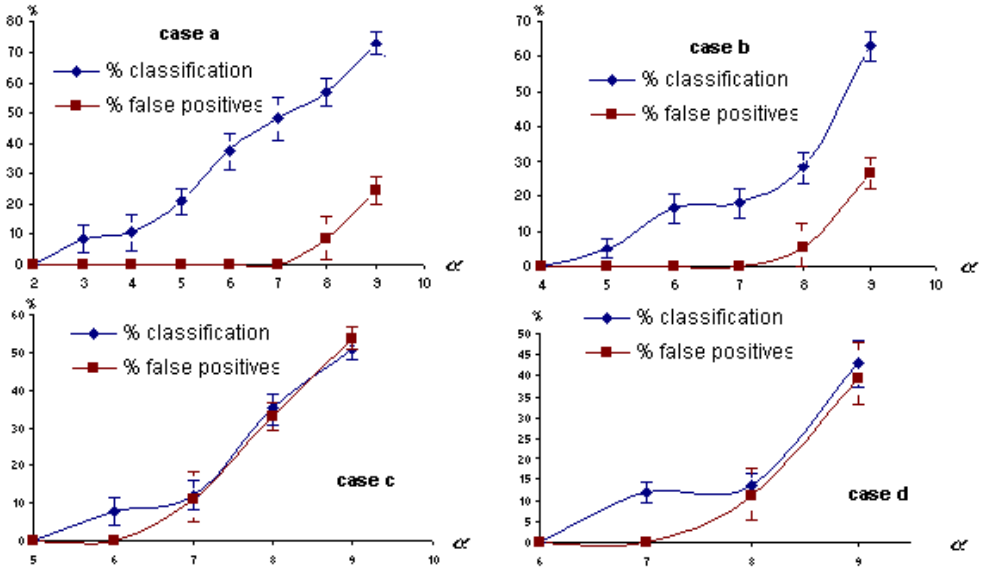


Figure 5.5 Response of the percentage of classification and percentage of false positives to the change of the parameter α.

The response of the algorithm is expressed in terms of two measurements:

Percentage of Classification as the ratio between the number of descriptors matched (correctly or incorrectly) and those that potentially can be found, we talk about "potentially can be found" because one of the objectives of our methods is running fast; to achieve this goal, we consider only a finite number of possible locations in each frame to be matches. These possible locations are established feature detecting algorithm of the same tracker used in the learning phase. Then it would be possible that some features to match actually exist in the fame but algorithm does not detect them making impossible the matching process. The number of regions to consider in each frame can selected, in our experiment we chose forty.

Percentage of false positive as a quality measurement of detection, and it is defined as the ratio between false positive and the number of (correctly or incorrectly) descriptors matched.

In Figure 5.5 is relevant to show the trade-off between percentage of classification and percentage of false positive when varying α. The lower the threshold, the lower is the number of false positive, but consequently the percentage of classification is also low. The computational speed in the matching process, among the forty possible descriptors in each frame versus one thousand descriptors in the database is about 15 Hz.

5.3 Learning Variability of Image Feature Appearance

Invariance to some changes like point of view or illumination are difficult to represent in descriptors created from a single frame since there does not exist a single-view statistic that is invariant with respect to viewpoint or lighting conditions. In applications like vision-based SLAM or robot localization a video stream is available making possible to detect and then track the features across the images with small base line (frame to frame); several trackers [146], [147] can be used for this. The correspondence problem is easy in a small base line. While a robot moves through their environment and detect image features with its camera, the appearance of the features changes due to natural changes in illumination and changes in the point of view. We can take advantage of matching in a small base line for capture the variations in the appearance of each feature along the time to make the feature descriptors more robust to changes like illumination and point of view.

In previous section the design of an image feature descriptors called ICAD based on ICA (independent component analysis) for matching image features with a wide base-line from the incoming video at real time with the feature descriptors previously stored in a database. Now in this section we propose to use statistical learning methods for capture variability of image feature appearance. In the learning phase we use small base line tracking methods for capturing the appearance of the image feature along the time, then we build their descriptors, assigning a specific label; these labeled descriptors will be the input to the statistical learning method. In the recognition phase we use the statistical methods to identify the labels of the image features from the incoming video, in real time.

5.3.1 General Methodology

We consider the feature matching wide baseline problem under the context of simultaneous map building and localization of mobile robots (SLAM). The viewing conditions change drastically between the phases of map building (learning) and localization (recognition). Such changes affect both the domain of the image (deformations of the scene and geometric distortion due to changes of the viewpoint) as well as its range (changes in illumination). Such changes are due to both intrinsic properties of the scene (shape, reflectance) and to nuisance factors (illumination, viewpoint).

A feature is a statistic of the image that is intended to make easier the matching process; ideally one would want a statistic feature invariant to all kind of changes. Invariant descriptors to changes like scale or rotation can be created from one single view and they are suitable for many applications, nevertheless in the context of mobile robot, changes like viewpoint or illumination can be appreciably significant between learning and

recognition stages. Unfortunately there exist no single view statistic that is invariant respect to the point of view or lighting conditions.

On the other hand in the context of mobile robots a high frame-rate video is available during both building and localization. So multiple adjacent views of the same scene are available, as for instance in a video from a moving camera and, at least in theory, the point of view could be explicitly accounted for. Additionally, changes in viewpoint cause irradiance changes that are due to the interplay of reflectance and illumination. In this section we want to find if the performance of a scheme for matching image features in a wide base line based in a single feature descriptor is increased, if we use multiple descriptors from the same feature taken at different time, for training a statistical learning method, and then use this learning method for the matching process.

So in this work we are interested in the results of a general scheme for matching images features in a wide base line, instead of a specific method performance. Next, a general and modular scheme is presented to address the problem, the idea is to attempt different kind of descriptors and statistical methods of learning and classification, the modularity makes possible to interchange different methods and descriptors.

Therefore, the scheme is divided into two stages: learning (map building) and recognition (localization).

5.3.2 Learning Phase:

Small base-line tracking: Features are detected and tracked using a conventional small base-line tracker, specifically in this work the Lucas-Kanade Tracker (KLT) was used, but any efficient tracker could be used.

Window extraction: For each feature detected a p-by-p pixels window around the feature center is extracted, in our work we used a 12-by-12 pixel window. Other window sizes have been tested but the results were worse.

Descriptor creation: A descriptor x_i is obtained for each window area. A good descriptor has to be as invariant as possible to changes like rotation or scale and insensitive to changes like illumination. We used two statistical techniques: principal component analysis (PCA) and independent component analysis (ICA), but others descriptors like SIFT can be used.

Storage in database: For each frame, descriptors have to be scaled and stored in a data base. Descriptors created from the same feature tracked along the time with small base-line are stored with the same label y_i.

Feature class creation: A statistical method is used in order to create and represent the descriptors stored in the database with the same label y_i like a unique class V that represent the feature. This feature class V is created with the purpose of capturing the variations in the appearance of the feature along the time. The capacity of V to represent these variations depends on the number of descriptors and the changes of

scene conditions in which it was created. For this study we used Support Vector Machine (SVM) and variations of K-Nearest Neighbor (KNN).

5.3.3 Recognition Phase:

Feature Detection: In order to improve the computational performance in the recognition phase, the same small base-line tracker in the learning phase is used to detect features, but it is not used to track feature candidates to match.

Descriptors candidates: For each candidate feature a descriptor x_j is created in the same way that it was created in the learning phase.

Recognition: The same statistical method used in the learning phase is used now to classify the candidate descriptor x_j in the more adequate feature class V. Depending on the kind of selected statistical method a quality scheme of correspondence between the candidate descriptor and its associated class can be implemented.

5.3.4 Statistical Methods and Descriptors

In this work we employ two image features descriptors: The ICAD and the PCAD, similar to ICAD descriptor, but using PCA (principal component analysis) instead of ICA (independent component analysis). We used two statistical learning methods: KNN (k-nearest neighbor) and SVM (support vector machine).

The idea is comparing the performance of ICAD and PCAD alone, with their performance using them together with KNN and SVM. Consequently in the experiments we are comparing six methods: ICAD, PCAD, SVM-ICA, SVM-PCA, KNN-ICA, and KNN-PCA. In this section first we explain the theoretical bases for SVM and KNN (PCA and ICA were explained in Section 5.2.1), and later, we briefly explain our methods used for this approach.

5.3.5 SVM and KNN

Support Vector Machine (SVM) is a technique used in data classification. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

Given a training set of instance-label pairs $(x_i,y_i), i = 1,\ldots,l$ where $x_i \in \mathrm{R}^n$ and $y \in \{1,-1\}^l$, the SVM [149] require the solution of the following optimization problem:

$$\min_{w,b,\xi} \left(\frac{1}{2} w^t w + C \sum_{i=1}^{l} \xi_i \right), \text{with } C > 0 \qquad (5.12)$$

Subject to

$$y_i(w^t\varphi(x_i)+b) \geq 1-\xi_i \ \text{, where } \xi_i \ \geq 0 \tag{5.13}$$

Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function φ. Then SVM finds a linear separating hyper-plane with the maximal margin in this higher dimensional space. C is the penalty parameter of the error term. Furthermore $K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j)$ is called the kernel function. In this work a radial basis function (RBF) was used like kernel function:

$$K(x_i x_j) = \exp\left(-\gamma\left\|x_i - x_j\right\|^2\right), \gamma > 0 \tag{5.14}$$

The K-nearest neighbor (KNN) is statistical method of classification well known and very simple, nevertheless has come to demonstrate to be very effective in a wide variety of applications. It works based on minimum distance from the query instance x_j to the training samples (x_i,y_i), to determine the K-nearest neighbors. After we gather k nearest neighbors, we take simple majority of these K-nearest neighbors to be the prediction of the query instance.

We use a 5-nearest neighbor; we use a parameter α like threshold for considering a good match between an x_j candidate descriptor and the selected more voted feature class V:

$$\alpha = \frac{K\sum_{j=1}^{l} d(x_j, x)}{l^2} \tag{5.15}$$

Where $x_j \in V$, $d(x_i,x)$ is the Euclidean distance and l is the number of votes for the more voted class V. In this way the average Euclidean distance is used like threshold but is penalized according to the number of votes received for V respect to K.

5.3.6 ICAD and PCAD Methods

Many ICA and PCA algorithms are available. A computationally efficient ICA algorithm, called the FastICA [145] algorithm and the PCA Snapshot Method [150] have been chosen for this work.

When the KLT (small base-line tracker) locates a feature (feature i at frame f), a p-by-p pixels window around the feature center is stored as a vector u_{fi} of length p-by-p, with a distinctive label; in the following frames the feature is tracked and repeating the above process, storing the window with the same label. Immediately vectors with the same label (same feature i) are regrouped in a matrix $U_i = [u_{1i}, \ldots, u_{ni}]^T$ where n is the number of frames where the feature has been tracked.

Then for each matrix U the ICA or PCA is applied as it has been shown in the section 5.2.2 along with dimensional reduction selecting the largest eigenvalue to be retained.

At the output of the ICA or PCA we obtain a descriptor q_i with a dimension that equals to the feature window size. The descriptors are stored in a database with a unique label for each feature.

In the recognition phase features are detected but not tracked by the KLT for each incoming frame. Then for each feature detected a window is obtained in the same way than the learning stage and sorted in a vector v_i, the ICA or PCA is directly applied to this vector without dimensional reduction producing a descriptor x_j. A fast k-nearest neighbor algorithm is applied to the database as was explained in section 5.2.3.

5.3.7 SVM-ICA and SVM-PCA Methods

We used the LIBSVM [151] for the implementation of the SVM. The method follows exactly the same steps than the feature-class method (section 5.3.1): In step 3 (descriptor creation) unlike the ICAD and PCAD methods, the ICA or PCA is applied directly to the vector u_{fi} obtained from de p-by-p pixels window (step 2) and stored in the database (step 4). In the output of the ICA or PCA we obtain a descriptor with the same dimension than the pixel window.

For step 5 (feature-class creation) we employ the descriptors-database for training a SVM classifier with a radial basis function (RBF) as a kernel function, equation 4.

The parameters $C = 8$ and $\gamma = 1$ used in RBF were selected by cross-validation and grid search. For the recognition phase the SVM output model is used to predict the feature class V of the candidate descriptor x_j, as it has been explained in section 5.3.1 (recognition phase).

5.3.8 KNN-ICA and KNN-PCA Methods

For the implementation of KNN we employ a computationally efficient algorithm called approximate nearest neighbor (ANN) [148]. The method follows the same steps than SVM-ICA and SVM-PCA except that a training model is not generated from the descriptor-database. Prior to the recognition phase the whole database is loaded in memory by the ANN algorithm. In experiments we used 5-NN. For recognition phase ANN is applied as it has been explained above. The threshold α_t is used to consider a good match between a candidate descriptor x_j and the more voted selected feature class V (5.15).

5.3.9 Experiments

We have implemented a C++ version of the methods that runs on a PC 2GHz Pentium IV processor, 512MB RAM. A non-expensive USB Webcam with a maximum resolution of 640-by-480 pixels and 30 fps has been used.

Similar experiments to those that were made in section 5.2.4 with ICAD were performed in order to show the performance of the aforementioned methods. For each method in the learning phase, a video sequence of a rigid environment desktop scene was recorded moving the camera slowly and continuously in order to obtain a change of some degrees in the 3D point of view and rotation of the camera. Later, twenty descriptors were created from this video sequence as it was described in section 5.3.4. Together with these 20 descriptors, the database contains another 1000 descriptors corresponding to other video sequences. The objective of these experiments consists of observing the response of the methods when a set of descriptors coming from an online video sequence (close to the learning sequence, as it is explained below) will be matched with the descriptors database. The response was observed in four different cases:

a) Change in 3D viewpoint with respect to the position in the learning phase and little change in illumination.
b) Change in 3D point of view plus change in rotation and little change in illumination.
c) Change in 3D point of view plus change in scale (camera zoom) and little change in illumination.
d) Change in 3D point of view plus great change in illumination.

The following measurements were used in order to comparing the performance of the methods:

Error of classification: It is the ratio between the percentage of false positive and percentage of classification. We define percentage of false positive as the ratio between the number of features wrong classified and the total of features classified in the scene (correctly or incorrectly). We define percentage of classification as the ratio between the total of features classified in the scene (correctly or incorrectly) and the total that potentially could be matched (we consider only a finite number of possible locations in each frame to be matched, step 1 of recognition phase). For example in the methods based in KNN the percentage of false positive for a threshold α =0 .6 could be 16 percent but the percentage of classification is 50 percent, consequently we define the error of classification as 0.32. On the other hand in SVM is difficult to establish a threshold for classification, because SVM is a method of the kind “choose the best candidate”. Therefore we consider the percentage of classification for SVM as 100 percent. For example a 38 percent of false positive in SVM means 38 percent of classification error.

Computational cost: We have calculated the time for each frame (or the frequency, it is the same) that the different methods take to classify 30 possible features with the 1000 descriptors that are in the database.

The results for the experiments and the measurements are shown in Table 5.1.

Case	PCAD	ICAD	SVM-PCA	SVM-ICA	KNN-ICA	KNN-ICA
a	.40	.35	.33	.21	.26	.21
b	.47	.45	.44	.38	.31	.32
c	.47	.33	.41	.28	.50	.35
d	.48	.47	.43	.42	.37	.32
CPU	5.34Hz	5.05Hz	2.32Hz	2.20Hz	5.34Hz	3.84Hz
CPU*	16.30Hz	10.70Hz	3.22Hz	2.94Hz	21.72Hz	7.09Hz

Table 5.1: Error of classification for each condition case (a,b,c and d) in the recognition phase and their computational cost. CPU* does not include the time to detect features by KLT tracker.

In the results of the experiments Table 5.1, we can observe a lower error of classification in the methods based in statistical learning (SVM-PCA, SVM-ICA, KNNPCA and KNN-ICA) comparing with ICAD and PCAD methods. Therefore the performance of the ICAD and PCAD methods was increased using statistical learning approaches.

On the other hand, as we expected, ICA-based descriptors show lower error of classification than PCA-based descriptors but computationally the cost for ICA is greater than PCA. We observe a similar computational cost in the case PCAD and ICAD with KNNPCA and KNN-ICA. Finally we also observe that KNN shows better performance than SVM in error of classification as well as in computational cost.

5.4 SIFT and SURF for Bearing-Only SLAM

There are many kinds of sensors to archive SLAM. Vision-based SLAM uses cameras as the only sensors. Obviously images themselves cannot be input to any SLAM algorithm so extracting good image features is a prerequisite for all vision-based SLAM solutions. We should extract enough but not too many features which can be obtained efficiently, matched reliably and located accurately for vision-based SLAM applications.

These criteria come from three important requirements of vision-based SLAM. First, SLAM algorithms must run in real time. Second, state updating requires reliable data association, which is related directly to reliable feature matching. Third, in Bearing-Only SLAM 3D positions of features cannot be obtained from a single image, accurate and well-conditioned features have to be initialized with an initialization algorithm, which requires accurate image feature locations.

In this section we propose a simple but efficient alternative for addressing the data association problem in a context of Bearing-Only SLAM. The method is based on SIFT

and SURF descriptors. Usually those descriptors are difficult to include directly in Bearing-Only SLAM algorithms. On the other hand, making some “tricks”, these descriptors can be used for reliable data association in Bearing-Only SLAM algorithms.

5.4.1 Challenges for using SIFT and SURF

The approaches presented above (section 5.3) share the idea of using several frames for collecting information in order to create the features descriptors. On the other hand, there may be some applications where is convenient to match features using only two frames: one for the learning phase and other for the recognition phase. In that sense it has been seen that SIFT and SURF descriptors represented the most reliable solution to the data association problem when descriptors are created from a single frame [141], [142].

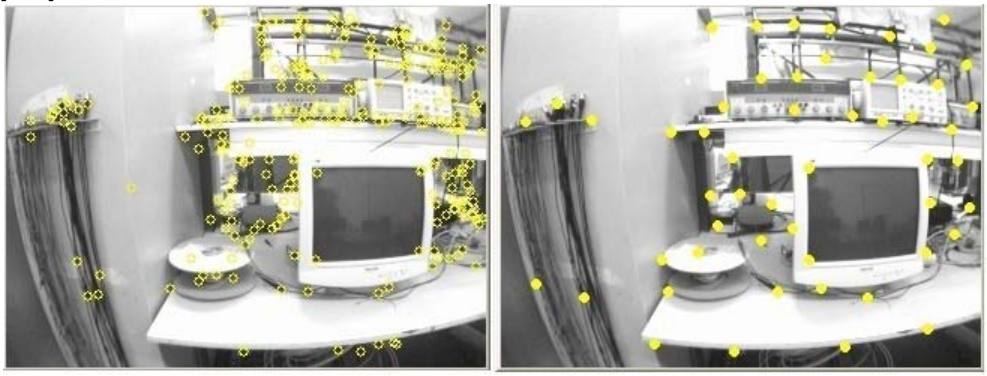


Figure 5.6 In this image 340 SIFT keypoints were detected (left). On the same image, 55 features were found by the Harris Corner detector. It is easy to tune the Harris corner detector for locating strong and spatial-sparse features.

However the original SIFT and SURF algorithm is no practical for Bearing-Only SLAM especially in environments with rich texture. For example, usually from the thousands of SIFT features extracted from one image only 10% features can find their matches in another image even when the change between the two viewpoints is small. In addition, some of the matches are incorrect. In Figure 5.6 (left) it can be appreciated that too many SIFT keypoints were detected in areas with rich texture. In Figure 5.7 shows an example of the matches founded using SIFT descriptors over images taken from different point of view. SIFT and SURF descriptors in general gives excellent results but also produces some false-positives. Unfortunately Kalman-based SLAM is extremely sensitive to the mismatches. Since a SLAM algorithm should run in real time, extracting a large number of features which can hardly find their matches does not support the requirement of speed. In feature initialization algorithms for Bearing-Only SLAM, all the features extracted from an image will be put into the initialization procedures and outliers will be

discarded gradually by different methods based on reliable matching. As a result, there is no way to run any feature initialization algorithm online if an appropriate number of good features cannot be extracted and mismatches cannot be pruned. Consequently, how to extract good features for vision-based SLAM is critical.

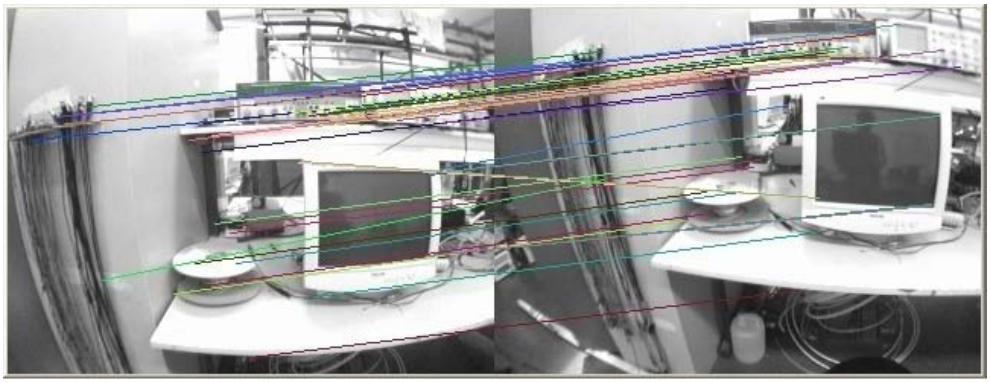


Figure 5.7 Two images were taken with different point of view of some area in a laboratory. 46 matches were found using SIFF descriptors, 6 been false positives. In general SIFT descriptors gives impressive results; however in Kalman-based SLAM a few mismatches can be enough for producing divergence in the filter.

Been the most robust feature detectors, SIFT and SURF can hardly be used in any Bearing-Only SLAM algorithm (especially for un-delayed algorithms). There are two main reasons. The first is that too many features are extracted with a low matching ratio, so no feature initialization algorithm or particle filter related algorithms can afford the calculation cost. The second is that unless on uses small size images, SIFT or SURF algorithm are not fast enough to be used for real time SLAM algorithms.

In [152] a feature extraction method is proposed for selectively detect SIFT features based on analyzing stable matching ratios at different scales, for detecting a limited but sufficient number of SIFT features for Bearing-Only SLAM. Below we present a simpler alternative that we guess, gives acceptable results.

5.4.2 Combining Descriptors and Features Corners

In [130] it was found that the Harris Corner detector is the most suitable point detector for vision-based SLAM, mainly, due to its characteristics of repeatability and saliency. Another attribute of this kind of saliency operators (Harris, Shi-Tomasi, Canny etc.) is that they are easy to tune in order to detect sparse features in the images, in counterpart to the SIFT and SURF descriptors. Figure 5.6 (right) shows an example of the points extracted by the Harris Corner Detector, in this case 55 features were extracted

compared with the 340 SIFT key-points. Moreover corners can be rapidly extracted from the images. All those attribute have motivated their use in vision-based SLAM approaches.

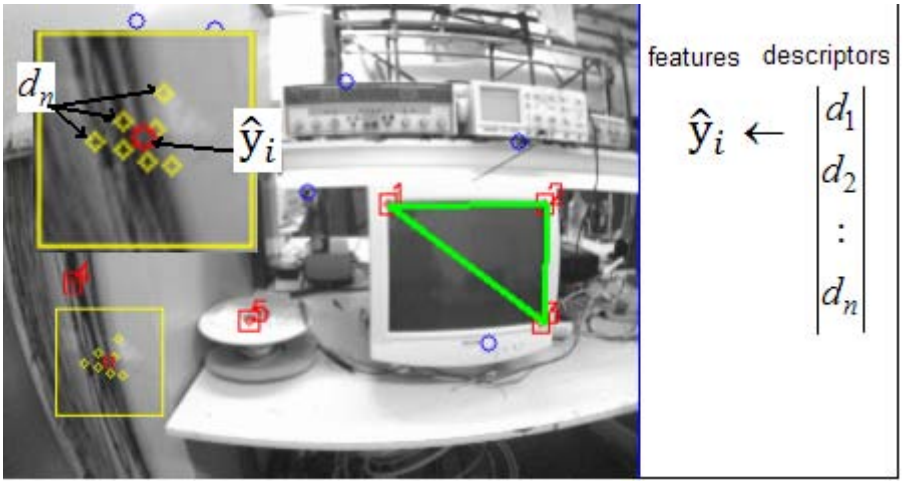


Figure 5.8 A saliency operator like Harris detector is used to detect points of interest to initialize new features y_i. When y_i is initialized, descriptors are extracted from a *p-by-p* pixels patch centered in y_i and associated to a feature.

Usually corner-features together with gray level patches and cross-correlation techniques are used in vision-based SLAM for matching the features frame to frame because the correspondence problem is easy in a small base line. Nevertheless when the vehicle travels far away and return to a previously mapped sited the appearance of the features can drastically varying due the changes of illumination or view point. In this scenario features descriptors like SIFT or SURF have shown to be the most reliable option for addressing the data association problem. However we also have seen that is hard the use of this kind of descriptors directly in SLAM. Mainly due to SIFT and SUFT produces too many keypoints and also due to their computational cost.

The proposed data association approach is very simple but takes advantage of the better characteristic of both techniques: Saliency operators and Descriptors:

In SLAM new features are gradually added to the map, in this case when a new feature is initialized (Figure 5.8), the proposed approach is to apply some of the saliency operators used in most of the vision-SLAM methods (Shi-Tomasi, Canny, etc) and to extract n SURF or SIFT descriptors d_i from a *p-by-p* patch centered in the point detected by the saliency operator. All extracted descriptors are stored and related to with the $\hat{\mathbf{y}}_i$ feature. Because usually in SLAM one feature is initialized at once the computational cost of extracting descriptors from a patch of a *p-by-p* pixels (in experiments we use 40-by-40 patches) is really minimum compared with computational cost of extracting descriptors

from the entire image. Then when a feature is initialized, their related d_n descriptors are stored in a data base and labeled each one with the corresponding $\hat{y}_i$ feature. In Figure 5.8 it can be appreciated that seven descriptors were detected in the patch related to $\hat{y}_i$ feature. In this way, each feature $\hat{y}_i$ can have a lot of useful information in order to be matched in the future.

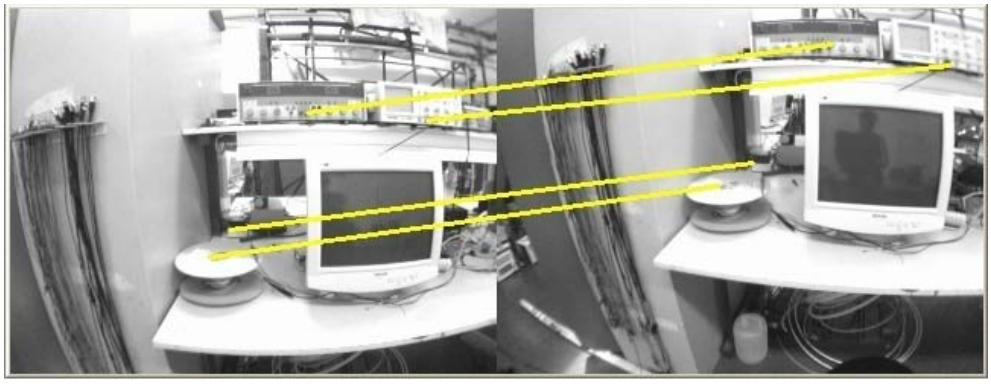


Figure 5.9 Using the combined features four strongest matches were found.

Of course, it could be happen that no descriptors are extracted for a related feature; in this case the features are only useful for small base-line tracking. On the other hand in SLAM a few good matches are enough in order to minimize the drift in the estimates.

When the combined feature-descriptors wants to be matched (with a wide base-line) and for minimizing the probabilities of mismatch, a matching of an incoming feature with an already mapped one is only considered under the following circumstance: at least two different descriptors d_n from the measured feature must match with two different descriptors related with a single feature in the map. A fast approximate *k*-nearest neighbor technique is used for matching descriptors. A positive matching is considered if the Euclidean distance to the second d_{k2} nearest neighbor is shorter than $0.7d_{k1}$ to the nearest neighbor.

In Figure 5.9 shows the results of applying the proposed methodology to the same images used in the SIFT experiment (Figure 5.7). For comparison purpose we also used SIFT descriptors but other descriptors as SURF can be used. In the figure, it can be appreciated that four matches (all correct) were found. The proposed methodology is congruent with the vision-based SLAM schemes because a less but sufficient number of rich-features could be initialized in the map. This features contain useful information (in the form of related descriptors) to be matched in the future.

5.5 Conclusions

The data association problem plays a fundamental role in SLAM. Cameras yields a huge amount of information related to the appearance of the landmarks and therefore they are an excellent option for addressing the data association problem. In this chapter we have presented several approaches for solving the data association problem using appearance-based local features descriptors.

First we have described a method called ICAD for matching image features in a wide based line. This method is based on ICA descriptors. In general the method shows good results, especially in terms of computational performance. Nevertheless the response is still a little sensitive respect to some changes like point of view. In general ICA descriptors represent an intermediate alternative, having an intermediary performance in, computational cost and robustness, respect to SIFT and SURF descriptors on the one hand and the Gray level patches on the other: ICA descriptors are more robust than Gray level patches but less than SIFT or SURF. ICA descriptors also have a lower computational cost than SIFT or SURF but a little more expensive than the Gray level patches.

Later, a study of the application of statistical methods (SVM and KNN) for capturing the variability of image feature appearance was presented. The experiments were performed over different variations of ICAD descriptors. The results are very promising since they show that the proposed methodology increased the robustness of the descriptors. One characteristic of the proposed methodology is the modularity, so it can be extended for their application with other kind of descriptors and statistical methods.

Finally, it was seen that the application of SIFT or SURF descriptors to SLAM is not straightforward. In that sense, a simple but effective framework based on combining descriptors and saliency operators, like the Harris corner detector, was presented. The experiment show that a lower (but sufficient for SLAM) number, all correct, of matches were founded. In the next chapter, a real SLAM application is presented, based on this technique, for matching features with a wide base-line.

Chapter 6

Monocular SLAM

Cameras have become more and more interesting for the robotic research community as sensors, because they yield a lot of information. It is a sensor from which 3D information can be extracted. Even for indoor robots whose pose can be represented in 2D, the ability to gather 3D information on the environment is essential. Cameras are well adapted for embedded systems: they are light, cheap and power saving. As computational power grows, an inexpensive camera can be used to perform range and appearance–based sensing simultaneously, by replacing typical sensors as laser and sonar rings for range measurement and encoders for dead reckoning. A wide variety of algorithms can be obtained from the vision research community in order to extract high level primitives from the image, and matching them with primitives stored in the map thus allowing reliable data association. This is one the most important problems to solve in SLAM.

In this context, the 6-DOF monocular camera case (Monocular SLAM) possibly represents the harder variant of SLAM. Monocular SLAM is closely related to the structure-from-motion (SFM) problem of reconstructing scene geometry. SFM techniques ([153], [154]) have been used successfully for recovering camera position and scene structure. However, the SFM techniques coming from the vision community research have been formulated as off-line algorithms and required batch, simultaneous processing of all the images acquired in the sequence. In contrast to SFM approaches that rely on global nonlinear optimization, recursive estimation methods allow online operation, which is highly desirable for a SLAM system. Some hybrid techniques (SFM-Kalman Filtering) as quoted in reference [155], which is based on stereo-vision, have also appeared. Nevertheless some of the drawbacks (in a purely robotic context) of these methods remain due to the global optimization nature of the SFM methods.

In section 6.1 an introduction to the monocular SLAM topic is presented. In this section the main (and current) challenges are pointed out and the particularities and

differences of the monocular SLAM in relation to the general Bearing-Only SLAM are explained. A small survive in monocular SLAM is also included, presenting the most relevant related-work on the matter. Finally the viability of the monocular SLAM systems is reviewed.

In section 6.2 a novel scheme for monocular SLAM is presented. This approach, called Delayed Inverse Depth Monocular SLAM method, is an extension of our general Bearing-Only SLAM algorithm, described in chapter 4, for being used in a monocular SLAM context. This approach aims to contribute to the robustness of Monocular SLAM systems.

One of the major challenges in monocular SLAM consist in extend the application of the current methods to large and dynamic environments. In section 6.3 a novel framework, called Distributed Monocular SLAM is proposed, in order to address the problem of building and maintaining a global and consistent map of large environments at real time.

Finally in section 6.4 the conclusions of the chapter are given.

6.1 Introduction

Monocular camera is a projective sensor which measures the bearing of images features. Therefore depth information cannot be obtained in a single frame. To infer the depth of a feature, the camera must observe it repeatedly as it translates though the scene, each time capturing a ray of light from the feature to its optical center. In chapter 5 we present several data association techniques that can be potentially used for detecting, tracking and matching visual features in monocular SLAM. The angle between the captured rays is the feature's parallax, allowing depth estimation. Features need an especial treatment before to be added to the stochastic map. In chapter 4 we have presented techniques for initializing new features in Bearing-only SLAM.

In this chapter we deal with the hardest variant of Bearing-Only SLAM: the 6-DOF monocular camera case (Monocular SLAM). In this extreme case the only sensory input to SLAM is a single low-cost "webcam", with no odometry, inertial sensing or stereo capability for direct depth perception. An example is a camera carried by a walking person.

In chapter 4 the general Bearing-Only problematic was introduced. It was seen that one of the hardest challenges is related with the initialization of new features, due to bearing sensors does not provides depth information directly. In monocular SLAM there are some additional issues that have to be addressed. In algorithms presented in chapter 4, it was assumed the robot Odometry as the control input system, but now in monocular SLAM, we are considering a single camera as the only available sensor. This fact leads to a second issue: the scale of the map. So far, odometry has also been used in order to retrieve the scale of the map, nevertheless the scale of the observed world cannot be

obtained using a bearing only sensor. In spite of these new challenges, as we will see, the monocular SLAM problem has a huge potential.

6.1.1 Unknown Control Input

So far we have been considering the robot odometry as the control input of the system. In this case the Kalman prediction step is a function of discrete motion prediction model and the control input $u(k)$:

$$\hat{\mathrm{x}}_{v(k+1)} = f\left(\hat{\mathrm{x}}_{v(k)}, u(k)\right); u(k) = \begin{bmatrix}\delta_l \\ \delta_r\end{bmatrix} \tag{6.1}$$

$$\begin{bmatrix}x_{v(k+1)} \\ y_{v(k+1)} \\ \theta_{v(k+1)}\end{bmatrix} = \begin{bmatrix}x_{v(k)} + \delta\cos\theta' \\ y_{v(k)} + \delta\sin\theta' \\ \theta_{v(k)} + \theta'\end{bmatrix} \tag{6.2}$$

being δ the distance traveled by the robot's center and θ' the turn realized by the robot at each instant k:

$$\delta = \frac{\delta_l + \delta_r}{2}, \theta' = \frac{\delta_l - \delta_r}{b} \tag{6.3}$$

derived from the distance traveled for each robot's wheels δ_l, δ_r and the separation b between them. Where δ_l, δ_r are obtained from the robot's encoders. In this case the uncertainty is modeled inserting a noise term into the control signal u such that:

$$u(k) = u_n(k) + v(k) \tag{6.4}$$

where $u_n(k)$ is a nominal (intended) control signal and $v(k)$ is a zero mean Gaussian distributed noise vector:

$$v(k) \sim \aleph(0, \begin{bmatrix}\sigma_{\delta_l}^2 & 0 \\ 0 & \sigma_{\delta_r}^2\end{bmatrix}) \tag{6.5}$$

$$u(k) \sim \aleph(u_n(k), \begin{bmatrix}\sigma_{\delta_l}^2 & 0 \\ 0 & \sigma_{\delta_r}^2\end{bmatrix}) \tag{6.6}$$

In our current monocular SLAM context, we assume that lectures δ_l, δ_r obtained from the vehicle encoders are not available, and therefore we don't have such prior information about the camera-vehicle movement. However, it is important to remember that both cases are just points on the continuum of types of model for representing physical systems. Every model must stop at some level of detail and a probabilistic assumption is made about the discrepancy between this model and the reality: This is what is referred to as process uncertainty (noise). In the case of a wheeled robot, this uncertainty term takes account of factors such as potential wheel slippage, surface

irregularities, and other predominantly unsystematic effects which have not been explicitly modeled. In the case of and camera-robot, it takes account of the unknown dynamics and intentions of the human or robot carrier, but these too can be probabilistically modeled [156].

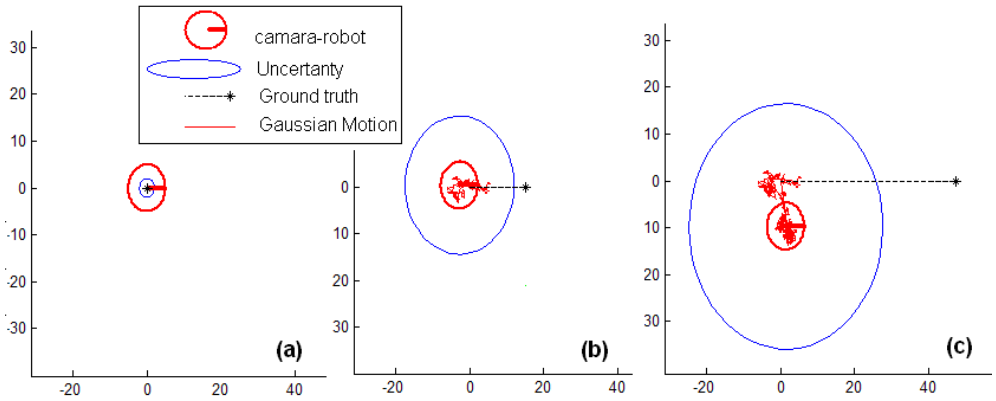


Figure 6.1 Gaussian motion simulation (plots a, b and c). In this simulation the robot moves in a simple straight trajectory. Nevertheless, because there is not available odometry, then the robot's movement is modeled as a Gaussian (random) motion. Note how the uncertainty increases over the time.

Because we cannot make a prior assumption about the camera-robot movement then the idea is to model it like a Brownian motion (Gaussian Motion). Brownian motion is among the simplest stochastic processes, and it is a limit of both simpler and more complicated stochastic processes.

In this case the simplest model is such that represent the control signal u as an uncorrelated Gaussian noise.

$$u(k) = v(k) \tag{6.7}$$

$$v(k) \sim \aleph(0, \begin{bmatrix} \sigma_{x_v}^2 & 0 & 0 \\ 0 & \sigma_{y_v}^2 & 0 \\ 0 & 0 & \sigma_{\theta_v}^2 \end{bmatrix}) \tag{6.8}$$

This produces a discrete motion prediction model:

$$\begin{bmatrix} x_{v(k+1)} \\ y_{v(k+1)} \\ \theta_{v(k+1)} \end{bmatrix} = \begin{bmatrix} x_{v(k)} + \sigma_{x_v}^2 \\ y_{v(k)} + \sigma_{y_v}^2 \\ \theta_{v(k)} + \sigma_{\theta_v}^2 \end{bmatrix} \tag{6.9}$$

Figure 6.1 shows a simulation of a Gaussian motion using the model described in equation (6.9). In this case landmarks have not been included in the simulation. In this context (without sensorial information) Brownian motion is reasonable manner for modeling the (unknown) control signal input. In the graphics it can be appreciated the random movement of the robot. Of course along with this kind of model of motion, there is an increase on the uncertainty, which is represented in the state covariance matrix.

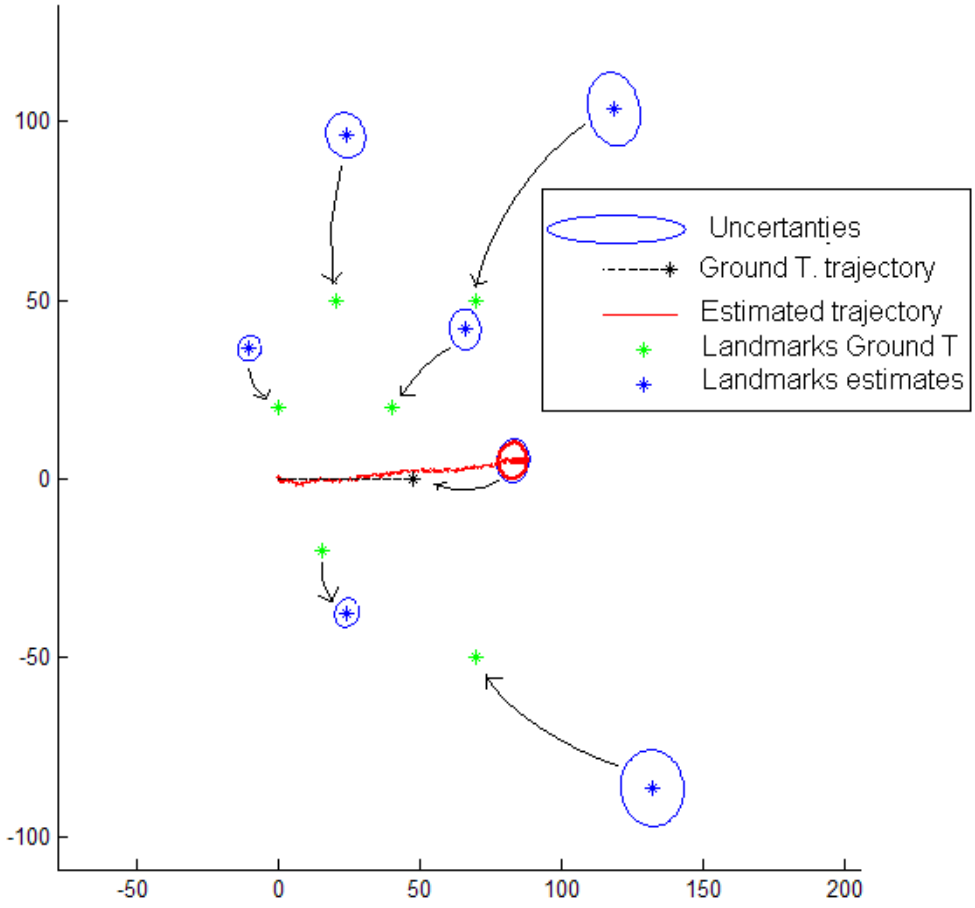


Figure 6.2 Bearing-Only SLAM Simulation using Gaussian noise as control input. Note that the map and robot trajectory have been reasonably well estimated but with a different scale respect to the ground truth. When the bearing sensor is the only source of information available then the scale of the world cannot be retrieved.

For the simulation illustrated in Figure 6.2 several landmarks have been included. In counterpart to the previous simulation (where the robot is completely blind) in this case the robot is capable of measuring the bearing respect to the landmarks.

When bearing information is available; the random movement of the robot, (produced in the prediction step of the Kalman Filter) is bounded for every incoming bearing measurement, (in the Kalman update step) making possible that the estimated movement direction of the robot converges to the real one. At the same time, if the movement of the robot is reasonable well predicted (emulating the availability of odometry) then the location of the landmarks can also be estimated. And moreover, if the landmarks locations are well estimated then the estimated robot's location is improved.

6.1.2 The Scale of the Map

In the previous section, it was seen that is possible to replace the odometry input for Gaussian noise (emulating a random walk) if bearing information is the solely information available in the system. On the other hand in Figure 6.2 can be clearly appreciated a collateral effect in the estimated map and trajectory when odometry is not available: The scale of the world cannot be retrieved using only bearing information.

In Figure 6.2 it can be seen that the trajectory and map where reasonably well estimated but in a different scale respect to their ground truth. For some applications the indeterminacy of the scale could not be a matter. In [157] a monocular SLAM approach is proposed where up-to-scale quantities are estimated using a dimensionless parameterization.

This thesis focuses on the robotics applications of the SLAM, and consequently it is important to consider the metric of the world. In this case, when bearing information is the barely source of information, a manner of retrieving the scale of the world is to provide some degree of initial metric information to the system.

Therefore, the simplest solution is initializing the system with some known features in the map.

In this case these features, called initial features of reference, with all their parameters known, play the role of a metric reference. Each feature of reference $\hat{y}_{ref}$ defined by:

$$\hat{y}_{ref_i} = \left[\hat{x}_i, \hat{y}_i, \hat{\theta}_i, \hat{\rho}_i\right]^{\mathrm{T}} \tag{6.10}$$

is included in the map, previously to the first Kalman step:

$$\hat{\mathrm{x}}_{\mathrm{ini}} = \begin{bmatrix} x_v \\ y_{ref_1} \\ \vdots \\ \hat{y}_{ref_n} \end{bmatrix} \tag{6.11}$$

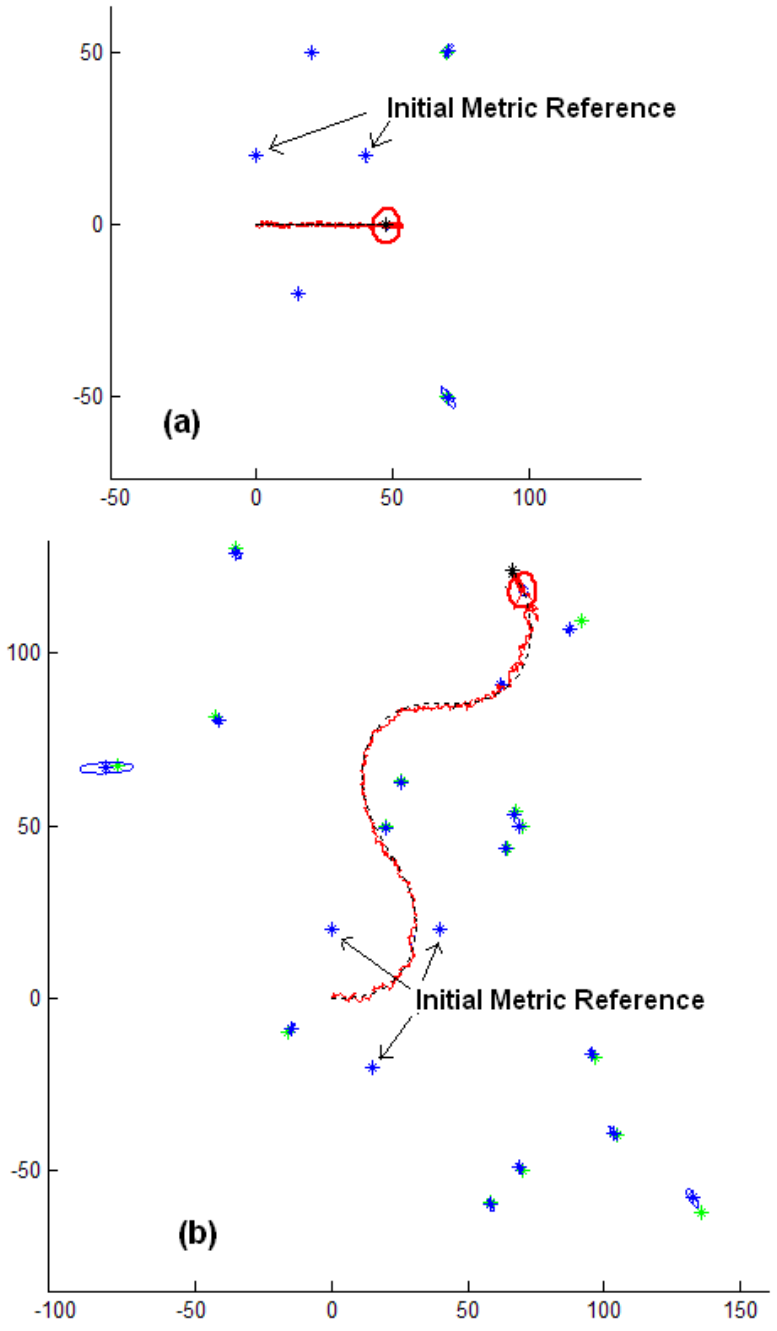


Figure 6.3 Simulations using an initial metric reference for recovering the scale of the world.

Since the parameters of the initial features of reference $\hat{y}_{ref}$ are perfectly known, then their corresponding variances in the state covariance matrix P are set to zero or a nearly zero value.

For example an initial covariance matrix P_{ini} including a single feature of reference $\hat{y}_{ref}$ would be as simple as:

$$P_{\text{ini}} = \begin{bmatrix} \underbrace{\begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix}}_{x_v} & \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \underbrace{\begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{Bmatrix}}_{y_{ref\ 1}} \end{bmatrix} \tag{6.12}$$

Initializing initial features of reference in the state covariance matrix with zero allows that, the metric scale of subsequent new features converges to the same metric scale established by these initial features of reference. Moreover, due to the strong correlation between map and trajectory estimates, then the estimated trajectory of the robot also converges to a metric scale established by the initial features of reference.

Figure 6.3 (a) illustrates the simulations results for the same experiment showed in Figure 6.2, but in this new case, prior to the first Kalman update, two features have been manually added to the map (illustrated in the plot as metric reference). It can be clearly appreciated how the map and trajectory converges to the ground truth.

Figure 6.3 illustrates a more complex simulation using 20 random landmarks and moving the robot in an undulate trajectory. In this case three landmarks have been chosen for being initialized in the map as initial features of reference. For this case the estimated map and trajectory also converges to their ground truth.

So far we have seen that is likely to perform SLAM using a bearing sensor as the only input of information of the system. In subsequent sections our 2D-2DOF Bearing-Only SLAM algorithm will be extended to a 3D-6DOF monocular SLAM context.

6.1.3 Current Challenges in Monocular SLAM

Simultaneous Localization and Mapping (SLAM) and a real-time (30fps) monocular implementation was first described by Davison (recently summarized in [156]). After that and due to a huge potential, several proposals have emerged (most of them, reviewed in chapter 4).

A long term goal in SLAM shared by many would be to achieve a system with the following performance: A single low-cost camera attached to a portable computer would be switched on at arbitrary location in an unknown scene, then carried off by a fast-moving robot (perhaps flying or jumping) or even a running human through an arbitrarily

large domain, all the time effortlessly recovering its trajectory in real time and building a detailed, persistent map of all it has seen [156].

Nevertheless, currently there are two main problems with most existing monocular SLAM implementations:

- *Lack of robustness.*
- *Limited to room-sized domains.*

Lack of robustness: One of the main problems with most existing monocular SLAM implementations is a lack of robustness. Typically tracking systems rely on a prior over current pose and this prior is used to limit the search for visual feature correspondences, yielding very rapid frame-to-frame localization. However rapid camera motions, occlusion, and motion blur violate the assumptions in the prior and therefore can often cause tracking to fail. While this is inconvenient with any tracking system, tracking failure is particularly problematic for SLAM systems: not only is camera pose lost, but the estimated map could become corrupted as well. [158] It presents a monocular SLAM system which can recover from the frame-to-frame tracking failures which are inevitable in real-world operation. Instead of trying to avoid tracking failure altogether, the system automatically detects the failure, halts the SLAM system, and begins relocalizing instead. Mapping is only resumed when the camera pose has been redetermined, thus preserving map integrity. Relocalisation is performed by first using a randomized list classifier to establish feature correspondences in the image and then RANSAC to determine the pose robustly from these correspondences.

Most of the monocular SLAM algorithms have demonstrated SLAM operation when camera motion is smooth and consistent. However real applications will require that operation is maintained even when camera motion is discontinuous and erratic; if applications are to be usable, then it is critical that SLAM operation recovers quickly and in a stable manner following such movement. In several methods, feature detection is generally based on correlating with reference templates. This has the drawback that detection is not invariant to viewing direction and although local predictive warping can alleviate the problem ([159], [115]), such methods are always likely to be of limited utility. Correlating templates is also known to be highly susceptible to detection ambiguity, especially in cluttered environments, and this makes reliable detection difficult to achieve, particularly when search regions grow due to large uncertainty in camera localization. In [160]a variation of SIFT descriptors is proposed for feature detection and tracking. In this approach, for overcome the computational cost limitation of the SIFT for real-time applications, potential features are located using a fast saliency operators, assisted by search regions derived from the covariance within the SLAM filter. And scale invariance is achieved by constructing descriptors over multiple resolutions for each feature only when it is first detected.

Point features are the most common kind of features used in monocular SLAM algorithms, which are relatively easy to localize and characterize. However, their appearance changes substantively with the camera´s location, and while invariant feature descriptors can be used (e.g. SIFT), they can be expensive to compute and with more invariance comes less discrimination. One solution, that is possible within a SLAM framework, is to augment the map of the world with estimates of the planes upon which each feature lies. These can then be used to predict the deformation in the feature, as in [159] which give better performance. On the other hand, alternative kind of features can also be used. [161] Present, a real-time monocular EKF SLAM system, which uses straight lines as its features. Line features provide a camera tracking system with natural illumination and viewpoint invariance, and could provides a more intuitive map, and a compact representation.

Limited to room-sized domains: Another main problem with most of the current approaches of monocular SLAM is that their operation is limited to room-sized domains. For example a major drawback of the system proposed by Davison in [109] relies in the particle-filter-based technique used for initializing new features in the map: due the initial distribution of particles has to cover all possible depth values for a landmark then its work-range is around of 5 meters, which is adequate for a desktop domain application. However its application to large environments is not straightforward, as it would require a huge number of particles. In that sense one of the best contributions to the monocular SLAM systems was made by Montiel in [1]. In this work the proposed Unified Inverse Depth Parameterization makes possible to code in a simple manner features depths from nearby to infinity. This attribute is highly desirable if a monocular SLAM system wants to be applied to larger domains because distant features can be included in the map.

On the other hand, one of the challenges that should be addressed in order to extend the application of the current available monocular SLAM systems to larger domains is related to number of sparse features that can be maintained in the map for real-time operation. Approaches like the presented by Davison or Montiel have shown good result in real-time 6-DOF camera pose and orientation estimation, and building 3D maps of 50-100 sparse features. However if larger trajectories wants to be estimated and maps of bigger environments wants to be built, then a mayor number of features should be handled in the system, without affecting the real time operation.

Some attempts have been made in order to improve the efficiency of monocular SLAM algorithms. In [162] for improving efficiency of the inverse depth parametrization scheme [1], a transform of features to an XYZ encoding is proposed, as soon as this more computational efficient parametrization becomes well-behaved, meaning that a Gaussian distribution in these coordinates is a good fit for the uncertainty in the point location. So, retaining the inverse depth method for features (initialized with six parameters) is important at low parallax, but as the estimation evolves, if the 3 parameters XYZ

encoding becomes well-behaved, the feature is transformed from inverse depth to XYZ. A test for the transformation is also proposed, relating to the feature parallax and estimation accuracy, which is tested individually for each feature at every estimation step. It is shown that algorithm performance does not degrade when compared to keeping every feature with an inverse depth encoding, but computational efficiency is increased by decreasing the state vector size. Using this transformation, the number of features, which can be maintained in the map for real-time operation, can be similar to the quantity of features maintained by the Davison approach (near to 100).

Possibly one of the main drawbacks of the Kalman Filters based methods is related with poor scalability in terms of computational cost of the Kalman Filter. In that sense, [115] presents a system based on the FastSLAM algorithm [73], which combines particle filtering for localization with Kalman filtering for mapping, FastSLAM has the advantage that it scales better with the number of features (this method maintains around of 250 features in the map), but the absence of an explicit full covariance matrix can make loop-closing more difficult.

As we seen in chapter 4, the technique for initialization of features proposed by Montiel in [1] has some drawbacks, mainly related with a lack of robustness for some circumstances. This fact has motivated us to propose the *Delayed Inverse Depth Feature Initialization Method* (also introduced in chapter 4 for a 2DOF odometry context). This method is the core of our general Bearing-Only SLAM algorithm.

In the current chapter, our general Bearing-Only SLAM algorithm is widely extended for working in a fully monocular SLAM context. In that sense, section 6.2 describes a novel approach for monocular SLAM called "*Delayed Inverse Depth Monocular SLAM*". This approach intends to contribute to the robustness of Monocular SLAM systems, by introducing a versatile and robust new method for initializing features in the map.

As we seen above, one of the current limitations in most of the current methods for monocular SLAM is that they are limited to a room-sized working area. In that sense, in section 6.3, a Distributed monocular SLAM Framework is presented for addressing the problem of building and maintaining a global and consistent map of large environments at real time. This framework intends to contribute to the effort of make possible, in a near future, that vision-based SLAM emulates (and ultimately surpass) the results in large-scale mapping achieved using laser range-finder sensors, aiming to build vision-only SLAM systems with the potential of guiding autonomous robots in their exploration and operation in large and complex environments.

6.2 Delayed Inverse Depth Monocular SLAM

In this section, we extend our general Bearing-Only SLAM algorithm, described in chapter 4, for being used in a monocular SLAM context.

6.2.1 Camera Motion Model

Instead a vehicle model, as the described in Chapter 3.2, a free camera-robot moving in any direction in $\Re^3 \times SO(3)$ is considered. The camera state $\hat{\mathrm{x}}_v$ is defined by:

$$\hat{\mathrm{x}}_v = [r^{WC}\ q^{WC}\ v^W\ \omega^W] \tag{6.13}$$

where:

$$r^{WC} = [x_v\ y_v\ z_v] \tag{6.14}$$

represents the camera optical center position, and:

$$q^{WC} = [q_0\ q_1\ q_2\ q_3] \tag{6.15}$$

represents the camera orientation by a quaternion. Unit quaternions provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. Compared to Euler angles they are simpler to compose and avoid the problem of gimbal lock. Compared to rotation matrices they are more efficient and more numerically stable [163].

$$v^W = \left[v_x\ v_y\ v_z\right] \tag{6.16}$$

$$\omega^W = \left[\omega_x\ \omega_y\ \omega_z\right] \tag{6.17}$$

Denote linear and angular velocities respectively. At every step it is assumed an unknown linear and angular acceleration with zero mean and known covariance Gaussian processes, a^{W} and α^{W}, producing an impulse of linear and angular velocity such as:

$$u = \begin{pmatrix} V^W \\ \Omega^{\mathrm{W}} \end{pmatrix} = \begin{pmatrix} a^W \Delta t \\ \alpha^W \Delta t \end{pmatrix} \tag{6.18}$$

An unconstrained constant-velocity camera motion prediction model (proposed in [164] and [109]) is used and is defined by the following equation.

$$f_v = \begin{bmatrix} r_{k+1}^{WC} \\ q_{k+1}^{WC} \\ v_{k+1}^W \\ \omega_{k+1}^W \end{bmatrix} = \begin{bmatrix} r_k^{WC} + (v_k^W + V_k^W)\Delta t \\ q_k^{WC} \times q\left((\omega_k^W + \Omega^{\mathrm{W}})\Delta t\right) \\ v_k^W + V_k^W \\ \omega_k^W + \Omega^{\mathrm{W}} \end{bmatrix} \tag{6.19}$$

being $q((\omega^{W}{}_{k}+\Omega^{W})\Delta t)$ the quaternion defined by the rotation vector $(\omega^{W}{}_{k}+\Omega^{W})\Delta t$. The superscripts ^{WC} and ^{W} denote magnitudes expressed in world reference and camera reference respectively. Figure 6.4 illustrates how this models potential deviations from a constant velocity trajectory.

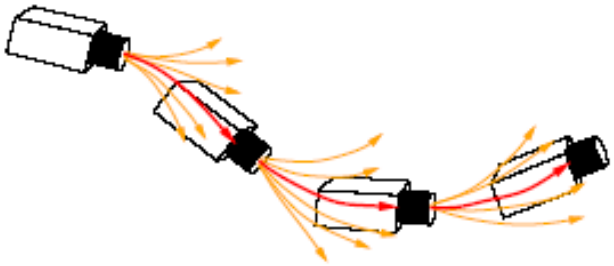


Figure 6.4 Visualization of the "constant velocity" model for smooth motion. Graphic taken from [156].

6.2.2 Camera Motion Prediction in the EKF

The prediction step for the Extended Kalman Filter is defined as follows:

$$\hat{\mathrm{x}}_{\mathrm{k+1}} = \begin{bmatrix} f_v(\hat{\mathrm{x}}_{v_k}, u_k) \\ \hat{\mathrm{y}}_1 \\ \vdots \\ \hat{\mathrm{y}}_n \end{bmatrix} \tag{6.20}$$

$$P_{k+1} = \nabla F_x P(k) \nabla F_x{}^{\mathsf{T}} + \nabla F_u Q \nabla F_u{}^{\mathsf{T}} \tag{6.21}$$

Where, f_v is defined in equation (6.19). We are assuming a static map where the features remain static; hence note that the system state part corresponding to the map does not change in the prediction step. Features $\hat{\mathrm{y}}$ are defined in section 6.2.4.

The Jacobians ∇F_x and ∇F_u are defined by:

$$\nabla F_x = \begin{bmatrix} \frac{\partial f_v}{\partial \hat{x}_v} & 0 \\ 0 & \mathrm{I}_{n\times n} \end{bmatrix} \quad , \quad \nabla F_u = \begin{bmatrix} \frac{\partial f_v}{\partial u} & 0 \\ 0 & 0_{n\times n} \end{bmatrix} \tag{6.22}$$

$$\mathrm{Q} = \begin{bmatrix} U & 0 \\ 0 & 0_{n\times n} \end{bmatrix} \tag{6.23}$$

where n is equal to the dimension corresponding to the map features in the system state, and:

$$U = \begin{bmatrix} (\sigma_v \Delta t)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\sigma_v \Delta t)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\sigma_v \Delta t)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\sigma_\omega \Delta t)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (\sigma_\omega \Delta t)^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (\sigma_\omega \Delta t)^2 \end{bmatrix} \quad (6.24)$$

The values for σ_v and σ_ω proposed in [156] are 6 ms^{-2} and 6 $_{rad}s^{-2}$.

6.2.3 Jacobians for the Camera Motion Model

The Jacobians for the camera motion model f_v (6.19), used in the EKF, can be estimated as follows:

$$\frac{\partial f_v}{\partial \hat{x}_v} = \begin{bmatrix} I_{3\times3} & 0 & \frac{\partial r}{\partial v} & 0 \\ 0 & \frac{\partial q_3}{\partial q_2} & 0 & \frac{\partial q}{\partial \omega} \\ 0 & 0 & I_{3\times3} & 0 \\ 0 & 0 & 0 & I_{3\times3} \end{bmatrix} \quad (6.25)$$

$\partial f_v / \partial \hat{x}_v$ is a 13-by-13 matrix representing the Jacobian of f_v respect to the state vector $\hat{x}_v$. I_{3x3} are identities matrix of size 3-by-3. The rest of components of $\partial f_v / \partial \hat{x}_v$ are defined by:

$$\frac{\partial r}{\partial v} = \begin{bmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \quad (6.26)$$

where

$$\frac{\partial q_3}{\partial q_2} = \begin{bmatrix} q_R & -q_X & -q_Y & -q_Z \\ q_X & q_R & q_Z & -q_Y \\ q_Y & -q_Z & q_R & q_X \\ q_Z & q_Y & -q_X & q_R \end{bmatrix} \quad (6.27)$$

The use of quaternions implies the use of several functions for converting from regular vectors to quaternions and the opposite case. For expressing angular velocities ω^W (expressed in Euler angles) in equations defined by quaternions (as the camera motion model f_v) the rotational vector ω^W is converted to a quaternion q:

$$\begin{bmatrix} q_R \\ q_X \\ q_Y \\ q_Z \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mu_x \\ \sin(\theta/2)\mu_y \\ \sin(\theta/2)\mu_z \end{bmatrix} \quad (6.28)$$

$$\theta = \|\omega^W \Delta t\| \quad \mu = \frac{\omega^W \Delta t}{\|\omega^W \Delta t\|} \quad (6.29)$$

The last component for $\partial f_v/\partial \hat{\mathbf{x}}_v$ is defined by:

$$\frac{\partial q}{\partial \omega} = \frac{\partial q_3}{\partial q_1}\frac{\partial \omega_t}{\omega} \tag{6.30}$$

Where:

$$\frac{\partial q_3}{\partial q_1} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \tag{6.31}$$

and

$$\frac{\partial \omega_t}{\partial \omega} = \begin{bmatrix} \frac{\partial q_0}{\partial \omega} \\ \frac{\partial q_1}{\partial \omega} \\ \frac{\partial q_2}{\partial \omega} \\ \frac{\partial q_2}{\partial \omega} \end{bmatrix} \tag{6.32}$$

Being:

$$\frac{\partial q_0}{\partial \omega} = \begin{bmatrix} -\frac{\Delta t}{2}\frac{\omega_x}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \\ -\frac{\Delta t}{2}\frac{\omega_y}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \\ -\frac{\Delta t}{2}\frac{\omega_z}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \end{bmatrix}^{\intercal} \tag{6.33}$$

$$\frac{\partial q_1}{\partial \omega} = \begin{bmatrix} \frac{\Delta t}{2}\frac{{\omega_x}^2}{{\omega_{mod}}^2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) + \frac{1}{\omega_{mod}}\left(1 - \frac{{\omega_x}^2}{{\omega_{mod}}^2}\right)\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \\ \frac{\omega_x\omega_y}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \\ \frac{\omega_x\omega_z}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \end{bmatrix}^{\intercal} \tag{6.34}$$

$$\frac{\partial q_2}{\partial \omega} = \begin{bmatrix} \frac{\omega_y\omega_x}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \\ \frac{\Delta t}{2}\frac{{\omega_y}^2}{{\omega_{mod}}^2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) + \frac{1}{\omega_{mod}}\left(1 - \frac{{\omega_x}^2}{{\omega_{mod}}^2}\right)\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \\ \frac{\omega_y\omega_z}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \end{bmatrix}^{\intercal} \tag{6.35}$$

$$\frac{\partial q_3}{\partial \omega} = \begin{bmatrix} \frac{\omega_z \omega_x}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \\ \frac{\omega_z \omega_y}{{\omega_{mod}}^2}\left(\frac{\Delta t}{2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) - \frac{1}{\omega_{mod}}\sin\left(\omega_{mod}\frac{\Delta t}{2}\right)\right) \\ \frac{\Delta t}{2}\frac{{\omega_z}^2}{{\omega_{mod}}^2}\cos\left(\omega_{mod}\frac{\Delta t}{2}\right) + \frac{1}{\omega_{mod}}\left(1 - \frac{{\omega_x}^2}{{\omega_{mod}}^2}\right)\sin\left(\omega_{mod}\frac{\Delta t}{2}\right) \end{bmatrix}^{\intercal} \tag{6.36}$$

$$\omega_{mod} = \|\omega^W\| \tag{6.37}$$

Finally, $\partial f_v/\partial u$ (of size 13-by6) represents the Jacobian of f_v respect to input noise Q and can be formed using components defined above for $\partial f_v/\partial \hat{\mathrm{x}}_\mathrm{v}$.

$$\frac{\partial f_v}{\partial u} = \begin{bmatrix} \frac{\partial r}{\partial v} & 0 \\ 0 & \frac{\partial q}{\partial \omega} \\ I_{3\times 3} & 0 \\ 0 & I_{3\times 3} \end{bmatrix} \tag{6.38}$$

6.2.4 Features Definition

The complete state $\hat{\mathrm{x}}$ that includes the features $\hat{\mathrm{y}}$ consists of:

$$\hat{\mathrm{x}} = \left[{x_v}^{\intercal}, {\hat{y}_1}^{\intercal}, {\hat{y}_2}^{\intercal}, \ldots {\hat{y}_n}^{\intercal}\right] \tag{6.39}$$

where a feature $\hat{\mathrm{y}}_\mathrm{i}$ represents a point *i* in the 3D scene defined by the following 6-dimension state vector:

$$\hat{y}_i = [x_i \;\; y_i \;\; z_i \;\; \theta_i \;\; \phi_i \;\; \rho_i] \tag{6.40}$$

which models the 3D point located at:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) \tag{6.41}$$

where x_i,y_i,z_i are the optical center coordinates of the camera in which the feature was firstly observed, and are taken from r^{WC}= [x_v, y_v, z_v] in $\hat{\mathrm{x}}_v$; and θ_i, Φ_i represent azimuth and elevation (in relation to the world reference) for the directional unitary vector $m(\theta_i,\Phi_i)$. The point depth r_i along the ray is coded by its inverse (Figure 6.5 illustrates the camera and features parameterization):

$$\rho_i = \frac{1}{r_i} \tag{6.42}$$

The directional unitary vector $m(\theta_i,\Phi_i)$ can be determined by:

$$m = [\cos\phi \sin\theta \quad -\sin\phi \quad \cos\phi \cos\theta]^{\intercal} \tag{6.43}$$

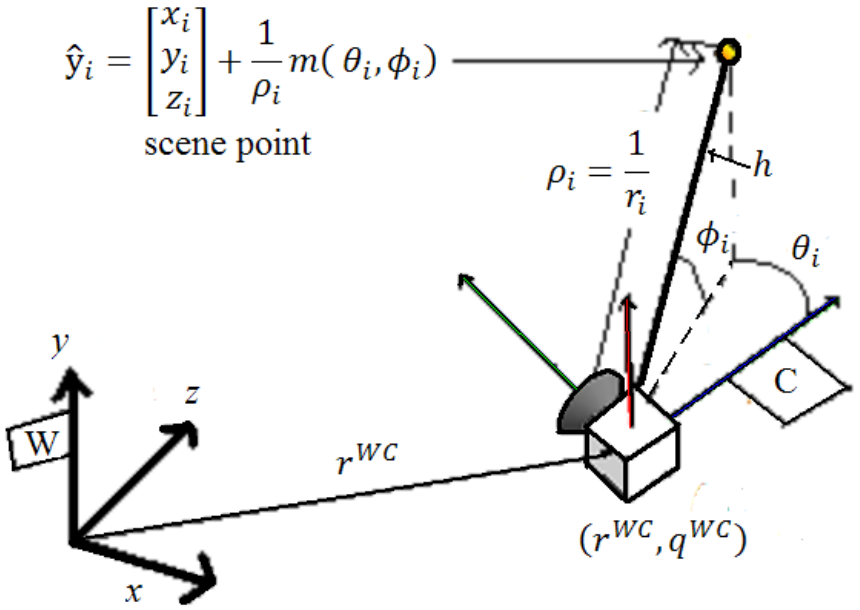


Figure 6.5, 6-DOF monocular SLAM camera and features parametrization.

6.2.5 Measurement Prediction Model.

The different locations of the camera, along with the location of the already mapped features, are used to predict the position of the feature in the image h_i.

The model observation of a point $\hat{y}_i$ from a camera location defines a ray expressed in the camera frame as:

$$h^c = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = R^{CW} \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) - r^{WC} \right) \tag{6.44}$$

h^C is observed by the camera through its projection in the image. R^{CW}, is the rotation matrix depending on the camera orientation quaternion q^{WC}. Equation (6.44) can also be rewritten as:

$$h^c = R^{CW} \left(\rho_i \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - r^{WC} \right) + m(\theta_i, \phi_i) \right) \tag{6.45}$$

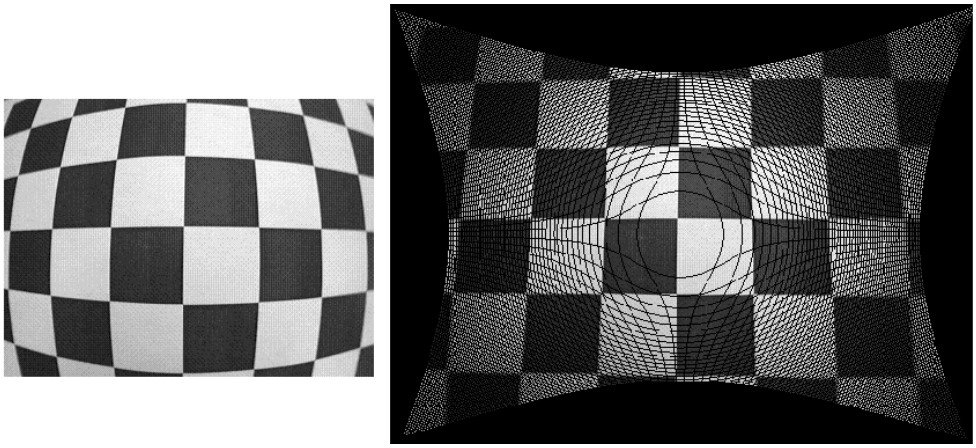


Figure 6.6 Example for the distortion-undistortion model used in this work. The wide-lens of the camera produces a radial distortion. The distortion is clearly appreciated on the picture taken to a board (left plot). The image is rectified (right plot) applying the undistortion model to the original image. Note, in the undistorted image, that the lines forming the squares, becomes straights.

R^{CW} Can be estimated from the Matrix transform $\mathrm{Q_R}$ which converts from a quaternion q to rotation matrix R.

$$R^{CW} = {Q_R}^{-1} \tag{6.46}$$

$$Q_R = \begin{bmatrix} ({q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2) & 2(q_1q_2 - q_0q_3) & 2(q_3q_1 + q_0q_3) \\ 2(q_1q_2 + q_0q_3) & ({q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_3) & 2(q_2q_3 + q_0q_1) & ({q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2) \end{bmatrix} \tag{6.47}$$

The projection is modeled using a full perspective wide angle camera. First the projection is modeled in the normalized retina

$$\begin{bmatrix} v \\ \nu \end{bmatrix} = \begin{bmatrix} \frac{h_x}{h_z} \\ \frac{h_y}{h_z} \end{bmatrix} \tag{6.48}$$

The camera calibration model is applied to produce the pixel coordinates for the predicted point:

$$h = \begin{bmatrix} u_u \\ v_u \end{bmatrix} = \begin{bmatrix} u_0 - \frac{f}{d_x} v \\ v_0 - \frac{f}{d_y} \nu \end{bmatrix} \tag{6.49}$$

where u_0,v_0 is the camera center in pixels, f is the focal length and d_x, and d_y the pixel size, and u_u, v_u the undistorted feature coordinates.

Finally, a radial distortion model is applied [110] for obtaining the distortion pixel coordinates u_d, v_d :

$$h_i = \begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} \frac{u_u - u_0}{\sqrt{1 + 2K_1 r^2}} + u_0 \\ \frac{v_u - v_0}{\sqrt{1 + 2K_1 r^2}} + v_0 \end{bmatrix} \tag{6.50}$$

Where K_1 is the distortion coefficient and:

$$r = \sqrt{(u_u - u_0)^2 + (v_u - v_0)^2} \tag{6.51}$$

Figure 6.6 illustrates the application of the radial distortion-undistortion model.

6.2.6 Active Search

Features search is constrained to regions around the predicted h_i. The regions are defined by the innovation covariance S_i:

$$S_i = H_i P_k H_i^{\mathrm{T}} + R \tag{6.52}$$

where H_i is the Jacobian of the sensor model with respect to the state, P_k is the prior state covariance, and measurements z assumed corrupted by zero mean Gaussian noise with covariance R.

$$R = \begin{bmatrix} {\sigma_u}^2 & 0 \\ 0 & {\sigma_v}^2 \end{bmatrix} \tag{6.53}$$

The values for the variances ${\sigma_u}^2$ and ${\sigma_v}^2$ are commonly set to 1 pixel. The matching process is described in the section 6.2.8.

6.2.7 Jacobians for the Feature Measurement Model

The Jacobian measurement model $\mathrm{H_i}$ includes both parts: the derivatives respect to vehicle state $\nabla \mathrm{H_{\hat{x}}}$ and the derivatives respect to the feature state $\nabla \mathrm{H_{\hat{y}i}}$:

$$\mathrm{H_i} = \begin{bmatrix} \underbrace{\nabla \mathrm{H_{\hat{x}}}}_{vehicle\ state} & \underbrace{\cdots 0 \cdots}_{other\ features} & \underbrace{\nabla \mathrm{H_{\hat{y}_i}}}_{observed\ feature} & \underbrace{\cdots 0 \cdots}_{other\ features} \end{bmatrix} \tag{6.54}$$

Note that $\nabla \mathrm{H_{yi}}$ it is only non-zero at the “location” (indexes) of the observed feature.

$\nabla \mathrm{H_{\hat{x}}}$ is determined by:

$$\nabla \mathrm{H}_{\hat{\mathrm{x}}} = \begin{bmatrix} \frac{\partial h}{r^{WC}} & \frac{\partial h}{\partial q^{WC}} & 0_{2\times 6} \end{bmatrix} \tag{6.55}$$

Being:

$$\frac{\partial h}{\partial r^{WC}} = \frac{\partial h}{\partial r_1}\frac{\partial r_1}{\partial r^{WC}} \tag{6.56}$$

$$\frac{\partial h}{\partial r_1} = \frac{\partial u_d}{\partial u}\frac{\partial u}{\partial r_1} \tag{6.57}$$

$$\frac{\partial h}{\partial q^{WC}} = \frac{\partial h}{\partial r_1}\frac{\partial r_1}{\partial q^{WC}} \tag{6.58}$$

$$\frac{\partial r_1}{\partial q^{WC}} = \frac{\partial R_q}{\partial q_t}\frac{\partial q_t}{\partial q} \tag{6.59}$$

The Jacobian of the distortion model (6.50) $\partial u_d/\partial u$ is defined by:

$$\frac{\partial u_d}{\partial u} = \begin{bmatrix} \frac{-2(u_u-u_0)^2K_1}{(1+2K_1r^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+2K_1r^2}} & \frac{-2(u_u-u_0)(v_u-v_0)K_1}{(1+2K_1r^2)^{\frac{3}{2}}} \\ \frac{-2(u_u-u_0)(v_u-v_0)K_1}{(1+2K_1r^2)^{\frac{3}{2}}} & \frac{-2(v_u-v_0)^2K_1}{(1+2K_1r^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+2K_1r^2}} \end{bmatrix} \tag{6.60}$$

The rest of components are:

$$\frac{\partial u}{\partial r_1} = \begin{bmatrix} \frac{-f}{d_x h_z} & 0 & \frac{h_x f}{d_x {h_z}^2} \\ 0 & \frac{-f}{d_y h_z} & \frac{h_v f}{d_v {h_z}^2} \end{bmatrix} \tag{6.61}$$

$$\frac{\partial r_1}{\partial r^{WC}} = {Q_R}^{-1}\rho_i \tag{6.62}$$

$$\frac{\partial q_t}{\partial q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{6.63}$$

And:

$$\frac{\partial R_q}{\partial q_t} = \begin{bmatrix} \frac{\partial R_q}{\partial q_0} & \frac{\partial R_q}{\partial q_1} & \frac{\partial R_q}{\partial q_2} & \frac{\partial R_q}{\partial q_3} \end{bmatrix} \tag{6.64}$$

The next four equations are estimated using the conjugate of the quaternion q^{WC}: $q = [q_0\ q_1\ q_2\ q_3] = conj(q^{WC})$.

$$\frac{\partial R_q}{\partial q_0} = \begin{bmatrix} 2q_0 & -2q_3 & -2q_2 \\ 2q_3 & 2q_0 & -2q_1 \\ -2q_2 & 2q_1 & 2q_0 \end{bmatrix} C \tag{6.65}$$

$$\frac{\partial R_q}{\partial q_1} = \begin{bmatrix} 2q_1 & 2q_2 & 2q_3 \\ 2q_2 & -2q_1 & -2q_0 \\ 2q_3 & 2q_0 & -2q_1 \end{bmatrix} C \tag{6.66}$$

$$\frac{\partial R_q}{\partial q_2} = \begin{bmatrix} -2q_2 & 2q_1 & 2q_0 \\ 2q_1 & 2q_2 & 2q_3 \\ -2q_0 & 2q_3 & -2q_2 \end{bmatrix} C \tag{6.67}$$

$$\frac{\partial R_q}{\partial q_3} = \begin{bmatrix} -2q_3 & -2q_0 & 2q_1 \\ 2q_0 & -2q_3 & 2q_2 \\ 2q_1 & 2q_2 & 2q_3 \end{bmatrix} C \tag{6.68}$$

where

$$C = \rho_i \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - r^{WC} \right) + m(\theta_i, \phi_i) \tag{6.69}$$

Finally $\nabla \mathrm{H}_{\hat{\mathrm{y}}\mathrm{i}}$ can be formed using components defined in $\nabla \mathrm{H}_{\hat{\mathrm{x}}}$:

$$\nabla \mathrm{H}_{\mathrm{x}} = \frac{\partial h}{\partial r_1} \frac{\partial r_1}{\partial y} \tag{6.70}$$

and:

$$\frac{\partial r_1}{\partial y} = \left[R^{CW} \rho_i \quad R^{CW} \begin{bmatrix} cos\,\phi\, cos\,\theta \\ 0 \\ -cos\,\phi\, sin\,\theta \end{bmatrix} \quad R^{CW} \begin{bmatrix} -\sin\phi \sin\theta \\ \cos\phi \\ -\sin\phi \cos\theta \end{bmatrix} \quad R^{CW} \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - r^{WC} \right) \right] \tag{6.71}$$

6.2.8 Matching and Updating

When the innovation covariance matrix S_i is estimated in (6.52), the feature search can be constrained to regions around each predicted feature h_i.

The size of the axis s_x,s_y of the search region (Illustrated in Figure 6.7) are determined as follows:

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} 2n\sqrt{S_{i(1,1)}} \\ 2n\sqrt{S_{i(2,2)}} \end{bmatrix} \tag{6.72}$$

Where n is the number of standard deviations of the desired region search.

When each feature is initialized in the map (described in subsequent sections) a unique image patch of *n-by-n* pixel is stored and related with the feature. For matching a feature in the current image frame, patch cross-correlation techniques (described in

Chapter 5), are applied in all the image locations determined by the region search. If the score of a pixel location (u_d,v_d) determined by the best cross-correlation between the feature patch and the patches defined by the region of search, are higher than a threshold, then this pixel location (u_d,v_d) is considered as the current feature measurement z_i.

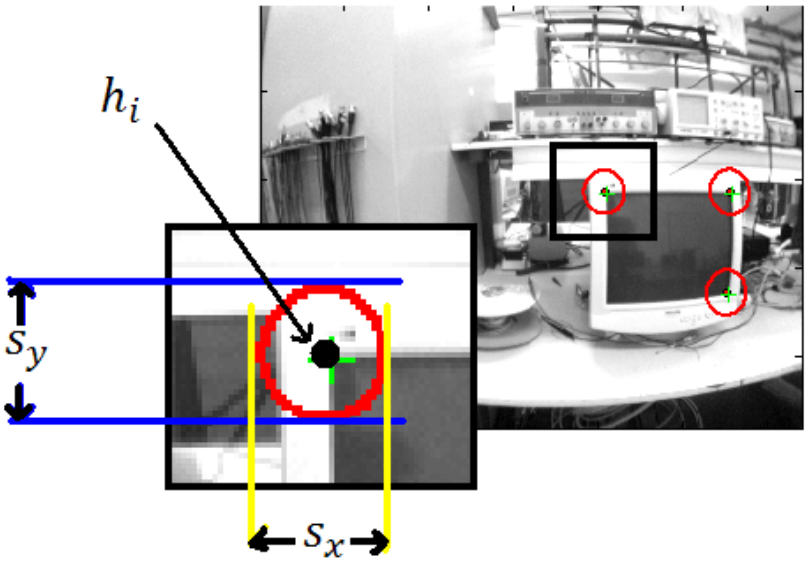


Figure 6.7 Active Search is used for maximize the computational speed and reduce the chance of mismatches in the data association process. Features search is constrained to regions around the predicted h_i. The regions are defined by the innovation covariance S_i.

If the matching process is successful, the Kalman Filter is updated as follows:

$$\hat{\mathrm{x}}_{\mathrm{k}} = \hat{\mathrm{x}}_{\mathrm{k+1}} + Wg \tag{6.73}$$

$$P_k = P_{k+1} - WS_iW^{\mathrm{T}} \tag{6.74}$$

where the innovation is

$$g = z_i - h_i \tag{6.75}$$

And W, is the Kalman gain:

$$W = P_{k+1}H_i^{\mathrm{T}}S^{-1} \tag{6.76}$$

6.2.9 Delayed Inverse Depth Feature Initialization

In Chapter 4 we present a novel approach, called *Delayed Inverse Depth Feature Initialization*, for initializing new features in Bearing-Only SLAM systems. Theoretical and practical justification for the method was also presented, along with several experiments with simulated and real data (SSLAM) in order to illustrate the performance of the method. In this section our method for initializing new features is extended for working in a full monocular SLAM context.

Also it was seen in Chapter 4, that in the *unified inverse depth method* presented by Montiel in [1], transition from partially to fully initialized features need not to be explicitly tackled, making it suitable for direct use in EKF-SLAM framework for sparse mapping. In this approach the features are initialized in the first frame observed (un-delayed initialization) with an initial fixed depth and uncertainty, determined heuristically to cover ranges from nearby to infinity, so distant points can be coded. Due to the clarity and scalability of this method, this approach is a good option for monocular-SLAM implementation.

This work is motivated by the problems of robot map building and localization, therefore, if monocular SLAM wants to be applied in this context, retrieving the metric scale of the world is very essential. In that sense, the experiments with monocular SLAM and un-delayed initialization (presented in section 6.2.16) show that, when initial reference points are used for establishing a metric scale in the map, the initial features depths have to be tuned, otherwise, is likely that new features added to the map never converges respect to the metric reference. Also initializing features distant to the robot center can increase the possibility that features depth become negative after a Kalman update step. This behavior may cause divergence in the filter. In experiments the same effects observed with the un-delayed initialization method for 2D Bearing-Only SLAM systems are also observed in the monocular SLAM case.

Moreover, initializing features in the first observed frame (un-delayed initialization) avoids the use of pre-initialized features in the state and allows the use of all the information available in the feature since it is detected. Nevertheless, when features are detected in the image with a saliency operator in order to be automatically added to the map, usually weak long-term image features are added to the map. Therefore it is difficult to match them in subsequent frames. When a minimum number of active image features want to be maintained, it could happen that unnecessary initializations are realized. Every new feature initialization introduces biases to the system [89].

The aforementioned issues can suggest, for the new features, that initial inverse depth and its uncertainty could be estimated dynamically before being added to the system state, instead of using a fixed initial depth and uncertainty. At the same time features can be tested prior to be added to map in order to prune weak long-term features.

6.2.10 Candidate Points

In our approach we consider a minimum number of features $\hat{y}_i$ to be predicted appearing in the image, otherwise new features have to be added to the map. In this latter case, new points are detected in the image with a saliency operator. Specifically, we use the Harris corner detector, although others detectors can be used. Only areas in the image free of previously detected points or features already mapped are considered for detecting new points, we call these points in the image, which does not have to be added yet to the map, as candidate points, λ.

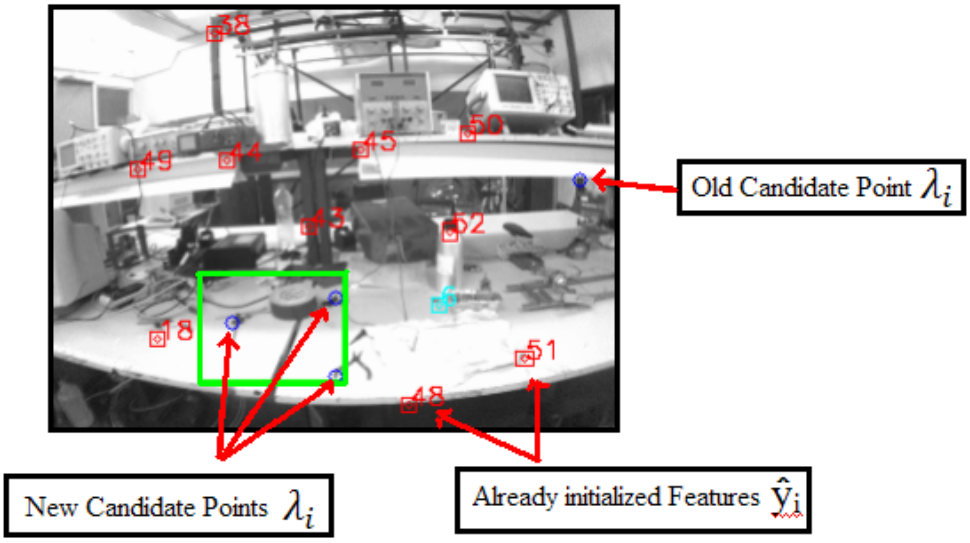


Figure 6.8 New features are tried to be added to the map. If the number of predicted features $\hat{y}_i$ to appear in the image, are lower than a threshold. First, random areas free-of-features are detected in the image (green rectangle). Then a saliency operator is applied to mentioned areas in order to detect new candidate points λ_i.

When a point is first detected by the saliency operator in a frame k, the candidate point is conformed by:

$$\lambda_i = \left(r_\lambda, \sigma_{r_\lambda}, q_\lambda, \sigma_{q_\lambda}, u_\lambda, v_\lambda\right) \tag{6.77}$$

Where:

$$r_\lambda = (x_\lambda, y_\lambda, z_\lambda) \tag{6.78}$$

The values x_λ, y_λ, z_λ taken from r^{WC}= [x_v, y_v, z_v] in $\hat{x}_v$, represents the current camera optical center position.

$$\sigma_{r_\lambda} = \left(\sigma_{x_\lambda}, \sigma_{y_\lambda}, \sigma_{z_\lambda}\right) \tag{6.79}$$

$\sigma_{r\lambda}$ represents the associated variances of x_λ, y_λ, z_λ taken from the state covariance matrix P_k.

$$q_\lambda = \left(q_{0_\lambda}, q_{1_\lambda}, q_{2_\lambda}, q_3\ \right) \tag{6.80}$$

Is the quaternion taken from $\hat{x}_v$, representing the current camera orientation and its associated variances $\sigma_{q\lambda}$ taken from the state covariance matrix P_k,

$$\sigma_{q_\lambda} = \left(\sigma_{q_{0_\lambda}}, \sigma_{q_{1_\lambda}}, \sigma_{q_{2_\lambda}}, \sigma_{q_{3_\lambda}}\right) \tag{6.81}$$

Finally u_λ, v_λ is the current pixel coordinates for the point λ_i.

In subsequent frames, λ_i is tracked. But in practice some λ_i points cannot be tracked. This implicit process is used for pruning weakest image features. For tracking purposes any method can be used. The tracking for every candidate point λ_i is realized until it is pruned or initialized in the system. In practice for every frame, some new candidate points λ_i could be detected, others candidate points could be pruned and others could be considered to be added to the map. In our experiments an average of 5 to 15 points λ_i are maintained at every step.

6.2.11 Conditions for adding features to the state

As the camera freely moves through the environment, the translation produces parallax in features. Parallax is really the key that allows to estimating features depth. In the case of indoor sequences, centimeters are enough to produce parallax, on the other hand, the more distant the features, the more the camera have to be translated to produce parallax.

In our approach we want dynamically to estimate an initial depth and its associated uncertainty for the features added to the map. For near features, a small translation is enough to reproduce some parallax. We use a minimum parallax threshold $\alpha_{\min}$ for considering a candidate point λ_i to be added to the map as a feature $\hat{y}_i$. On the other hand, as was mentioned before, distant features will not produce parallax but are useful to estimate the camera orientation. Therefore it is advantageous to include some distant features in the map (with big depth uncertainty). A minimum base-line camera translation $|b|_{min}$ is also considered for adding a candidate point $\hat{y}_i$ to the map. In Chapter 4 (Figure 4.11) a simulation is shown, illustrating the decrementing uncertainty in feature depth estimation respect with the increase of parallax angle. It can be observed that a few parallax degrees are enough for reducing significantly the depth uncertainty.

A good attribute of the cameras is its bearing precision. The camera´s bearing precision makes possible to use a lower threshold $\alpha_{\min}$, compared with the $\alpha_{\min}$ used with the less accurate sound sensor (chapter 4). In experiments we use a threshold $\alpha_{\min}= 3^o$.

The minimum base-line b_{min} was heuristically established to be the base-line necessary to produce a parallax $\alpha \approx 6^{o}$ in the initial reference points. For example if the camera initial position is in average one meter away from the initial reference points then $b_{min} = 8cm$.

6.2.12 Estimating Parallax

If a candidate point λ_i shows a minimum parallax α_{min} then it will initialized as a new feature $\hat{y}_i$.

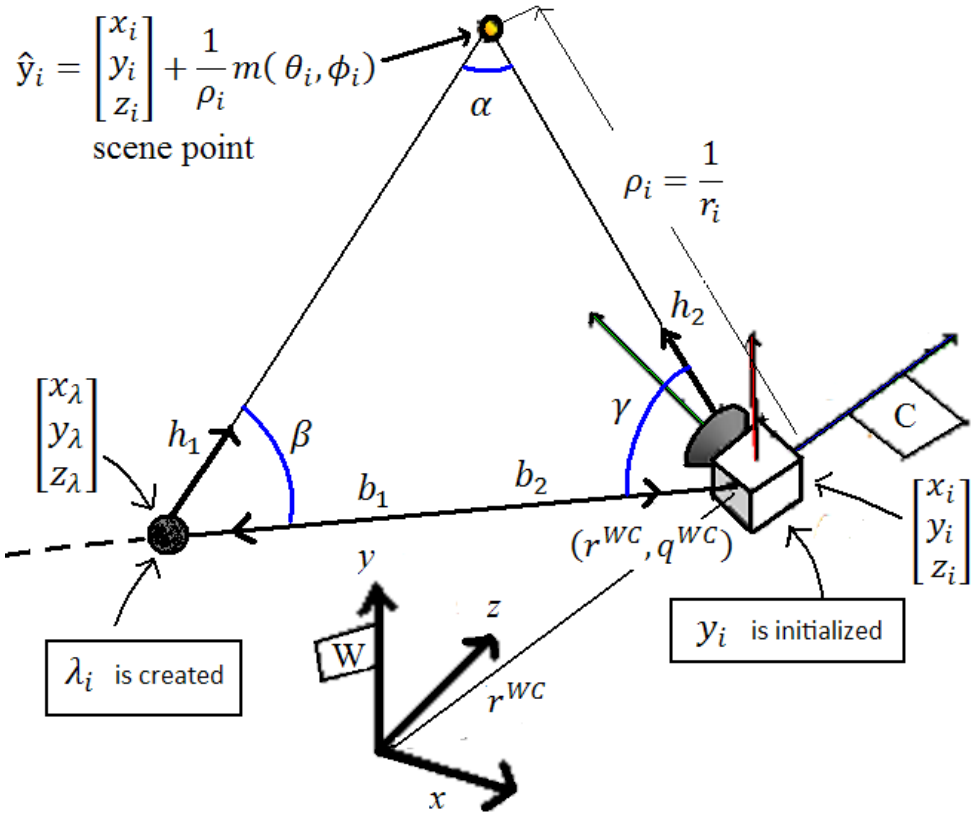


Figure 6.9 Schema for the features initialization process.

At this stage, the uncertainty of the measurements is not considered , and the parallax α is approximated using (i) the base line b, (ii) λ_i using its associated data and (iii) the current state and measurement ($\hat{x}_v$,z_i).

For each candidate point λ_i, every frame a new measurement z_i is available, then the parallax angle α can be estimated by (Figure 6.9):

$$\alpha = \pi - (\beta + \gamma) \tag{6.82}$$

The angle β is determined by the directional unitary vector h_1 and the vector b_1 defining the base-line b in the direction of the camera trajectory by:

$$\beta = cos^{-1}\left(\frac{h_1 \cdot b_1}{\|h_1\|\|b_1\|}\right) \tag{6.83}$$

Where the directional projection ray vector $h_1 = [h_{1x} \;\; h_{1y} \; h_{1z}]$ expressed in the absolute frame ^{W}, is computed from the camera position and the coordinates of the observed point when it was first observed, using the data stored in λ_i.

$$h_1 = R_{WC}(q_\lambda) h_1^C \begin{pmatrix} u_{\lambda u} \\ v_{\lambda u} \end{pmatrix} \tag{6.84}$$

With $R_{WC}(q_\lambda)$ being the rotation matrix depending on the stored camera orientation quaternion q_λ. (Equation (6.80)). R_{WC} can be determined by the Matrix transform Q_R (Equation (6.47)) which converts from a quaternion q to rotation matrix R.

$$R^{CW} = Q_R \tag{6.85}$$

The undistorted coordinates pixel $u_{\lambda u}$ and $v_{\lambda u}$ are obtained from u_λ,v_λ applying the radial undistortion model:

$$\begin{bmatrix} u_{\lambda u} \\ v_{\lambda u} \end{bmatrix} = \begin{bmatrix} \frac{u_\lambda - u_0}{\sqrt{1 + 2K_1 r_d{}^2}} + u_0 \\ \frac{v_\lambda - v_0}{\sqrt{1 + 2K_1 r_d{}^2}} + v_0 \end{bmatrix} \tag{6.86}$$

Where K_1 is the distortion coefficient and:

$$r_d = \sqrt{(u_\lambda - u_0)^2 + (v_\lambda - v_0)^2} \tag{6.87}$$

And $h_1{}^C(u_{\lambda u}, v_{\lambda u}) = [\, h_{1x}{}^C \;\; h_{1y}{}^C \; h_{1z}{}^C]$ is the directional vector in the camera frame ^{C}, when the candidate point λ_i was first observed, and can be estimated from:

$$h_1^C = \begin{bmatrix} \frac{(u_0 - u_{\lambda u}) d_x}{f} & \frac{(v_0 - v_{\lambda u}) d_y}{f} & 1 \end{bmatrix} \tag{6.88}$$

Where u_0,v_0 is the camera center in pixels, f is the focal length and d_x, and d_y the pixel size.

$b_1 = [\, b_{1x} \;\; b_{1y} \; b_{1z}]$ is the vector representing the camera base-line b between the camera optical center position x_λ, y_λ, z_λ where the point was first observed and the current optical center $r^{WC} = [\, x_v, y_v, z_v]$ taken from the current camera state x_v.

$$b_1 = [(x - x_\lambda), (y - y_\lambda), (z - z_\lambda)] \tag{6.89}$$

The angle γ is determined in a similar way as β but using the directional projection ray vector h_2 and the vector b_2 defining the base-line in the opposite direction of the camera trajectory by:

$$\gamma = cos^{-1}\left(\frac{h_2 \cdot b_2}{\|h_2\|\|b_2\|}\right) \tag{6.90}$$

The directional projection ray vector h_2 expressed in the absolute frame $^{\mathrm{W}}$, is computed in a similar way as (6.83) but using current camera position $\hat{\mathrm{x}}_v$ and current undistorted points coordinates u_u,v_u.

$$h_2 = R_{WC}(q^{WC}) h_2^C \begin{pmatrix} u_u \\ v_u \end{pmatrix} \tag{6.91}$$

The undistorted points coordinates u_u,v_u, are obtained applying again the undistorted model to the current (not the stored) pixel coordinates u,v of the candidate point λ_i.:

$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \begin{bmatrix} \dfrac{u - u_0}{\sqrt{1 + 2K_1 {r_d}^2}} + u_0 \\ \dfrac{v - v_0}{\sqrt{1 + 2K_1 {r_d}^2}} + v_0 \end{bmatrix} \tag{6.92}$$

$$r_d = \sqrt{(u - u_0)^2 + (v - v_0)^2} \tag{6.93}$$

And $h_2^C(u_u,v_u) = [\, {h_{2x}}^C \;\; {h_{2y}}^C \; {h_{2z}}^C]$ is the directional vector in the camera frame $^{\mathrm{C}}$ from the current camera position, and can be estimated from:

$$h_2^C = \begin{bmatrix} \dfrac{(u_0 - u_u) d_x}{f} & \dfrac{(v_0 - v_u) d_y}{f} & 1 \end{bmatrix} \tag{6.94}$$

$b_2 = [\, b_{2x} \;\; b_{2y} \; b_{2z}]$ is equal to b_1 but pointing to the opposite direction:

$$b_2 = [(x_\lambda - x), (y_\lambda - y), (z_\lambda - z)] \tag{6.95}$$

The base-line b is the module of b_2 or b_1:

$$b = \|b_1\| = \|b_2\| \tag{6.96}$$

6.2.13 Feature Initialization in state and covariance matrix

If $\alpha > \alpha_{min}$ then λ_i is initialized as a new feature map $\hat{y}_i$. In this case the new feature $\hat{y}_{new}$ is determined by:

$$\hat{y}_{new} = \left[\hat{x}_i \;\; \hat{y}_i \;\; \hat{z}_i \;\; \hat{\theta}_i \;\; \hat{\phi}_i \;\; \hat{\rho}_i\right]^T \tag{6.97}$$

The three first elements of $\hat{y}_{new}$ are obtained directly from the current camera optical center position $r^{WC}= [\, x_v, y_v, z_v]$ taken from $\hat{x}_v$.

$$\begin{bmatrix}\hat{x}_i \\ \hat{y}_i \\ \hat{z}_i\end{bmatrix} = \begin{bmatrix}x_v \\ y_v \\ z_v\end{bmatrix} \tag{6.98}$$

The angles can be derived by:

$$\begin{bmatrix}\hat{\theta}_i \\ \hat{\phi}_i\end{bmatrix} = \begin{bmatrix}\text{atan2}\left(-h_{2y}, \sqrt{{h_{2x}}^2 + {h_{2z}}^2}\right) \\ \text{atan2}(h_{2x}, h_{2z})\end{bmatrix} \tag{6.99}$$

where

$$h_2 = \begin{bmatrix}h_{2x} \\ h_{2y} \\ h_{2z}\end{bmatrix} \tag{6.100}$$

is obtained from Equation (6.91). atan2 is a two-argument function that computes the arctangent of y/x given y and x, within a range of $[-\pi, \pi]$.

Finally the inverse depth $\hat{\rho}_i$ is derived from the sine law:

$$\hat{\rho}_i = \frac{\sin\alpha}{b * \sin\beta} \tag{6.101}$$

where α, b and β are estimated as was shown in the previous section.

The new system state $\hat{x}$ is conformed simply adding the new feature $\hat{y}_{new}$ to the final of the vector state:

$$\hat{x} = \begin{bmatrix}x_v \\ y_1 \\ y_2\end{bmatrix} \quad \hat{x}_{new} = \begin{bmatrix}x_v \\ y_1 \\ y_2 \\ \hat{y}_{new}\end{bmatrix} \tag{6.102}$$

The covariance for the new feature $\hat{y}_{new}$ is derived from the error diagonal covariance matrix measurement R_j and the current state covariance matrix P.

$$R_j = \begin{bmatrix} {\sigma_{u2}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & {\sigma_{v2}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & {\sigma_{u1}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & {\sigma_{v1}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x_\lambda} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{y_\lambda} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{z_\lambda} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{q_{0_\lambda}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{q_{1_\lambda}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{q_{2_\lambda}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{q_{3_\lambda}} \end{bmatrix} \tag{6.103}$$

Note that R_j is now conformed by the image measurement error variance σ_{u2}, σ_{v2}, σ_{u1}, σ_{v1} and the variances stored in λ_i related to $\sigma_{r\lambda}$ and $\sigma_{q\lambda}$.

Depending on the algorithms used for extract features in the image, is possible to obtain even sub-pixel precision. Although, in this work, we considering a error variance $\sigma_{u2} = \sigma_{v2} = \sigma_{u1} = \sigma_{v1} = 1$ pixel.

The new state covariance matrix, after initialization, is:

$$P_{new} = \nabla Y \begin{pmatrix} P_k & 0 \\ 0 & R_j \end{pmatrix} \nabla Y^T \tag{6.104}$$

6.2.14 Jacobian ∇Y

The Jacobian ∇Y for the initialization process is:

$$\nabla Y = \left(\frac{\partial \hat{y}}{\partial \hat{x}_v}, 0, \ldots, 0, \begin{matrix} I \\ \\ \end{matrix} \begin{matrix} 0 \\ \frac{\partial \hat{y}}{\partial R_j} \end{matrix} \right) \tag{6.105}$$

Where I is the identity matrix with the same dimension of P_k, $\partial\hat{y}/\partial x_v$ are the derivatives of the initializations equations with respect to the camera state $\hat{x}_v$ and $\partial\hat{y}/\partial R_j$ the derivatives respect to the parameters of the covariance matrix R_j.

Most part of the equations needed for estimating the new feature $\hat{y}_{new}$ are related with the estimation of the initial feature´s inverse-depth $\hat{\rho}_i$. For clarity we isolate the derivatives related to the first five parameters of $\hat{y}$, and the derivatives related to $\hat{\rho}$. Thus $\partial\hat{y}/\partial x_v$ and $\partial\hat{y}/\partial R_j$ are defined by:

$$\frac{\partial \hat{y}}{\partial \hat{x}_v} = \begin{bmatrix} \frac{\partial \hat{y}_o}{\partial \hat{x}_v} \\ \frac{\partial \rho}{\partial \hat{x}_v} \end{bmatrix} \tag{6.106}$$

and

$$\frac{\partial \hat{y}}{\partial R_j} = \begin{bmatrix} \frac{\partial \hat{y}_o}{\partial R_j} \\ \frac{\partial \rho}{\partial R_j} \end{bmatrix} \tag{6.107}$$

First, we start defining $\partial \hat{y}_o / \partial x_v$ as follows:

$$\frac{\partial \hat{y}_o}{\partial \hat{x}_v} = \begin{bmatrix} \frac{\partial \hat{y}_o}{\partial r^{WC}} & \frac{\partial \hat{y}_o}{\partial q^{WC}} & 0_{5x6} \end{bmatrix} \tag{6.108}$$

$$\frac{\partial \hat{y}_o}{\partial r^{WC}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.109}$$

$$\frac{\partial \hat{y}_o}{\partial q^{WC}} = \begin{bmatrix} 0_{3x4} \\ \frac{\partial \theta}{\partial q^{WC}} \\ \frac{\partial \phi}{\partial q^{WC}} \end{bmatrix} \tag{6.110}$$

where:

$$\frac{\partial \theta}{\partial q^{WC}} = \begin{bmatrix} \frac{\partial \theta}{\partial R_q} \frac{\partial R_q}{\partial q_t} \end{bmatrix} \tag{6.111}$$

$$\frac{\partial \phi}{\partial q^{WC}} = \begin{bmatrix} \frac{\partial \phi}{\partial R_q} \frac{\partial R_q}{\partial q_t} \end{bmatrix} \tag{6.112}$$

$$\frac{\partial \theta}{\partial R_q} = \begin{bmatrix} \frac{h_{2z}}{(h_{2x})^2 + (h_{2z})^2} & 0 & \frac{h_{2x}}{(h_{2x})^2 + (h_{2z})^2} \end{bmatrix} \tag{6.113}$$

$$\frac{\partial \phi}{\partial R_q} = \begin{bmatrix} \frac{(h_{2x})(h_{2y})}{\left((h_{2x})^2 + (h_{2y})^2 + (h_{2z})^2\right)\sqrt{(h_{2x})^2 + (h_{2z})^2}} \\ \frac{-\sqrt{(h_{2x})^2 + (h_{2z})^2}}{\left((h_{2x})^2 + (h_{2y})^2 + (h_{2z})^2\right)} \\ \frac{(h_{2z})(h_{2y})}{\left((h_{2x})^2 + (h_{2y})^2 + (h_{2z})^2\right)\sqrt{(h_{2x})^2 + (h_{2z})^2}} \end{bmatrix}^T \tag{6.114}$$

The component $\partial R_q / \partial q_t$ is obtained in the same way than equation (6.64) but using the current q^{WC}: $q = [q_0\ q_1\ q_2\ q_3]$ taken from the camera state $\hat{x}_v$. The vector C, used in the estimation of equation (6.64), is equal to the vector h_2, estimated in equation (6.91).

The second component of equation (6.106); $\partial \rho / \partial x_v$ is formed as follows:

$$\frac{\partial \rho}{\partial \hat{x}_v} = \begin{bmatrix} \frac{\partial \rho}{\partial r^{WC}} & \frac{\partial \rho}{\partial q^{WC}} & 0_{1x6} \end{bmatrix} \tag{6.115}$$

Where $\partial\rho/\partial r^{WC}$ is defined by:

$$\frac{\partial \rho}{\partial r^{WC}} = \begin{bmatrix} \frac{\partial \rho_\gamma}{\partial r^{WC}} + \frac{\partial \rho_\beta}{\partial r^{WC}} + \frac{\partial \rho_{b_2}}{\partial r^{WC}} \end{bmatrix} \tag{6.116}$$

Being:

$$\frac{\partial \rho_\gamma}{\partial r^{WC}} = \begin{bmatrix} \frac{\partial \rho_\gamma}{\partial \rho_{\gamma o}} \frac{\partial \rho_{\gamma o}}{\partial \rho_{\gamma c}} \frac{\partial \rho_{\gamma c}}{\partial \rho_{\gamma b}} \frac{\partial \rho_{\gamma b}}{\partial r^{WC}} \end{bmatrix} \tag{6.117}$$

$$\frac{\partial \rho_\gamma}{\partial \rho_{\gamma o}} = \frac{\cos(\beta + \gamma)}{\mathrm{b} \sin \beta} \tag{6.118}$$

$$\frac{\partial \rho_{\gamma o}}{\partial \rho_{\gamma c}} = \frac{-1}{\left(\sqrt{1 - (\gamma_{cos})^2}\right)} \tag{6.119}$$

$$\gamma_{cos} = \left(\frac{h_2 \cdot b_2}{\|h_2\| \|b_2\|} \right) \tag{6.120}$$

$$\frac{\partial \rho_{\gamma c}}{\partial \rho_{\gamma b}} = \begin{bmatrix} \frac{h_{2x}}{\mathrm{a}^{\left(\frac{1}{2}\right)} \mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g}) b_{2x}}{\mathrm{s}^{\left(\frac{1}{2}\right)} \mathrm{a}^{\left(\frac{3}{2}\right)}} \\ \frac{h_{2y}}{\mathrm{a}^{\left(\frac{1}{2}\right)} \mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g}) b_{2y}}{\mathrm{s}^{\left(\frac{1}{2}\right)} \mathrm{a}^{\left(\frac{3}{2}\right)}} \\ \frac{h_{2z}}{\mathrm{a}^{\left(\frac{1}{2}\right)} \mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g}) b_{2z}}{\mathrm{s}^{\left(\frac{1}{2}\right)} \mathrm{a}^{\left(\frac{3}{2}\right)}} \end{bmatrix}^{\mathrm{T}} \tag{6.121}$$

$$\begin{gathered} \mathrm{a} = (b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2 \\ \mathrm{s} = (h_{2x})^2 + \left(h_{2y}\right)^2 + (h_{2z})^2 \\ \mathrm{g} = (h_{2x})(b_{2x}) + \left(h_{2y}\right)\left(b_{2y}\right) + (h_{2z})(b_{2z}) \end{gathered} \tag{6.122}$$

$$\frac{\partial \rho_{\gamma b}}{\partial r^{WC}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{6.123}$$

And:

$$\frac{\partial \rho_\beta}{\partial r^{WC}} = \begin{bmatrix} \frac{\partial \rho_\beta}{\partial \rho_{\beta o}} \frac{\partial \rho_{\beta o}}{\partial \rho_{\beta c}} \frac{\partial \rho_{\beta c}}{\partial \rho_{\beta b}} \frac{\partial \rho_{\beta b}}{\partial r^{WC}} \end{bmatrix} \tag{6.124}$$

$$\frac{\partial \rho_{\beta}}{\partial \rho_{\beta o}} = \frac{\cos(\beta + \gamma)}{\mathrm{b}\sin\beta} - \frac{\sin(\beta + \gamma)\cos\beta}{\mathrm{b}\,(\sin\beta)^2} \tag{6.125}$$

$$\frac{\partial \rho_{\beta o}}{\partial \rho_{\beta c}} = \frac{-1}{\left(\sqrt{1 - (\beta_{cos})^2}\right)} \tag{6.126}$$

$$\beta_{cos} = \left(\frac{h_1 \cdot b_1}{\|h_1\|\|b_1\|}\right) \tag{6.127}$$

$$\frac{\partial \rho_{\beta c}}{\partial \rho_{\beta b}} = \begin{bmatrix} \dfrac{h_{1x}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \dfrac{(\mathrm{g})b_{1x}}{\mathrm{s}^{\left(\frac{1}{2}\right)}\mathrm{a}^{\left(\frac{3}{2}\right)}} \\ \dfrac{h_{1y}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \dfrac{(\mathrm{g})b_{1y}}{\mathrm{s}^{\left(\frac{1}{2}\right)}\mathrm{a}^{\left(\frac{3}{2}\right)}} \\ \dfrac{h_{1z}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \dfrac{(\mathrm{g})b_{1z}}{\mathrm{s}^{\left(\frac{1}{2}\right)}\mathrm{a}^{\left(\frac{3}{2}\right)}} \end{bmatrix}^{\mathrm{T}} \tag{6.128}$$

$$\begin{aligned} \mathrm{a} &= (b_{1x})^2 + \left(b_{1y}\right)^2 + (b_{1z})^2 \\ \mathrm{s} &= (h_{1x})^2 + \left(h_{1y}\right)^2 + (h_{1z})^2 \\ \mathrm{g} &= (h_{1x})(b_{1x}) + \left(h_{1y}\right)\left(b_{1y}\right) + (h_{1z})(b_{1z}) \end{aligned} \tag{6.129}$$

The next component is an identity matrix and is showed only for clarity:

$$\frac{\partial \rho_{\beta b}}{\partial r^{WC}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.130}$$

And:

$$\frac{\partial \rho_{b_2}}{\partial r^{WC}} = \left[\frac{\partial \rho_{b_2}}{\partial \rho_{\beta o}}\frac{\partial \rho_{\beta o}}{\partial r^{WC}}\right] \tag{6.131}$$

$$\frac{\partial \rho_{b_2}}{\partial \rho_{\beta o}} = \frac{\sin(\beta + \gamma)}{\mathrm{b}^2\sin\beta} \tag{6.132}$$

$$\frac{\partial \rho_{\beta o}}{\partial r^{WC}} = \begin{bmatrix} \dfrac{-b_{2x}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \\ \dfrac{-b_{2y}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \\ \dfrac{-b_{2z}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \end{bmatrix}^{\mathrm{T}} \tag{6.133}$$

The last component of equation (6.115); $\partial\rho/\partial q^{WC}$ is defined by:

$$\frac{\partial \rho}{\partial q^{WC}} = \left[\frac{\partial \rho_\gamma}{\partial \rho_{\gamma o}} \frac{\partial \rho_{\gamma o}}{\partial \rho_{\gamma c}} \frac{\partial \rho_{\gamma c}}{\partial \rho_{h_2}} \frac{\partial \rho_{h_2}}{\partial q^{WC}}\right] \tag{6.134}$$

The two first elements of $\partial\rho/\partial q^{WC}$ are estimated from equations (6.118) and (6.119). The rest of elements are:

$$\frac{\partial \rho_{\gamma c}}{\partial \rho_{h_2}} = \begin{bmatrix} \frac{b_{2x}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g})h_{2x}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{3}{2}\right)}} \\ \frac{b_{2y}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g})h_{2y}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{3}{2}\right)}} \\ \frac{b_{2z}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{1}{2}\right)}} - \frac{(\mathrm{g})h_{2z}}{\mathrm{a}^{\left(\frac{1}{2}\right)}\mathrm{s}^{\left(\frac{3}{2}\right)}} \end{bmatrix}^{\mathrm{T}} \tag{6.135}$$

$$\begin{aligned} \mathrm{a} &= (b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2 \\ \mathrm{s} &= (h_{2x})^2 + \left(h_{2y}\right)^2 + (h_{2z})^2 \\ \mathrm{g} &= (h_{2x})(b_{2x}) + \left(h_{2y}\right)\left(b_{2y}\right) + (h_{2z})(b_{2z}) \end{aligned} \tag{6.136}$$

The component $\partial\rho_{h2}/\partial q^{WC}$ is obtained in the same manner than equation (6.64) using the current q^{WC}: $q = [q_0\ q_1\ q_2\ q_3]$ taken from the camera state $\hat{\mathrm{x}}_{\mathrm{v}}$. The vector C, (used in the estimation of equation (6.64)), is the same than the vector h_2, (estimated in equation (6.91)).

Now, we continue with the expansion of $\partial\hat{y}/\partial R_{\mathrm{j}}$ which is defined in equation (6.107):

$$\frac{\partial \hat{y}_o}{\partial R_j} = \begin{bmatrix} 0_{3\mathrm{x}11} \\ \frac{\partial \hat{y}_{\theta\phi}}{\partial R_j} \end{bmatrix} \tag{6.137}$$

Being:

$$\frac{\partial \hat{y}_{\theta\phi}}{\partial R_j} = \left[\frac{\partial \hat{y}_{\theta\phi}}{\partial u_2 v_2} \quad 0_{2\mathrm{x}9}\right] \tag{6.138}$$

Where:

$$\frac{\partial \hat{y}_{\theta\phi}}{\partial u_2 v_2} = \left[\frac{\partial \hat{y}_{\theta\phi}}{\partial h_2} \frac{\partial h_2}{\partial R_{WC}} \frac{\partial R_{WC}}{\partial h_u} \frac{\partial h_u}{\partial u_2 v_2}\right] \tag{6.139}$$

$$\frac{\partial \hat{y}_{\theta\phi}}{\partial h_2} = \left[\underset{\mathrm{a}}{\frac{h_{2z}}{(h_{2x})^2 + (h_{2z})^2}} \quad \underset{\mathrm{s}}{0} \quad \underset{\mathrm{g}}{\frac{-h_{2x}}{(h_{2x})^2 + (h_{2z})^2}}\right] \tag{6.140}$$

$$\begin{aligned} \mathrm{a} &= \frac{h_{2x}h_{2y}}{\left((h_{2x})^2+(h_{2y})^2+(h_{2z})^2\right)\sqrt{(h_{2x})^2+(h_{2z})^2}} \\ \mathrm{s} &= \frac{-\sqrt{(h_{2x})^2+(h_{2z})^2}}{\left((h_{2x})^2+(h_{2y})^2+(h_{2z})^2\right)} \\ \mathrm{g} &= \frac{h_{2z}h_{2y}}{\left((h_{2x})^2+(h_{2y})^2+(h_{2z})^2\right)\sqrt{(h_{2x})^2+(h_{2z})^2}} \end{aligned} \tag{6.141}$$

$$\frac{\partial h_2}{\partial R_{WC}} = R_{WC}(q^{WC}) \tag{6.142}$$

With $R_{WC}(q^{WC})$ being the rotation matrix depending on the current camera orientation quaternion q^{WC}. R_{WC} can be determined by the Matrix transform $\mathrm{Q_R}$ (Equation (6.47)) which converts from a quaternion q to rotation matrix R.

$$\frac{\partial R_{WC}}{\partial h_u} = \begin{bmatrix} {}^{-1}/_{f} & 0 & 0 \\ 0 & {}^{-1}/_{f} & 0 \end{bmatrix} \tag{6.143}$$

Is the jacobian for the pin hole camera model. And:

$$\frac{\partial h_u}{\partial u_2 v_2} = \begin{bmatrix} \frac{2(u-u_0)^2K_1}{(1+2K_1r^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+2K_1r^2}} & \frac{2(u-u_0)(v-v_0)K_1}{(1+2K_1r^2)^{\frac{3}{2}}} \\ \frac{2(u-u_0)(v-v_0)K_1}{(1+2K_1r^2)^{\frac{3}{2}}} & \frac{2(v-v_0)^2K_1}{(1+2K_1r^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+2K_1r^2}} \end{bmatrix} \tag{6.144}$$

Is the jacobian for the undistorted model (6.92).

Finally, the expansion of $\partial\rho/\partial \mathrm{R_j}$ which is defined in equation (6.107), is stated as follows:

$$\frac{\partial \rho}{\partial R_j} = \begin{bmatrix} \frac{\partial \rho}{\partial u_2 v_2} & \frac{\partial \rho}{\partial u_1 v_1} & \frac{\partial \rho}{\partial r^{WC}{}_{\gamma}} & \frac{\partial \rho}{\partial q^{WC}{}_{\gamma}} \end{bmatrix} \tag{6.145}$$

Where:

$$\frac{\partial \rho}{\partial u_2 v_2} = \begin{bmatrix} \frac{\partial \rho_\gamma}{\partial \rho_{\gamma o}} \frac{\partial \rho_{\gamma o}}{\partial \rho_{\gamma c}} \frac{\partial \rho_{\gamma c}}{\partial \rho_{h_2}} \frac{\partial h_2}{\partial R_{WC}} \frac{\partial R_{WC}}{\partial h_u} \frac{\partial h_u}{\partial u_2 v_2} \end{bmatrix} \tag{6.146}$$

In the equation above, the components are estimated respectively from (6.118), (6.119), (6.135), (6.142), (6.143) and (6.144). The next component of $\partial\rho/\partial \mathrm{R_j}$ is:

$$\frac{\partial \rho}{\partial u_1 v_1} = \begin{bmatrix} \frac{\partial \rho_\beta}{\partial \rho_{\beta o}} \frac{\partial \rho_{\beta o}}{\partial \rho_{\beta c}} \frac{\partial \rho_{\beta c}}{\partial \rho_{h_1}} \frac{\partial h_1}{\partial R_{WC}} \frac{\partial R_{WC}}{\partial h_u} \frac{\partial h_u}{\partial u_1 v_1} \end{bmatrix} \tag{6.147}$$

Where components are estimated using equations (6.125), (6.126) and:

$$\frac{\partial \rho_{\beta c}}{\partial \rho_{h_1}} = \begin{bmatrix} \frac{b_{1x}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{1}{2}\right)}} - \frac{(g) h_{1x}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{3}{2}\right)}} \\ \frac{b_{1y}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{1}{2}\right)}} - \frac{(g) h_{1y}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{3}{2}\right)}} \\ \frac{b_{1z}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{1}{2}\right)}} - \frac{(g) h_{1z}}{a^{\left(\frac{1}{2}\right)} s^{\left(\frac{3}{2}\right)}} \end{bmatrix}^{\intercal} \tag{6.148}$$

$$\begin{aligned} a &= (b_{1x})^2 + \left(b_{1y}\right)^2 + (b_{1z})^2 \\ s &= (h_{1x})^2 + \left(h_{1y}\right)^2 + (h_{1z})^2 \\ g &= (h_{1x})(b_{1x}) + \left(h_{1y}\right)\left(b_{1y}\right) + (h_{1z})(b_{1z}) \end{aligned} \tag{6.149}$$

Component $\partial h_1/\partial R_{wc}$ is estimated in the same manner than equation (6.142) but using the stored camera orientation quaternion q_λ which is defined in equation (6.80). $\partial R_{wc}/\partial h_u$ is estimated from (6.143), and $\partial h_u/\partial u_1 v_1$ is estimated in the same manner as (6.144) but using the stored pixel coordinates u_λ, v_λ defined in λ_i instead of the current pixel coordinates u, v.

$$\frac{\partial \rho}{\partial r^{WC}{}_\gamma} = \left[\frac{\partial \rho_\gamma}{\partial r^{WC}{}_\gamma} + \frac{\partial \rho_\beta}{\partial r^{WC}{}_\gamma} + \frac{\partial \rho_{b_2}}{\partial r^{WC}{}_\gamma}\right] \tag{6.150}$$

Being:

$$\frac{\partial \rho_\gamma}{\partial r^{WC}{}_\gamma} = \left[\frac{\partial \rho_\gamma}{\partial \rho_{\gamma o}} \frac{\partial \rho_{\gamma o}}{\partial \rho_{\gamma c}} \frac{\partial \rho_{\gamma c}}{\partial \rho_{\gamma b}} \frac{\partial \rho_{\gamma b}}{\partial r^{WC}{}_\gamma}\right] \tag{6.151}$$

The above component is estimated in the same manner as equation (6.117). The only difference is that the last factor is equal to an identity matrix:

$$\frac{\partial \rho_{\gamma b}}{\partial r^{WC}{}_\gamma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.152}$$

$$\frac{\partial \rho_\beta}{\partial r^{WC}{}_\gamma} = \left[\frac{\partial \rho_\beta}{\partial \rho_{\beta o}} \frac{\partial \rho_{\beta o}}{\partial \rho_{\beta c}} \frac{\partial \rho_{\beta c}}{\partial \rho_{\beta b}} \frac{\partial \rho_{\beta b}}{\partial r^{WC}{}_\gamma}\right] \tag{6.153}$$

The above component is estimated in the same manner as equation (6.124). The only difference is the last factor:

$$\frac{\partial \rho_{\beta b}}{\partial r^{WC}{}_\gamma} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{6.154}$$

$$\frac{\partial \rho_{b_2}}{\partial r^{WC}{}_\gamma} = \left[\frac{\partial \rho_{b_2}}{\partial \rho_{\beta o}} \frac{\partial \rho_{\beta o}}{\partial r^{WC}{}_\gamma}\right] \tag{6.155}$$

The above component is estimated in the same manner as equation (6.131). The only difference is the last factor:

$$\frac{\partial \rho_{\beta o}}{\partial r^{WC}{}_{\gamma}} = \begin{bmatrix} \frac{b_{2x}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \\ \frac{b_{2y}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \\ \frac{b_{2z}}{\sqrt{(b_{2x})^2 + \left(b_{2y}\right)^2 + (b_{2z})^2}} \end{bmatrix}^{\mathsf{T}} \tag{6.156}$$

$$\frac{\partial \rho}{\partial q^{WC}{}_{\gamma}} = \left[\frac{\partial \rho_{\beta}}{\partial \rho_{\beta o}} \frac{\partial \rho_{\beta o}}{\partial \rho_{\beta c}} \frac{\partial \rho_{\beta c}}{\partial \rho_{h_1}} \frac{\partial \rho_{h_1}}{\partial q^{WC}{}_{\gamma}}\right] \tag{6.157}$$

The three first components are estimated in the same manner as equations (6.125), (6.126) and (6.148).

The component $\partial\rho/\partial q^{WC}{}_{\gamma}$ is obtained in the same way than equation (6.64) but using but using the stored camera orientation quaternion q_{λ} which is defined in equation (6.80). The vector C, used in the estimation of equation (6.64), is equal to the vector h_1, defined in equation (6.84).

6.2.15 Establishing the Metric Scale

In monocular SLAM the scale of the observed world cannot be obtained using only vision. Therefore, another sensor or the observation of a known dimension reference has to be used in order to retrieve the scale of the world. In [156] an A4 sheet of paper is proposed to be used as initial metric reference. In this kind of metric initialization four points (each corner of the sheet) have to be previously known.

For our work, an only three point initial metric reference is proposed (Figure 6.10). The conditions for the initial metric reference are that the points r_1,r_2 and r_3 are enough salient in the image to be detected and matched, and the distance a,b and c between the points have to be known; several objects can be used for initial metric reference.

Prior to the first Kalman step, the three points are manually selected in the image. Therefore, the coordinates image projections of r_1, r_2 and $r_3$are known. Where:

$$r_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad r_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \quad r_3 = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} \tag{6.158}$$

Using equation (6.84) and the projections of r_1,r_2 and r_3 is possible to estimate three unitary vectors u_1 , $u_2$and u_3 pointing to r_1,r_2 and r_3 so:

$$u_1 = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{y1} \end{bmatrix} \quad u_2 = \begin{bmatrix} u_{x2} \\ u_{y2} \\ u_{z2} \end{bmatrix} \quad u_3 = \begin{bmatrix} u_{x3} \\ u_{y3} \\ u_{z3} \end{bmatrix} \tag{6.159}$$

$$r_1 = d_1 u_1 \quad r_2 = d_2 u_2 \quad r_3 = d_3 u_3 \tag{6.160}$$

Where d_1, d_2 and d_3 are the metric depth of r_1,r_2 and r_3. The Euclidean distances a, b and c are defined by:

$$\begin{aligned} a^2 &= (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \\ b^2 &= (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2 \\ z^2 &= (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 \end{aligned} \tag{6.161}$$

From equations (6.159), (6.160) and (6.161), the next non-linear system is stated:

$$\begin{aligned} a^2 &= (d_2 u_{x2} - d_1 u_{x1})^2 + \left(d_2 u_{y2} - d_1 u_{y1}\right)^2 + (d_2 u_{z2} - d_1 u_{z1})^2 \\ b^2 &= (d_3 u_{x3} - d_2 u_{x2})^2 + \left(d_3 u_{y3} - d_2 u_{y2}\right)^2 + (d_3 u_{z3} - d_2 u_{z2})^2 \\ z^2 &= (d_3 u_{x3} - d_1 u_{x1})^2 + \left(d_3 u_{y3} - d_1 u_{y1}\right)^2 + (d_3 u_{z3} - d_1 u_{z1})^2 \end{aligned} \tag{6.162}$$

The equations system in (6.162) is then solved for d_1, d_2 and d_3 using an optimization method.

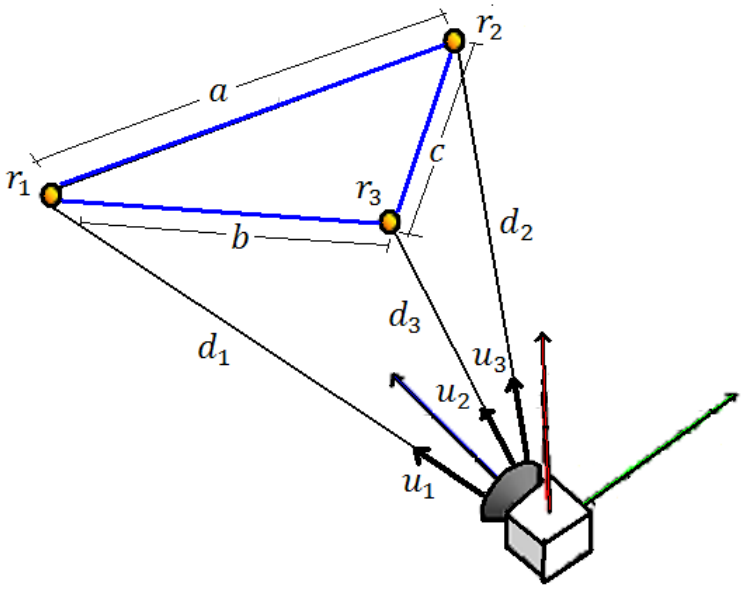


Figure 6.10 Initial metric reference parameterization.

A possible ambiguity in the solution can be avoided for example if one of the points r_1,r_2 and r_3 is always chosen to be the closest to the initial camera position, and the initial values for d_1, d_2 and d_3 for the optimization method are selected based in this rule.

The 3D positions of r_1, r_2 and r_3 with respect to the camera are estimated using equation (6.160). Finally r_1, r_2 and r_3 are added to the system state and the covariance matrix with zero uncertainty as was shown in Section 6.1.2.

Figure 6.11 illustrates the reconstruction of the 3D positions r_1, r_2 and r_3, from three corners of a computer monitor.

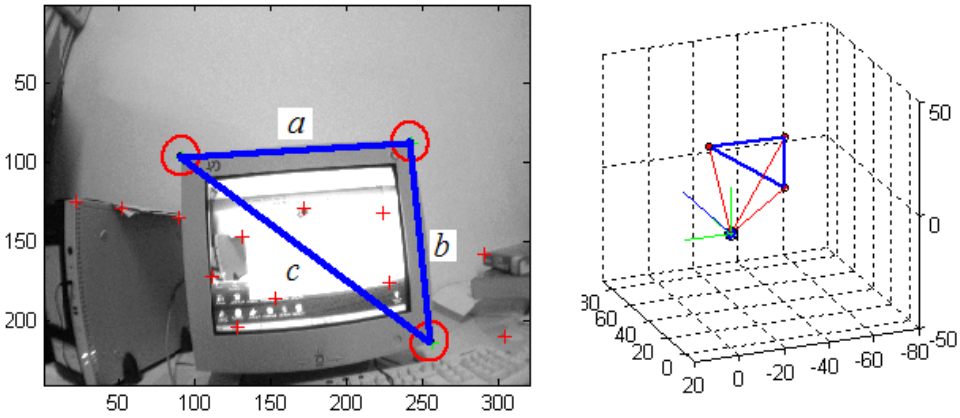
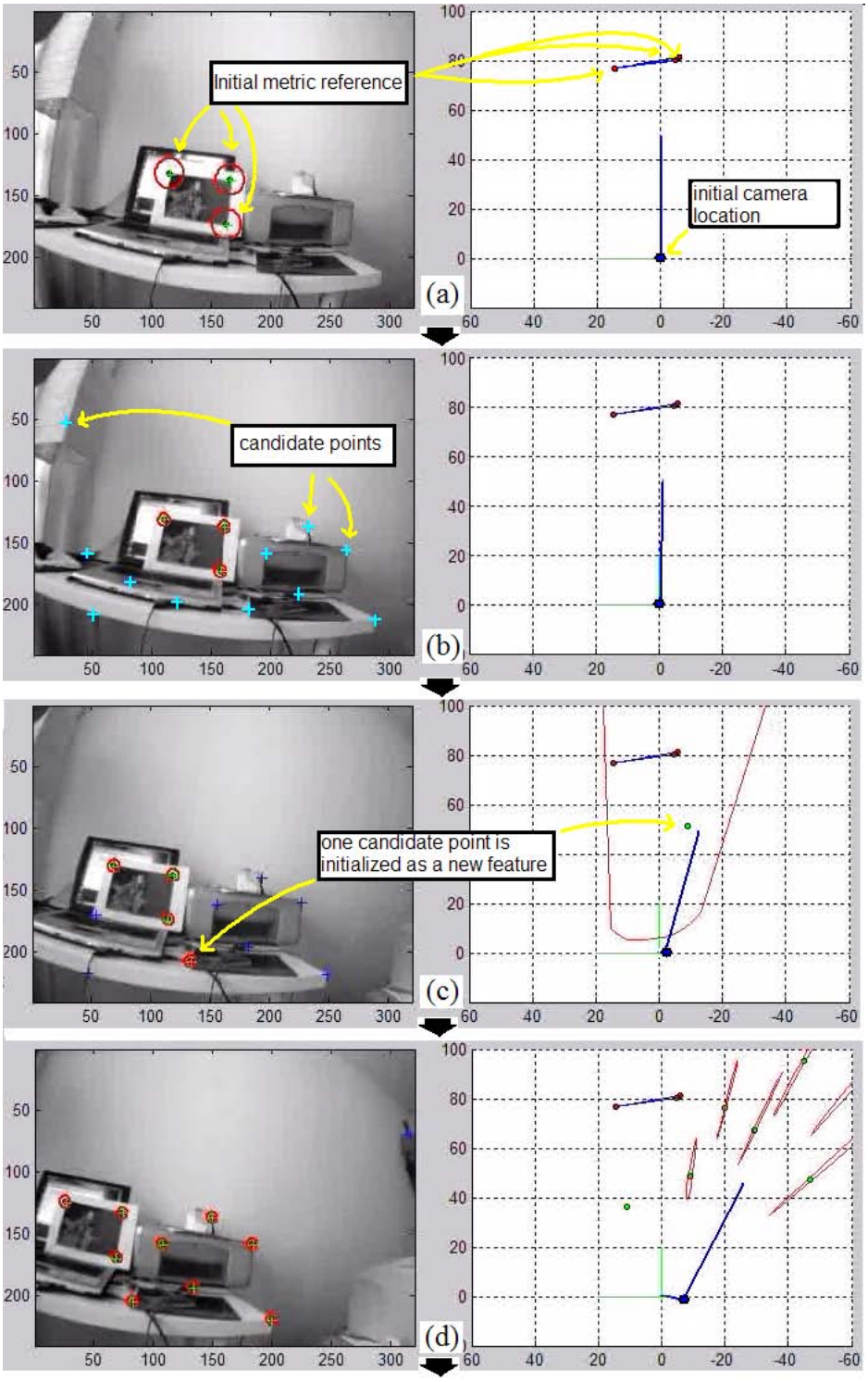


Figure 6.11 In this example, a computer monitor (left plot) was used as the only reference for recovering the scale of the world. The distance a,b and c between the points have to be known. 3D locations (right plot) for r_1, r_2 and r_3 are estimated from its 2D pixel location.

6.2.16 Experimental Results

Several experiments were realized in order to evaluate the performance of our proposed algorithm. For the entire set of monocular SLAM experiments presented in this thesis, an unexpensive monochrome IEEE web-cam was used. For each experiment, image sequences of 320 x 240 pixels were taken at 30 frames per second (30 fps).

The general methodology for the experiments consist on carry the camera on hand and move it over a predefined path while a video was reordered. After that, the video was used as the input for the proposed algorithms. Finally, the estimated camera trajectory was compared by inspection with the predefined path, in order to corroborate if the estimations were congruent with the real movement of the camera.


Initial metric reference
initial camera location
(a)
candidate points
(b)
one candidate point is initialized as a new feature
(c)
(d)

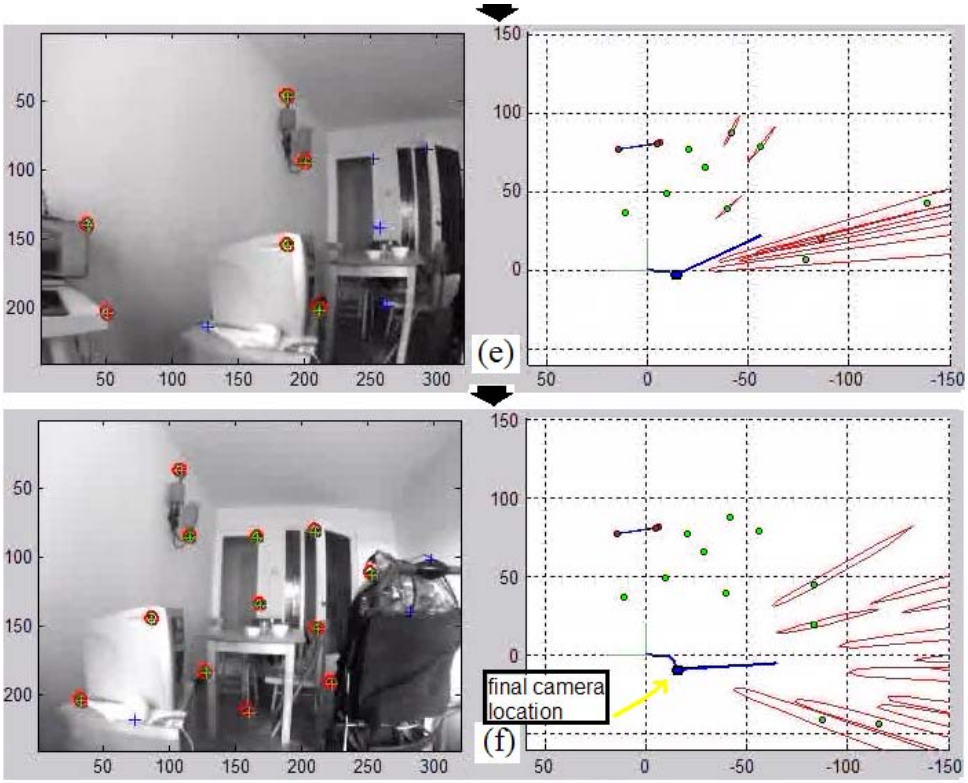


Figure 6.12 This sequence of plots illustrates the performance of the *Delayed Inverse Depth monocular SLAM method*. First a video was recorded while the camera was moved inside a living room. Later the video was used as the input of a MATLAB implementation of the algorithm. Prior to run the algorithm, three point of a sheet of paper (of knowing dimensions) were selected as the metric reference for recovering the metric scale of the world. At frame 1 (plot a) the initial metric reference is initialized in the map (plot a-right), the camera position is illustrated with a solid-blue sphere and its orientation with a blue line. Note that all the plots are shown from an X-Z view (top view). At frame 30 (plot b) several candidate points have been detected, note (in the image and map) that the camera slightly begins to move. Until frame 125 (plot c) one of the candidate point is initialized as a new feature map, in this case with a huge initial uncertainty (illustrated with the red ellipse). Note that features tracked with low uncertainty among the images sequence, could be mapped to its 3D position with a huge uncertainty. Later the gathered information is used to minimize uncertainty. At frame 200 (plot d) several features have been added to the map and the movement of the camera begins to be more evident. At frame 300 (plot e) the rotation of the camera is also notorious, observe that the estimated camera orientation represent the real orientation of the camera. Also note that, as the camera moves, new candidate points were detected in order to initialize new features for covering new unexplored regions. Plot f illustrates the final camera pose and map for this experiment.

The features map estimates were also checked by inspected for founding congruency with their real 3D location.

Figure 6.12 illustrates an experiment using the Delayed Inverse Depth Monocular SLAM algorithm. A video was taken moving the camera inside the living room of a house. At the end of the sequence it can be observed how the trajectory of the camera and part of the structured of the environment are recovered by the algorithm. In the next section, our proposed method is compared with the Un-delayed feature initialization method.

6.2.17 Comparing Delayed and Un-delayed Methods

In chapter 4 section 4.3.9, our delayed method was tested against the un-delayed method, for a 2DOF odometry context. In the current section we evaluate the performance of the delayed method compared with un-delayed method in a monocular SLAM context. It is important to note that the un-delayed method, initially proposed in [1] was defined as a fully monocular SLAM method. In fact, for experiments presented in section 4.3.9, the un-delayed method was modified for working in a 2D robotics context. On the other hand, the evaluation between methods presented in this section, was realized without modifying the un-delayed method.

Several image sequences, moving the camera through different trajectories and different scenarios, were recorded following a predefined path, in order to compare both methods. The trajectories were designed in order that, if a feature is left behind by the movement of the camera, this feature will not appear in image again in subsequent frames.

Usually, when an initial metric reference was used in order to recover the scale of the world (which is very important for several applications), it was observed a lack of robustness in experiments with the un-delayed method. This behavior is congruent with results observed with the un-delayed method applied to a 2D robotic context (section 4.3.6).

Figure 6.14 illustrates some of the drawbacks of the un-delayed method, as the need of tuning for ensures the convergence of estimations; the graphics in left part show the un-delayed method with an initial feature depth of $50\ \mathrm{cm}$, in frame 2 (left upper), it is possible to observe that reference points are located approximate $80\ \mathrm{cm}$ from the initial camera position and the first observed points are immediately initialized. However at frame 320 (left lower) the mapped features never converge, respect to the metric reference. Camera trajectory either converges; note the 4 points corresponding to the printer located besides the initial three point reference. On the other hand when we use an initial depth equal to $60\ \mathrm{cm}$, (right upper and lower graphics) the map and camera trajectory converge reasonably well.

Figure 6.13 illustrates the results using the delayed approach for the last experiment. In this case, the first feature is added to the map until frame 125 with a huge

initial uncertainty (upper graphic). Nevertheless at frame 320 (lower graphic) the map and trajectory converges. Note that the first added feature was initialized very near to its final position, and its uncertainty was minimized.

The condition for detecting new points with the Harris Corner detector for both methods is applied if the number of actives features in image goes below 30; in this case the detector is applied over the free features image regions.

Figure 6.15 shows the results for three different sequences. Real final camera position was manually added to the graphics (in black) in order to make easier the comparison. The initial and final frames are showed in the center for each sequence.

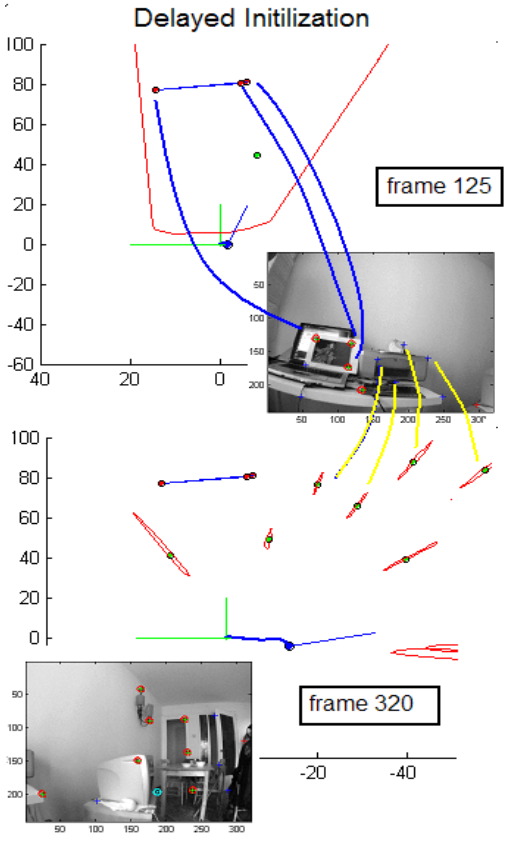


Figure 6.13 In this experiment with the delayed method (the same presented in the previous section 6.2.16), note that the features locations in the map (by the frame 320) are congruent with the observed image-locations (E.g. observe the features related to the printer besides the three-point initial metric reference). In this figure just remember that the maps are presented from a top-view (x-z view).

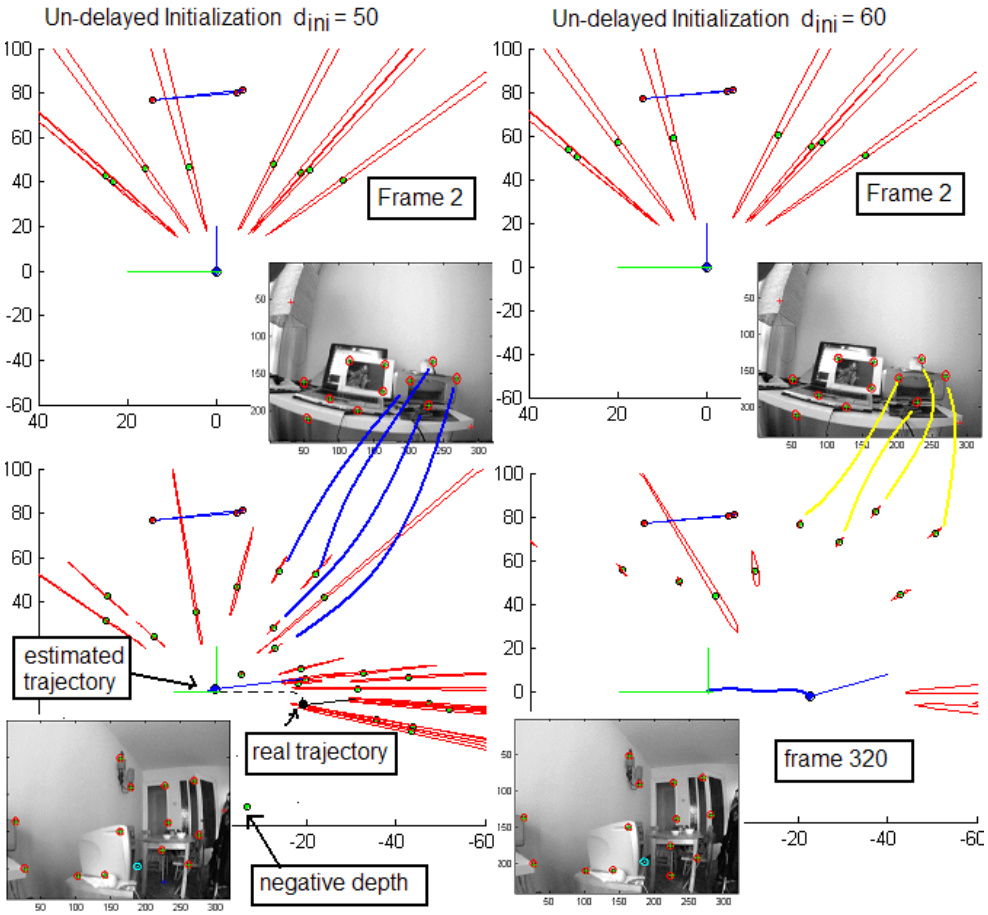


Figure 6.14 Using the un-delayed method, the same drawbacks, observed in a 2D robotic context (section 4.3.6), can be observed again for a monocular context. In this experiment (the same scenario presented in section 6.2.16) note that the initial inverse depth $\hat{\rho}_i = 1/d_{ini}$ has to be tuned in order to make the un-delayed method converge. When a d_{ini}=50cm is used (left) observe that neither the estimated camera trajectory nor the features estimated locations converge to the real ones. For example observe that the features related to the printer are estimated too close to the initial camera position, comparing with the three-point initial metric reference (In the reality, the sheet containing the metric reference and the printer are almost in the same plane). Moreover, it can be seen that one feature location was estimated in back to the camera (negative depth). On the other hand, when the initial inverse depth are tuned for initializing the features a little more close to the three-point initial metric reference, it can be note that the un-delayed algorithm converge reasonably well.

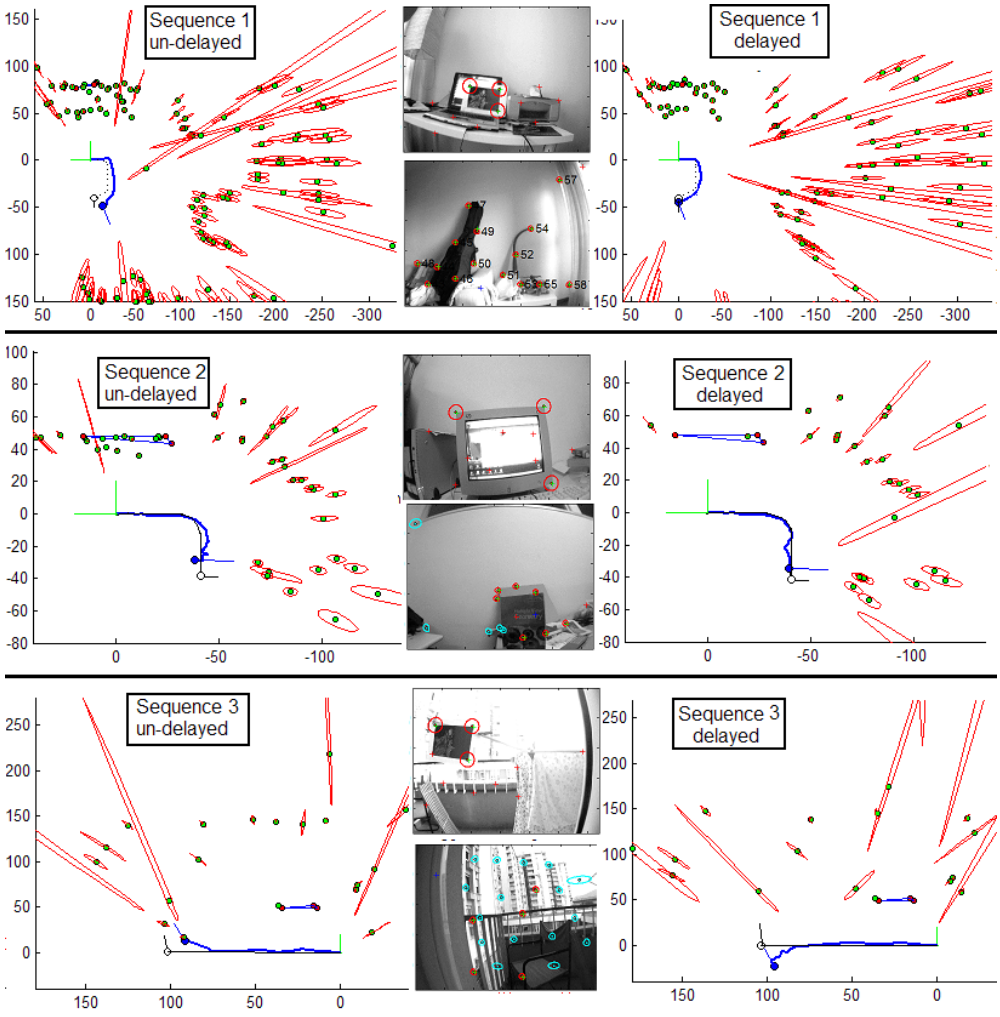


Figure 6.15 Camera trajectory and map estimates for three different video sequences. Un-delayed estimates are showed in left graphics and delayed estimates in right graphics. For each experiment, the first and the last frames of the sequences are showed. The first sequence corresponds to 760 frames of a house living room and it's the same sequence (extended) used in previous experiment. The second sequence corresponds to 480 frames taken in a workplace. Note that a PC monitor was used as initial metric reference. The third 360-frame sequence was taken following a simple linear path, but in a more occluded terrace building environment, with very near and very distant features. It is important to say that for these experiments the un-delayed method was tuned in order the ensure convergence.

Sequence	Method	σ ,x,y,z	Nf	%c	Nfc	E	Nf < 0
1	Un-delayed	4	47	42	114	5.7	0
	Delayed	4.1	35	27	110	1.44	0
2	Un-delayed	2.1	46	76	36	11.2	0
	Delayed	2.4	28	82	45	9.12	0
3	Un-delayed	1.4	34	44	45	17	2
	Delayed	2.5	27	55	58	19	1

Table 6.1 Results at the end of the three sequences. **(σ ,x,y,z)**: Summed standard deviation for the x,y,z position of the camera. **(Nf)**: Total number of features added to the system. **(%c)**: Percentage of features that present convergence. **(Nfc)**: The average number of frames needed for the convergence of the features. **(E)**: The metric error distance in cm from the real to final estimated trajectory. **(Nf < 0)**: Number of negative inverse depth estimated at the final of the trajectory.

Due to the lack of robustness observed for the un-delayed method, when an initial metric reference is used, the initial inverse depth parameter was tuned in order to ensure convergence. In this manner, it can be appreciated others interesting aspects for comparison between both methods.

Table 6.1 shows the results for each sequence for different aspects. In our experiments we consider that a feature converges when its depth uncertainty σ represents less than 5% of its depth, in this way we consider a convergence measurement proportional to the distance. The depth of a near feature should be estimated in a more accurate manner than a distant feature.

In these experiments the resulting camera trajectory estimate using the delayed method was similar to the estimate by the un-delayed method. In aspects relating with features depth convergence the results were similar for both methods.

As it was previously seen, initializing features in the first observed frame (un-delayed initialization) avoids the use of pre-initialized features in the state and allows the use of all the information available in the feature since it is detected. On the other hand, in real dynamics environments, it often happen that, rapid changes in lighting, moving objects, or even a PC screen, could produce that the saliency operator (used when features are automatically added to the map) detects, false, very weak or un-tractable features. In this case because the un-delayed method initialized features at the first frame observed, the quality of the detected points cannot be evaluated.

Weak long-term image features are difficult to match them in subsequent frames. Therefore when a minimum number of active image features want to be maintained in the map, it could happen that unnecessary initialization are realized. New feature initialization introduces biases to the system [89]. Moreover, it could happen that

some initialized feature violates the assumption of a rigid scene therefore increasing the chance of divergence.

When the delayed method does the tracking of candidate points implicitly test their quality. (e.g. if a bright produce a candidate point is almost sure that tracking process will prune this weak image point, avoiding its initialization as a new feature in the map). This is an interesting aspect of delayed method, because by nature is more restrictive for adding new features.

Therefore a reduced percentage of new features are added to the map (20-40%) respect to the un-delayed method, without losing the quality of the map. This aspect is very desirable, because bigger environments can be mapped with the same number of features and avoiding initialization of weak long term features can increase robustness.

On the other hand, it is clear than additional computational cost is added in the delayed method, since the candidate points have to be added to the estimation process and the Jacobian to estimate the new covariance matrix is more complex respect to the one used with the un-delayed method. However is well known that Kalman filter computation cost scales poorly with the size of the state, and the saving computational cost using 20-40% of the total amount of features should be higher than the computational cost added by the delayed method.

6.3 Distributed Monocular SLAM

In recent years monocular SLAM approaches ([109], [1], [156]) have shown good results in real-time 6-DOF camera pose and orientation estimation, and also building 3D maps of 50-100 sparse features. Several important improvements to the robustness of this kind of methods have also appeared ([158], [160], [161]). There are also some recent approaches for increasing the amount of features in the map maintaining real time operation ([115], [162]). Nevertheless, two of the main challenges at this moment are probably the use of these methods for applications that require more features in the map and the closing of loops with big drift in the estimation.

Davison [156] demonstrates the feasibility of real time operation SLAM with a single camera, using (as our approach) an Extended Kalman Filter (EKF) to propagate the camera pose and velocity estimates, as well as feature estimates.

The EKF maintains a full $N \; x \; N$ covariance matrix for N features, requiring $O(N^2)$ space. This covariance is updated with each measurement at $O(N^2)$ computation cost. This time and space requirements normally limit the total number of features into the map around of 100 (or even less) if real time operation is desired. Davison's approach estimates the features depth using a particle filter; when the estimate collapses, the

feature is fully initialized in the map. As it was seen in before, the main drawback of this kind of feature initialization is the measurement range depth; if depth want to be estimated in a fast manner, a limited number of particles have to be used, reducing the depth working range to 5 meters, therefore far features, typically found in outdoor environments, cannot be used.

The Unified Inverse Depth Method (Un-delayed method) presented by Montiel [1], overcomes this limitation, because depth parametrization can improve the linearity of the measurement equation, even for small changes in the camera position, converted in small changes in the parallax angle; this fact allows a Gaussian distribution to cover uncertainty in depth which spans a depth range from nearby to infinity, therefore distant points can be coded. Hence, far features can be included in the map (Distant features will not produce parallax but they are useful to estimate the camera orientation). This method can be extended to outdoor environments because far features can be included in the map. However, as the others EKF-based methods, it suffers from the limitation in the number of features, if real time operation wants to be achieved.

Possibly the first idea to deal with the size limitation of the maps, in these kind of methods, is to increase the computational power; as the power of the computers grows then bigger covariance matrix can be updated in real time. However the number of map features always will reach some level where the frame rate operation will become impractical.

A huge computational power, required for implementing monocular SLAM algorithms, could also reduce the range of applications, moreover in following paragraphs we will see that the problem is a little bit more difficult than increasing the computational power.

Another approach for increasing the size of the maps in monocular SLAM could be the use an efficient estimation technique that makes constant, or at least reduce, the dependence of the computational cost on the number of features included in the map.

In [115], Eade and Drummond apply the latter idea, adapting a FastSLAM scheme [73] for monocular SLAM. The position of each new partially initialized feature added to the map is parametrized with three coordinates representing its direction and inverse depth relative to the camera pose at the first observation, the estimates of these coordinates are refined within a set of Kalman Filters for each particle of the map.

The total cost of updating landmark estimates and optimizing the proposal over M particles given k observations is $O(Mk)$, independently of the number of landmarks N. Results show that the processing time per frame is nearly independent of the number of landmarks in the map, while 20-30 features observations updates are made every frame. This method shows to build maps of 250 features, representing the pose with 50 particles. But the absence of an explicit full covariance matrix can make loop-closing more difficult. So, why large environments cannot be mapped property with these methods?

6.3.1 SLAM Paradox

Monocular SLAM methods typically use patch cross correlation and active feature search in order to match the features in subsequent frames. Patch cross correlation is simple and fast, and active search is attractive because it only focuses directly on the area of interest, by maximising computational resources and minimising the chance of obtaining a mismatch, ([156], [165]). These techniques are suitable when the incoming video has to be processed online at frame rate. Such techniques can be applied if a single camera is being used as the only sensorial input for the system.

In [166], where SIFT descriptors are used for the matching process, a high frame rate cannot be addressed due to the computational cost of descriptors. Then the robot's odometry is used for reducing the propagation of the uncertainty when the robot moves. If any other sensor -apart from a single camera- wants to be used, a high frame rate is recommendable because the base-line between subsequent frames is small. Therefore, the propagation of uncertainty is also small, constraining the search for matching features to reduced regions. Nevertheless when the robot moves far away from its initialization point and describing a different path (closing loops problem), patch cross correlation and active search are not suitable.

Consider a typical run of a monocular system, for a sequence of 1750 frames taken following a cycled trajectory (illustrated by the rectangle) in a laboratory environment (Figure 6.16, upper). In Section 6.3.5 the general setup for this experiment is explained in depth. The final map contains about 350 features and therefore it has not been built in real time, but it is useful for illustrative purposes. It can be appreciated a huge drift at the end of the estimation. Note that the map and trajectory (except for the side effects due to the second turn) are tolerably consistent. Consequently, it is important to observe that the uncertainty in the camera position is maintained stable over the whole sequence (Figure 6.16, lower). Therefore, it doesn't reproduce the real error propagation. As a result, when the camera returns near to its initial position, the active search technique is neither able to predict the position of the old features nor to find the closing loops after long trajectories in order to minimize the drift.

$$P = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} \end{bmatrix}$$

$$P_{new} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \mathrm{P}_{xx}\nabla G_x{}^{\mathsf{T}} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \mathrm{P}_{y_1x}\nabla G_x{}^{\mathsf{T}} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \mathrm{P}_{y_2x}\nabla G_x{}^{\mathsf{T}} \\ \nabla G_x \mathrm{P}_{xx} & \nabla G_x \mathrm{P}_{xy_1} & \nabla G_x \mathrm{P}_{xy_2} & \nabla G_x \mathrm{P}_{xx} \nabla G_x{}^{\mathsf{T}} + \nabla G_z \mathrm{R} \nabla G_z{}^{\mathsf{T}} \end{bmatrix} \quad (6.163)$$

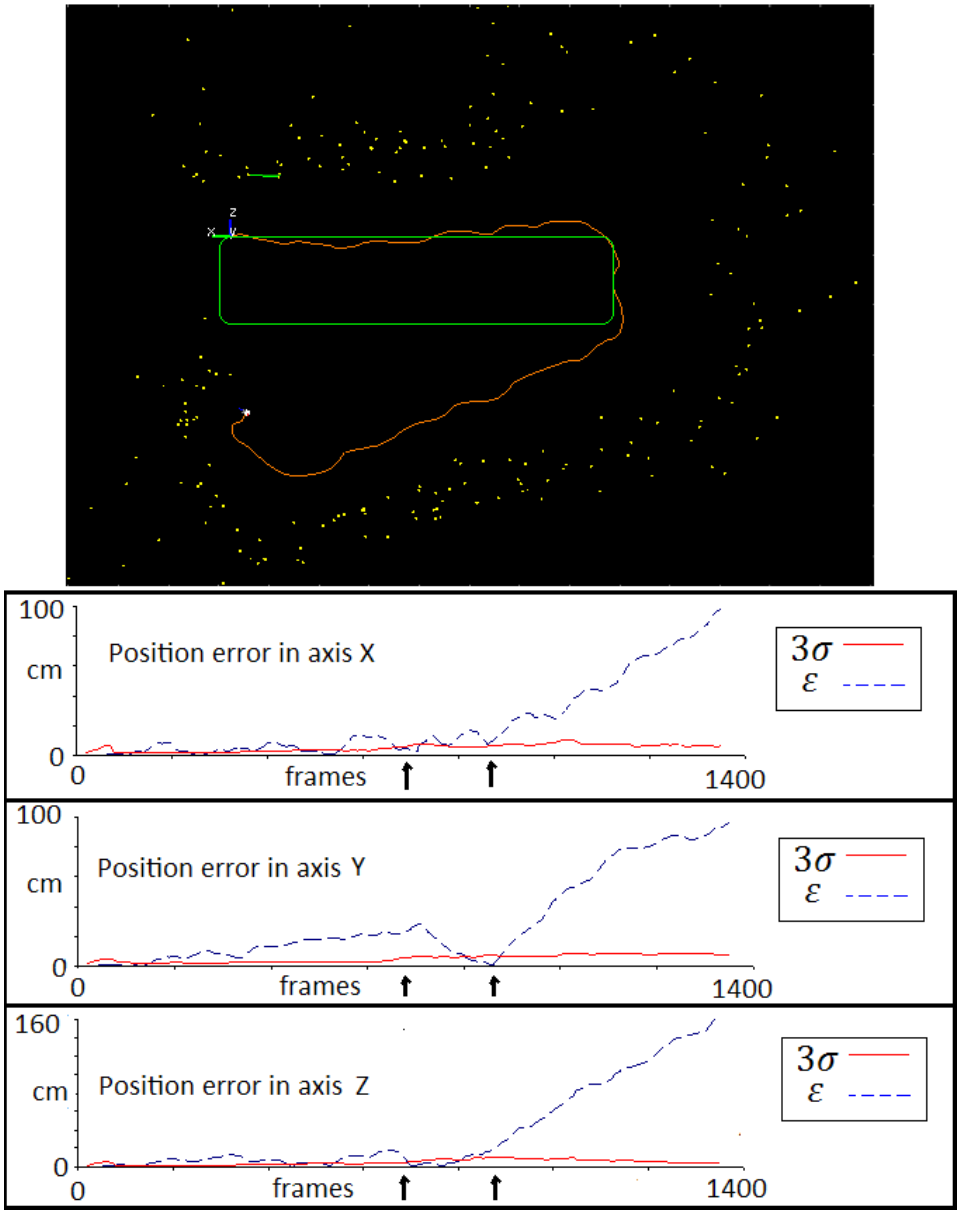


Figure 6.16 Drift in Monocular SLAM estimations. The upper plot illustrates the real and estimated camera trajectory for a sequence of 1700 frames. Lower plots show the estimation errors of camera position and their corresponding 3σ variance for 1350 frames. Note that the second turn is the main reason in error propagation.

In addition, when new features have to be added to the map and the covariance matrix P is updated (6.163), a bias is introduced in the system [89] due to the non-linearity in the measurement process. After several initializations, the bias introduced to the map could be substantial.

This bias can also affect the active search. In Active search techniques (section 6.2.6) the region search for a landmark is determined by projecting the expected landmark location into the image and calculating the innovation covariance S_i.

The innovation covariance S_i measures the deviation between the actual measurement z_i and its prediction. If some bias has been added to the system after several features initializations, the projection of the expected landmark location into the image and its image search region defined by S_i could be biased. The previous effect produces that the estimated uncertainty after a long trajectory could not represent truly the error propagation in camera pose, and therefore if the camera sees a previously mapped site, then the position in the image for old features cannot be predicted property.

Furthermore, when the camera moves far away from its initialization point and describing a different path, it is likely that the difference between the camera pose (when the features have been firstly observed) and the current camera pose is substantial, inducing huge changes in the image feature appearance due to variations in illumination or point of view. If the image feature appearance differs excessively, the patch cross correlation technique will be unfeasible to address the data association problem. In this context, image feature descriptors ([136], [167]) are techniques well adapted for addressing the data association problem in this latter context. Nevertheless, descriptors are difficult to apply directly to Monocular SLAM methods due to their high computational cost.

It is a clear trade-off between the need of a high frame rate operation and the data association techniques required for addressing the whole problem.

6.3.2 Distributed Approach

In the past, a single estimation process (EKF, particle filter...) has been tried when solving the whole problem of single camera SLAM. But it seems that trying to solve the whole problem at once is a very hard task. As a possible alternative, a distributed approach is proposed in which the whole task is divided into two concurrent estimation processes, each one designed to handle in a natural way the previously mentioned trade-off.

The general idea is to convert a monocular SLAM method (as the described in the previous part of this chapter) into a complex real-time “Virtual Sensor” that provides appearance-based sensing in the form of features descriptors and emulating typical sensors (laser and odometry), afterward a classic SLAM method is plugged in (decoupled from the camera’s frame rate) taking as its input the output of the Virtual Sensor; this

process, called Global SLAM, models the drift of the Virtual Sensor estimation as an independent uncertainty propagation, for estimating the global camera-robot pose and map. In our implementation, both estimation processes, the Virtual Sensor and the Global SLAM, run concurrently in different PCs in a local network, communicated with TCP/IP protocol. Figure 6.17 shows the diagram for the proposed scheme; note that same symbols are used to represent equivalent but not same variables in both Virtual Sensor and Global SLAM process.

From the estimation point of view, it might seem redundant if we used a SLAM process for building a global map by using the output of a modified Monocular SLAM process (Virtual Sensor) as its input. However, a closer view (as it was seen in previous section) reveals that Monocular SLAM is a particular case of the general SLAM problem and it is constrained to very specific considerations. In that sense, in following sections, we will see that our Distributed Framework handle in a natural manner the SLAM paradox described in previous section.

In a very recent work, [168], a sub-mapping technique for building maps over large camera trajectories has been presented. By applying such technique, real-time collected sub-maps are processed off-line for refinements and closing loop detection using an Iterated Kalman Filter. This method and the presented one in this paper share the idea of using an extra estimation process for building the final map. In addition, the architecture of the method proposed in this work is different from [168] among other aspects, because it looks for a fully concurrent and continuous online estimation of the global map.

An important aim of the system design is the modularity; in this work a basic implementation is proposed but not restricted to other methods and techniques; recent improvements to monocular SLAM, ([158], [160], [161]), can be included in the Virtual Sensor, different kind of feature descriptors, ([136], [167]), can be used for data association or a more efficient SLAM method can be used for implementing the Global SLAM module, [115].

6.3.3 Virtual Sensor

In section 6.2, a new kind of features initialization called “Delayed Inverse Depth Feature Initialization” is proposed in order to increase the robustness of Monocular SLAM methods when a reference is used in order to recover the metric scale of the environment. A new method to recover the scale was additionally proposed (section 6.2.15). In this section we explain how to modify a monocular SLAM method as a virtual sensor. This virtual sensor emulates range measurement and 3D odometry for dead reckoning and it extracts feature descriptors in order to handle the data association problem.

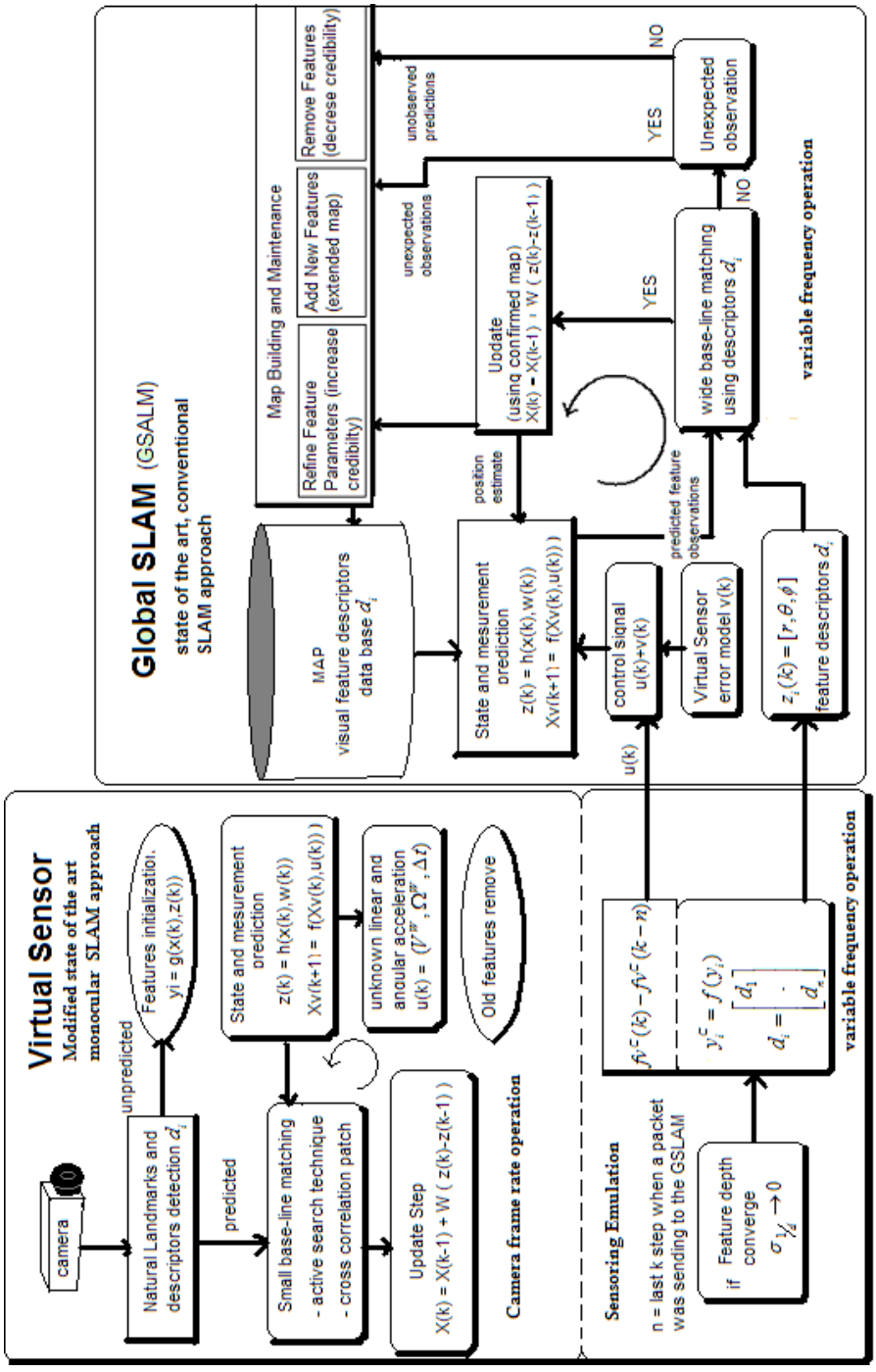


Figure 6.17 Diagram for the proposed Distributed Monocular SLAM.

As was stated before, in Monocular SLAM methods, in which the number of features in the system is increasing, maintaining real time operation becomes very difficult. Therefore the map size is limited typically around 100 features, reducing the working area of these methods. On the other hand, it is easy to remove old features from the state for maintaining a stable computational cost per frame and therefore a real time operation.

6.3.4 Real Time Operation

In this work the term "real-time" means that the time elapsed between acquired camera frames is large enough to processing a step of the algorithm. Hence for maintaining real time operation, the computation time for each EKF cycle should not exceed the time available between frames, therefore the number of features in the system have to be bounded. Removing features from the system state is much easier than adding them. To delete a feature from the state vector and covariance matrix, the rows and columns which contain it, have to be removed. An example in a system where the second of three features want to be removed is:

$$\begin{bmatrix} x_v \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \rightarrow \begin{bmatrix} x_v \\ y_1 \\ y_3 \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \mathrm{P}_{xy_3} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \mathrm{P}_{y_1y_3} \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \mathrm{P}_{y_2y_3} \\ \mathrm{P}_{y_3x} & \mathrm{P}_{y_3y_1} & \mathrm{P}_{y_3y_2} & \mathrm{P}_{y_3y_3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_3} \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_3} \\ \mathrm{P}_{y_3x} & \mathrm{P}_{y_3y_1} & \mathrm{P}_{y_3y_3} \end{bmatrix} \tag{6.164}$$

If the number of features exceeds a specific threshold, then older features that were left behind by camera movement are removed in order to maintain a stable amount of features.

6.3.5 Removing Old Features Experiment

A new C++ version of our Delayed monocular SLAM algorithm was implemented in order to increase speed. In previous experiments using the MATLAB implementation the camera was carry on hand and the working area of the experiment (for the camera's movement) was limited to the length of the camera's cable. In subsequent experiments, using the C++ implementation of the algorithm, the camera was mounted over a laptop, as it can be appreciated in Figure 6.18. Using this kind of setup, videos can be taken walking over long paths, increasing the working area of experiments.

Using the previously described configuration a video containing about 1720 frames (320 x 240 pixels at 30 frames per second) has been recorded walking over a predefined cycled trajectory inside a laboratory environment.

Figure 6.18 A real video, recorded in a laboratory environment, has been used in experiments. An inexpensive webcam is the only sensor system.

The path walked was a length of 12 meters (approximately). Here is important to note that, currently in the bearing-monocular SLAM community, there is not a formally established methodology, in order to test the performance of the methods, in terms of the size of the environments that algorithms can map. For example [168] states that their experiments were realized over very long out-door trajectories, around of 100 meters length. Nevertheless it can be appreciated that most of the features are located far from the camera (comparing with our experiments were the average of features are located at less than one meter of depth). As it know, the camera needs to travel more in order to "see parallax" in far features. From the algorithm point of view the scale is not a matter [157]; the estimation process for a camera traveling a distance x while it see a parallax y in a feature with depth 1 is the same that the estimation process for a camera traveling a distance $10x$ while see parallax y in a feature with depth 10. For that reason we guess that it can be inappropriate to use the size (in meters) of map and trajectory as parameter for evaluating the performance of bearing/monocular SLAM algorithms. Instead, we guess that the number of features, that the map can include, can give us a better idea of the performance of the methods.

Figure 6.19 (displayed in two pages) illustrates the results of applying the Delayed monocular SLAM algorithm, together with the auto-removing process of old features, to the image sequence previously described. Each plot shows (representing a single frame) the input image (augmented with different information) and the output (map and trajectory) of the algorithm.

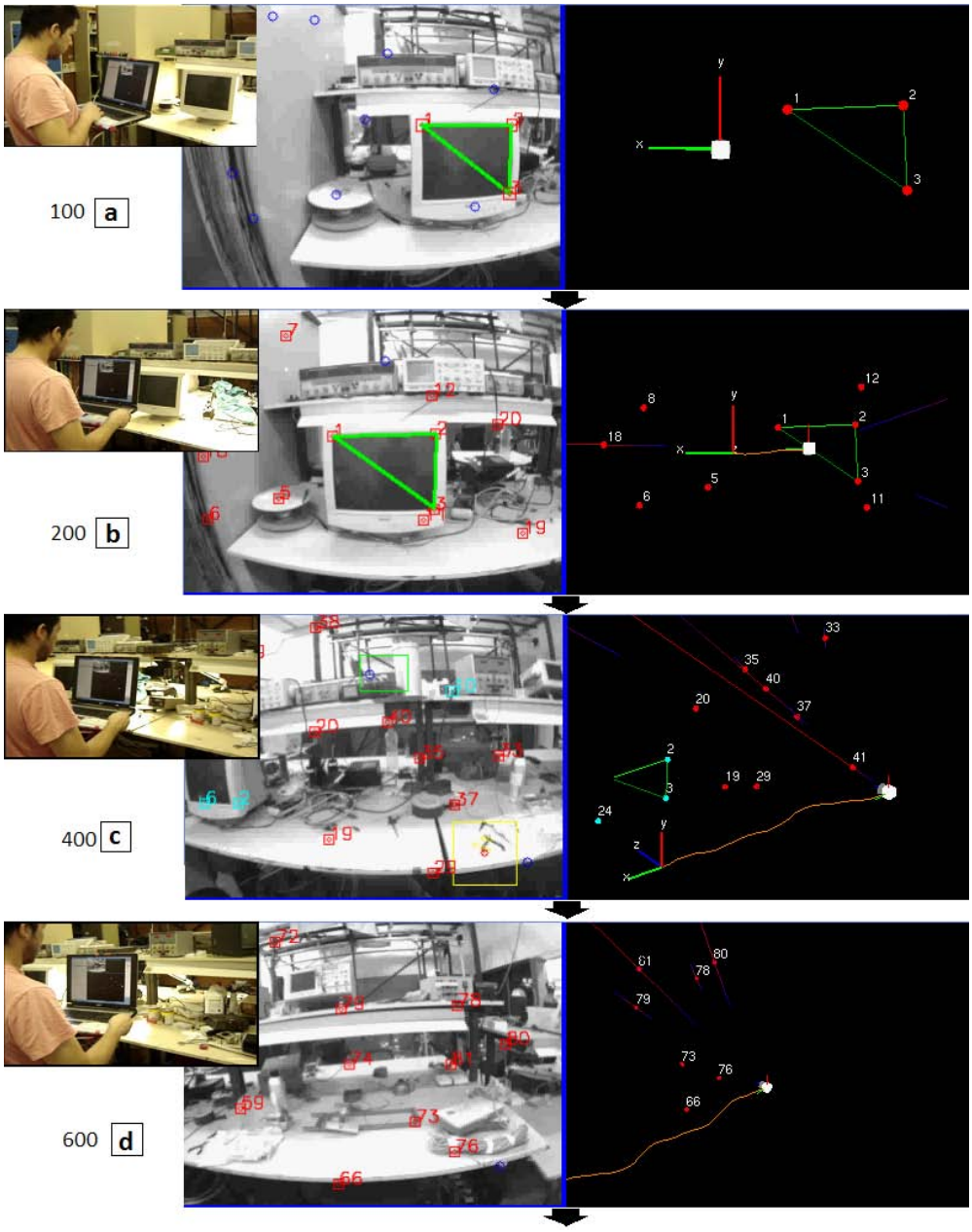
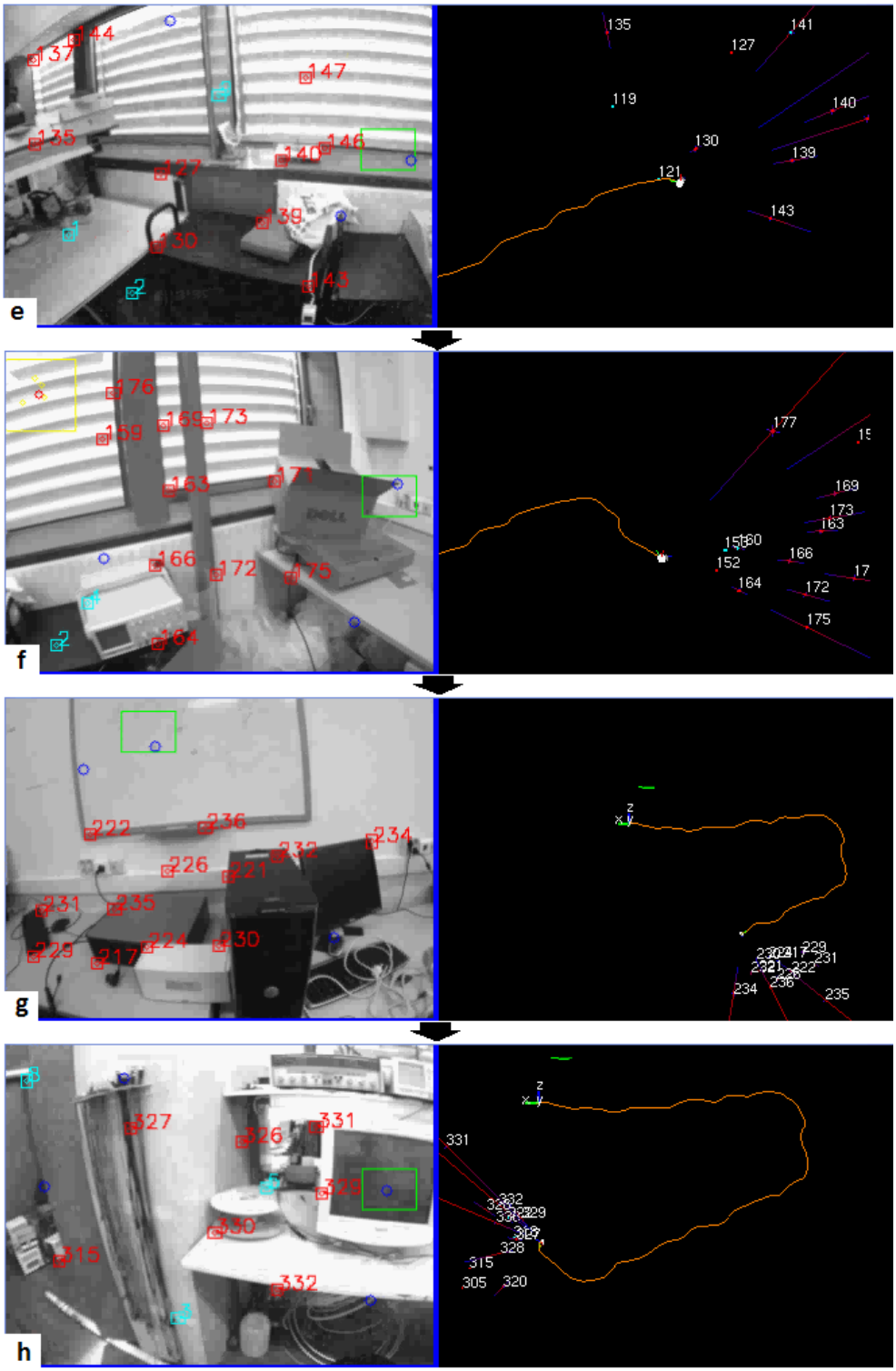


Figure 6.19 (Current and following page) Input and Output for the Delayed Monocular SLAM algorithm, removing old features for maintaining real time operation. In this experiment a video was taken inside a laboratory following (walking) a predefined closed trajectory. Note that estimated map and trajectory (right) are showed from different point of view.


e
f
g
h

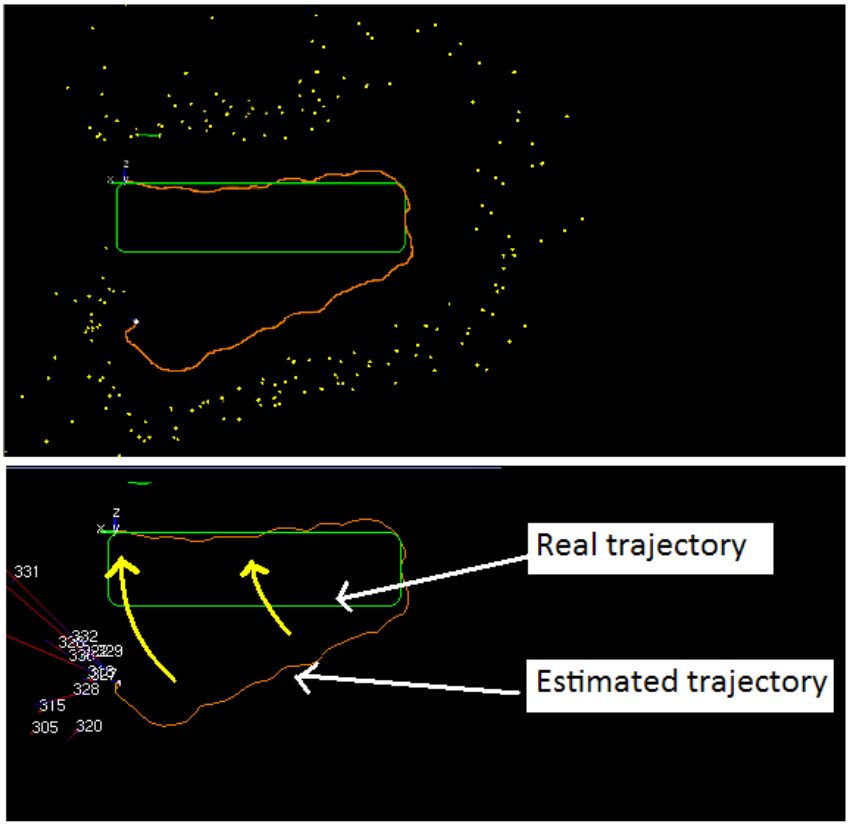


Figure 6.20 Estimated map and trajectory without removing features (upper plot). This map was not constructed at real time. Estimated map and trajectory removing features (lower plot). This map was constructed at real time. In both cases observe the drift in estimations.

For this C++ implementation the output is rendered using OpenGL, which it is useful in order to illustrate the 3D nature of the estimates. For the firsts four plots (left) an external picture of the experiments are showed, in order to point up the experimental methodology previously described.

At one of the initial frames (plot a), the three-point metric reference has been already initialized. Note that a PC monitor has been used for this purpose. At this early point of the sequence the three corners of the monitor (forming the metric reference) are the only features belonging to the map. But it can be appreciated that several candidate points have been already detected. Also note that the camera is initialized at the origin of the coordinate frame. In plot a, the map is showed from a X-Y view (aligned with the initial camera view).

By the frame 100 (plot b) the displacement of the camera to the right (to the $-x$ direction in the map absolute coordinate frame) starts to be notorious. Observe that

several candidate points have been initialized as features in map. By the frame 300 (plot c) the camera has moved more, leaving behind some features. Moreover, others ones cannot be successfully tracked. These features (illustrated in cyan) are aspirants to be removed from the system. Note that uncertainty in features estimation is exemplified by a 3D line, as the uncertainty decreases the line becomes shorter. In plot c the map is showed from an X-Y-Z view (3D view). By the frame 500 (plot d) it is very clear that several features have been removed from the map. In this experiment a feature is removed if it cannot be detected and tracked by 20 frames. By the frame 740 (plot e) the camera begins its first rotation, and by the frame 900 (plot f) the camera begins its second rotation. The frame 1200 is illustrated in plot g, in this plot the map is showed from an X-Z view (top view). At this point a discrepancy between the estimate and the real trajectory begins to be perceptible. Actually, the real path followed, in order to move the camera, is like a long rectangle (the real path is illustrated in Figure 6.20). By the frame 1700 (plot h) the camera has returned very near to its initial position (note that the PC monitor used as initial metric reference is appearing again). Observe that the number of features in the map has been maintained stabled over the whole sequence and therefore it was built at real time. For comparing purpose, in Figure 6.20 (upper plot) it can be appreciated the estimated map for the same sequence, if features are not removed from the system, in this case the map was not built at real time.

In Figure 6.20 is very clear that the drift, in the estimated map and trajectory, is similar in both experiments; without removing features (upper plot) and removing features (lower plot). Even when the oldest features remain in the map (upper plot), and for the reasons explained in section 6.3.1, the loop cannot be detected and thus making impossible to minimize the drift in estimates.

On the other hand, of course, if features are removed from the map, then previous mapped areas cannot be recognized in the future and loops cannot be detected. Therefore, the implicit drift in the estimations cannot be minimized. However a modified Monocular SLAM method to maintain stable computational operation can be viewed as a complex real-time “virtual sensor”, which provides appearance-based sensing and emulates typical sensors as laser for range measurements and encoders for dead reckoning (Visual Odometry).

6.3.6 Adapting monocular SLAM as Virtual Sensor

To adapt a monocular SLAM method as Virtual Sensor, first it must be modified in order to maintain stable computational cost per frame as was described in the latter section. In the subsequent we will refer to the “modified monocular SLAM sub-system” as the Virtual Sensor.

When a new feature is initialized (Figure 5.8) in the Virtual Sensor, we need to gather useful information in order to match it against future measurements. The followed approach is based in the matching technique explained in chapter 5, section 5.4:

A saliency operators, used in most of the mono-SLAM methods (Shi-Tomasi, Canny, etc), is applied in order to extract n SURF (Speeded Up Robust Features) descriptors d_i from a *p-by-p* patch centered in the point detected by the saliency operator, [136]. All extracted descriptors are stored and related to with the $\hat{y}_i$ feature. This way, each feature $\hat{y}_i$ can have a lot of useful information in order to be matched in the future. SURF descriptors are employed because they proof to have a performance which is similar to that of the SIFT (Scale-Invariant Feature Transform) [167], but they also have a lower computational cost [169].

Virtual Sensor and Global SLAM are communicated with the TCP/IP protocol. In our implementation, the Virtual Sensor is defined as the server which serves requests coming asynchronously from the Global SLAM. When a request is received, camera movement and features information (emulating a real range sensor) are sent.

Figure 6.21 illustrates the camera movement information o_{vs} sent to the Global SLAM defined by:

$$o_{vs} = \begin{bmatrix} o_x \\ o_y \\ o_z \\ o_\theta \\ o_\phi \end{bmatrix} = \begin{bmatrix} \Delta x^C \\ \Delta y^{WC} \\ \Delta z^C \\ \Delta\theta^{WC} \\ \Delta\phi^{WC} \end{bmatrix} = \begin{bmatrix} \Delta x^W \cos(\theta^{WC}_{k-n}) - \Delta z^W \sin(\theta^{WC}_{k-n}) \\ y^{WC}_k - y^{WC}_{k-n} \\ \Delta x^W \sin(\theta^{WC}_{k-n}) - \Delta z^W \cos(\theta^{WC}_{k-n}) \\ \theta^{WC}_k - \theta^{WC}_{k-n} \\ \phi^{WC}_k - \phi^{WC}_{k-n} \end{bmatrix} \quad (6.165)$$

where $\Delta x^W = x^W_k - x^W_{k-n}$, $\Delta z^W = z^W_k - z^W_{k-n}$ and (θ^{WC}, φ^{WC}) denote the camera orientation and they have been obtained from the quaternion q^{WC}. The k subscript denotes the current step and the k-n subscript is equal to the last k step at the specific moment in which information was sent to the Global SLAM.

Information about each feature $\hat{y}_i = [x\ y\ z\ \theta^{WC}\ \varphi^{WC}\ \rho]$ is only sent if its estimated depth converges and the feature is related to a minimum number i of descriptors d_i. The estimated depth r of a feature is considered to be converging if $100(\sigma_\rho)/\rho < l$, (in experiments l=5). The feature pose information is sent emulating a 3D range sensor (Figure 6.21):

$$s_i = \begin{bmatrix} r \\ \theta^C \\ \phi^C \end{bmatrix} = \begin{bmatrix} 1/\rho \\ atan2(\Delta z^W, \Delta x^W) \\ atan2\left(((\Delta x^W)^2 + (\Delta z^W)^2), \Delta y^W\right) - \phi^{WC}_k \end{bmatrix} \quad (6.166)$$

where r is the range and (θ^C, φ^C) defines the direction of r in the camera frame coordinates. The entire packet sent to the Global SLAM is made of [o_{vs},s_1,s_2,…s_n] and descriptors d_i. atan2 is a two-argument function that computes the arctangent of y/x given y and x, within a range of [-π, π].

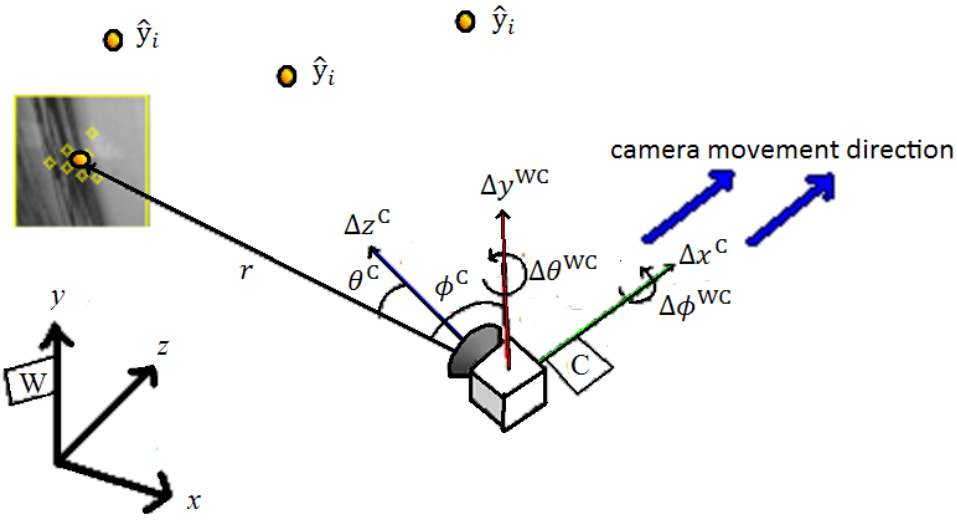


Figure 6.21 Parametrization for data sent by the Virtual Sensor.

6.3.7 Global SLAM: Minimizing Drift

In the context of the global SLAM subsystem, the Virtual Sensor, described in the previous section 6.3.3, is considered as a black box that provides appearance-based sensing in the form of features descriptors. Furthermore, it emulates typical sensors as laser for range measurements and encoders for dead reckoning.

Due to the main objective is to demonstrate the viability of the Distrubuted Framework for monocular SLAM, instead of demonstrate the performance of a particular kind of implementation, a very simple scheme of SLAM was used for implementing the Global SLAM subsystem, but other methods, such as [73], can be used. In that sense, is important to note that Jacobians related to Global SLAM subsystem are obviated.

For the inter-process communication the Global SLAM is defined as a client connected to the Virtual Sensor. In the Global SLAM, an EKF is also used to propagate the camera pose and the map. When a Kalman step is completed, a request is sent to the Virtual Sensor in order to obtain the latest odometry and range information $[o_{vs},s_1,s_2,\ldots s_n]$ as well as the descriptors.

As far the Global SLAM is concerned, the camera-robot state is defined by:

$$\hat{\mathrm{x}}_{\mathrm{v}} = [x\ y\ z\ \theta\ \phi]^{\mathrm{T}} \tag{6.167}$$

denoting pose and orientation in the world coordinate frame. The prediction model $f_v(\hat{\mathrm{x}}_{\mathrm{v\,(k)}}, u_{(\mathrm{k})})$ is:

$$f_v = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ \theta_{k+1} \\ \phi_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + u_x \cos(\theta_{v\_k}) - u_z \sin(\theta_{v\_k}) \\ y_k + u_y \\ z_k + u_x \sin(\theta_{v\_k}) - u_z \cos(\theta_{v\_k}) \\ \theta_k + u_\theta \\ \phi_k + u_\phi \end{bmatrix} \tag{6.168}$$

where $u_{(\mathrm{k})}$ is the control input:

$$u_{(k)} = u_{n(k)} + v_{(k)} \tag{6.169}$$

and $u_{\mathrm{n(k)}}=[\, u_{\mathrm{x}}\, u_{\mathrm{y}}\, u_{\mathrm{z}}\, u_{\theta}\, u_{\varphi}]^{\mathrm{T}}$ is taken from the data o_{vs} (6.165) sent by the Virtual Sensor, then $u_x=o_x$, $u_y=o_y$, $u_z=o_z$, $u_\theta=o_\theta$, $u_\varphi=o_\varphi$, being $v_{(\mathrm{k})}$ the noise added to the control input $u_{(\mathrm{k})}$ for modeling the uncertainty propagation of the Virtual Sensor odometry measurements. Then v_k is defined as Gaussian noise with zero mean and known diagonal covariance matrix U.

$$U = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\phi^2 \end{bmatrix} \tag{6.170}$$

In experiments variances are empirically tuned. Furthermore, in Figure 6.16 it can be appreciated that the traslational error is lower than the rotational error (a common behavior in visual SLAM). If it is assumed that the camera axis x is generally aligned with the traslational camera movement (Figure 6.21) (a common assumption in monocular SLAM using a wide lens camera), then it can be assumed that $\sigma_{\mathrm{x}} < \sigma_{\mathrm{z}}$.

The complete state that includes the features $\hat{\mathrm{y}}$ is built as $\hat{\mathrm{x}}=[\, \hat{\mathrm{x}}_{\mathrm{v}}{}^{T}, \hat{\mathrm{y}}_1{}^{T}, .. \hat{\mathrm{y}}_{\mathrm{n}}{}^{T}]^{\mathrm{T}}$ where a feature $\hat{\mathrm{y}}_i$ represents a 3D scene point i defined by $\hat{\mathrm{y}}_i=[x_i,y_i,z_i]$, denoting Euclidean position in the world reference.

The observation model $h(x_k,w_k)$ predicts a measurement $z_k=[r\ \theta\ \varphi]^{\mathrm{T}}$ of a 3D point $\hat{\mathrm{y}}_i$ as follows:

$$h^c = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2 + (z_i - z_v)^2} \\ atan2\big((z_i - z_v), (x_i - x_v)\big) - \theta_v \\ atan2\left(\sqrt{(x_i - x_v)^2 + (z_i - z_v)^2}, (y_i - y_v)\right) - \phi_v \end{bmatrix} \tag{6.171}$$

where w_k models uncertainty for the Virtual Sensor features measurements, assuming Gaussian noise with zero mean and covariance matrix R.

If a feature measurement $z_{\mathrm{i}}=[r\ \theta\ \varphi]^{\mathrm{T}}$ incoming from the Virtual Sensor (represented by s_i in (6.166)) is not matched against a previous mapped feature, then it is initialized as a new feature $\hat{\mathrm{y}}_{\mathrm{i}}$ in the map:

$$\hat{y}_{\text{new}} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} x_v + r\sin(\phi + \phi_v)\cos(\theta + \theta_v) \\ y_v + r\cos(\phi + \phi_v) \\ z_v + r\sin(\phi + \phi_v)\sin(\theta + \theta_v) \end{bmatrix} \quad (6.172)$$

When a feature is initialized its related d_{n} descriptors are stored in a data base and labeled with its corresponding $\hat{y}_{\text{i}}$ feature. For minimizing the probabilities of mismatch, a matching of an incoming feature with an already mapped one is only considered under the following circumstance: at least two different descriptors d_{n} from the measured feature must match with two different descriptors related with a single feature in the map. A fast approximate k-nearest neighbor technique is used for matching descriptors. A positive matching is considered if the Euclidean distance to the second d_{k2} nearest neighbor is shorter than $0.7d_{\text{k1}}$ to the nearest neighbor [136].

6.3.8 Experimental results

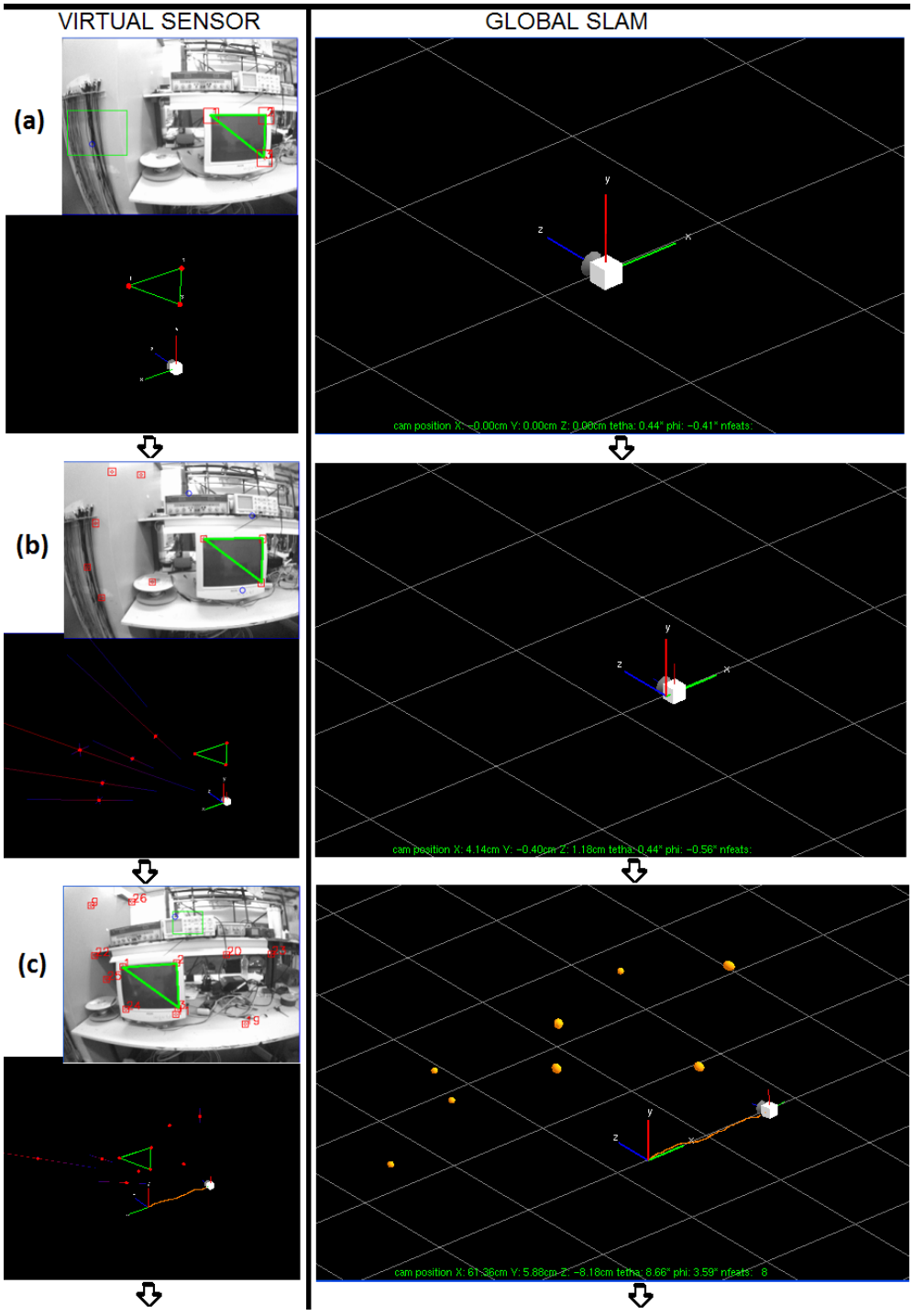
Figure 6.22 shows the results for the Distributed Monocular SLAM system. In this figure, the plots a, b , c, d, e, f, g and h illustrate the input, (upper-left) output (lower-left) of the Virtual Sensor (VS), and the output of the Global SLAM (GS), corresponding to respectively to frames 1, 60, 200, 560, 860, 1168, 1630 and 1700.

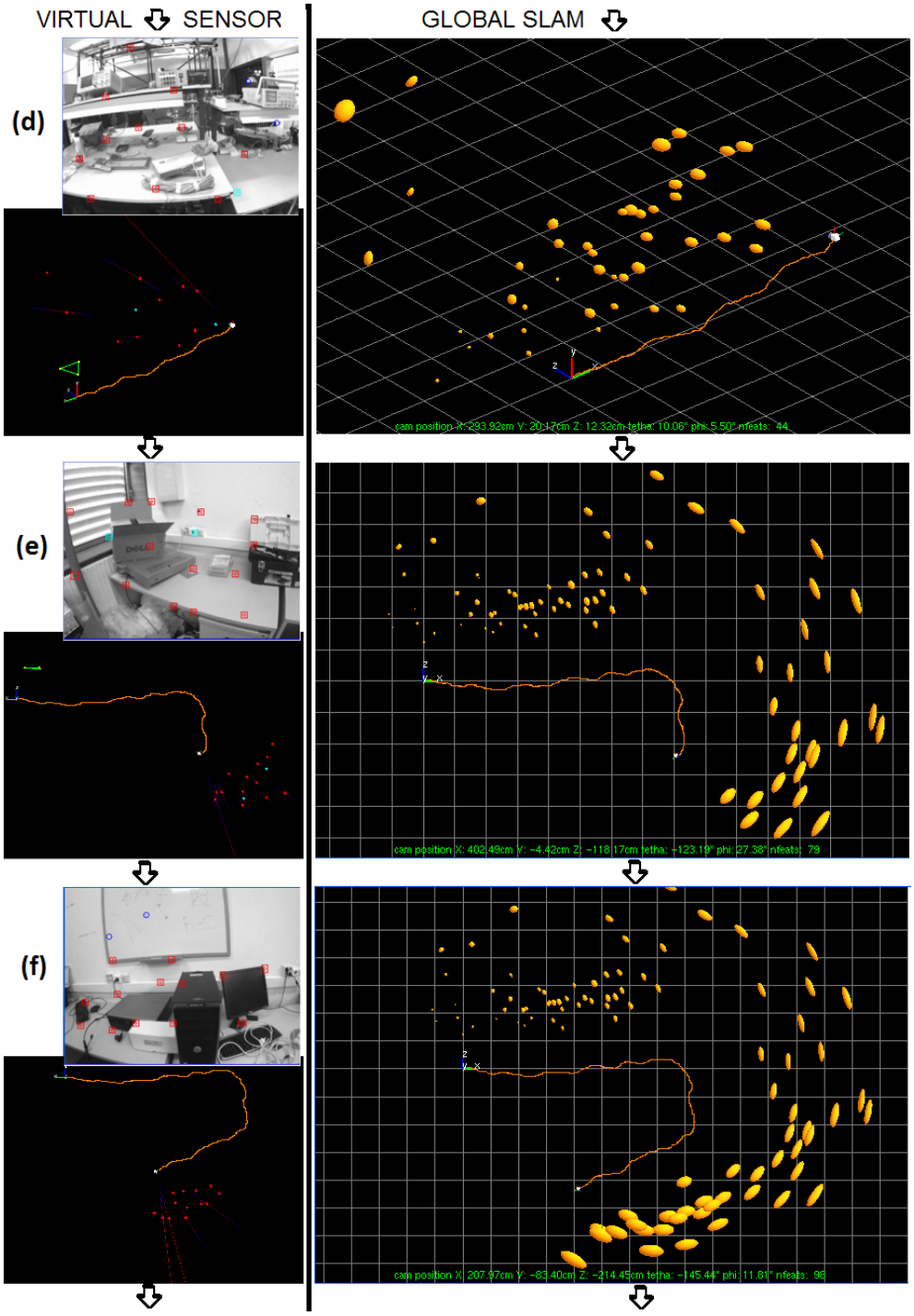
At frame 1 (plot a) the three-point metric reference has been already initialized. The three corners of the monitor (forming the metric reference) are the only features belonging to the map and the first candidate point has been detected. Note that the camera in both maps, VS and GS has been initialized at the origin of the coordinate frames. In plots a, b, c and d the map in both VS and GS graphics are showed from an X-Y-Z view (3D view). Also note that the screen of a PC monitor has been used as the only reference to recover the metric of the world (as others experiments). The grid, of the GS plots, measures 0.5x0.5 m.

At frame 60 (plot b), some features have been initialized in the VS, the GS has not yet received information about these features since their depth has not yet converged in the VS. At frame 200 (plot c) the firsts features have been initialized in the GS map. By the frame 560 (plot d) several features have been added to the GS map, but note that the VS has drop several features from the map for maintaining stable frequency operation.

Frame 860 (plot e) and frame 1168 (plot f) represents middle steps in the progression. In plots e, f, g and h the map in both VS and GS graphics are showed from an X-Z view (Top view).

At frame 1630 (plot g) there is a huge drift among the trajectory and the map. Note the difference between the estimated trajectory and the real one (In both VS and GS). In this case, it can be appreciated that rotations are the main cause of the error propagation (note the rotation at the second corner).


VIRTUAL SENSOR
GLOBAL SLAM
(a)
(b)
(c)
cam position X: -0.00cm Y: 0.00cm Z: 0.00cm tetha: 0.44° phi: -0.41° nfeats:
cam position X: 4.14cm Y: -0.40cm Z: 1.18cm tetha: 0.44° phi: -0.56° nfeats:
cam position X: 61.36cm Y: 5.88cm Z: -8.18cm tetha: 8.66° phi: 3.59° nfeats: 8


VIRTUAL SENSOR
GLOBAL SLAM
(d)
(e)
(f)

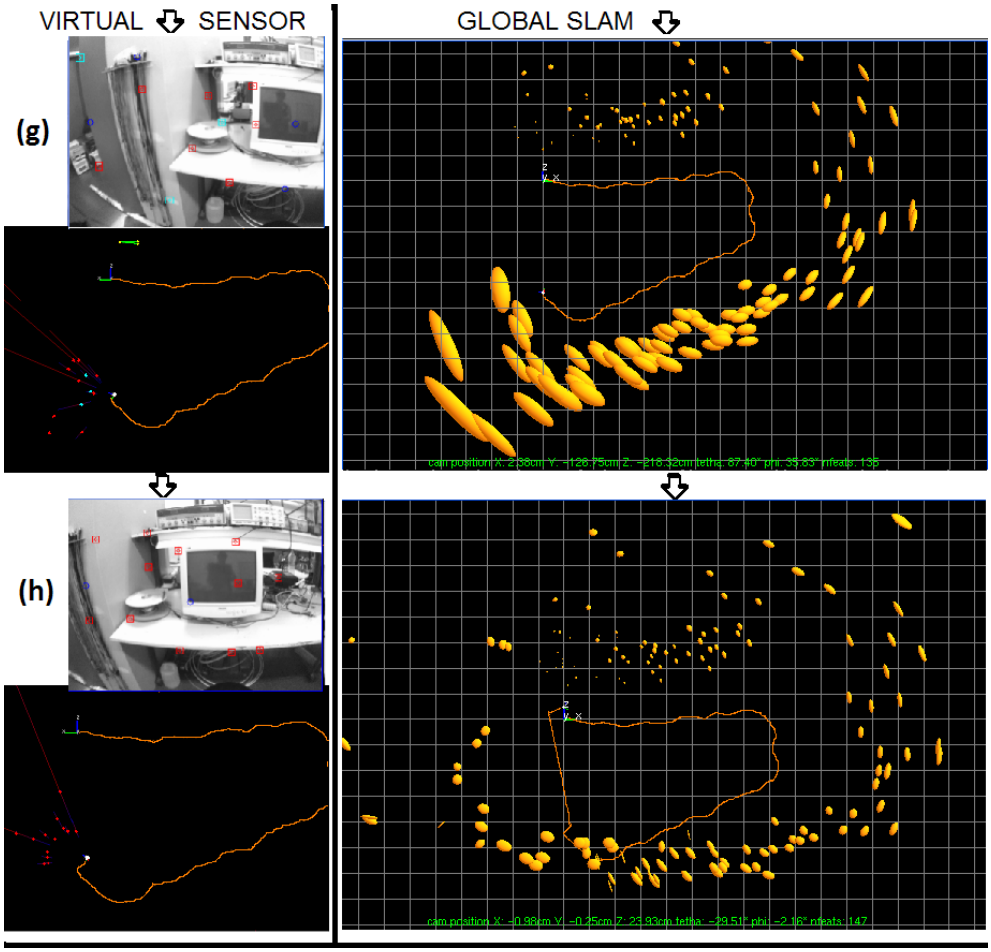


Figure 6.22 (displayed in three pages) Progression for the camera pose and map estimates for frames 1, 60, 200, 560, 860, 1168, 1630 and 1700 (plots a, b, c, d, e, g and h respectively) for both Virtual Sensor (VS) and Global SLAM (GS). Each plot illustrates the state of both VS and GS at one specific frame; the upper-left graphic is the input image for the VS (augmented), the lower-left graphic shows the VS map and trajectory estimates, and the right graphic shows the GS map and trajectory estimates. A video of 1720 frames (the same used in the experiment of section 6.3.5) has been captured following a predefined closed trajectory. Note that the Virtual Sensor (VS) map contains a stable number of features along the trajectory. At frame 1630 (plot g), it can be appreciated (X-Z view) the drift in the trajectory and the uncertainty propagation (represented by the ellipses) in the VS and GS features. At frame 1700, the Global SLAM has successfully detected some matches and the map has been correctly rebuilt and the camera pose has been set right (plot h). Also note how the features uncertainties have been minimized.

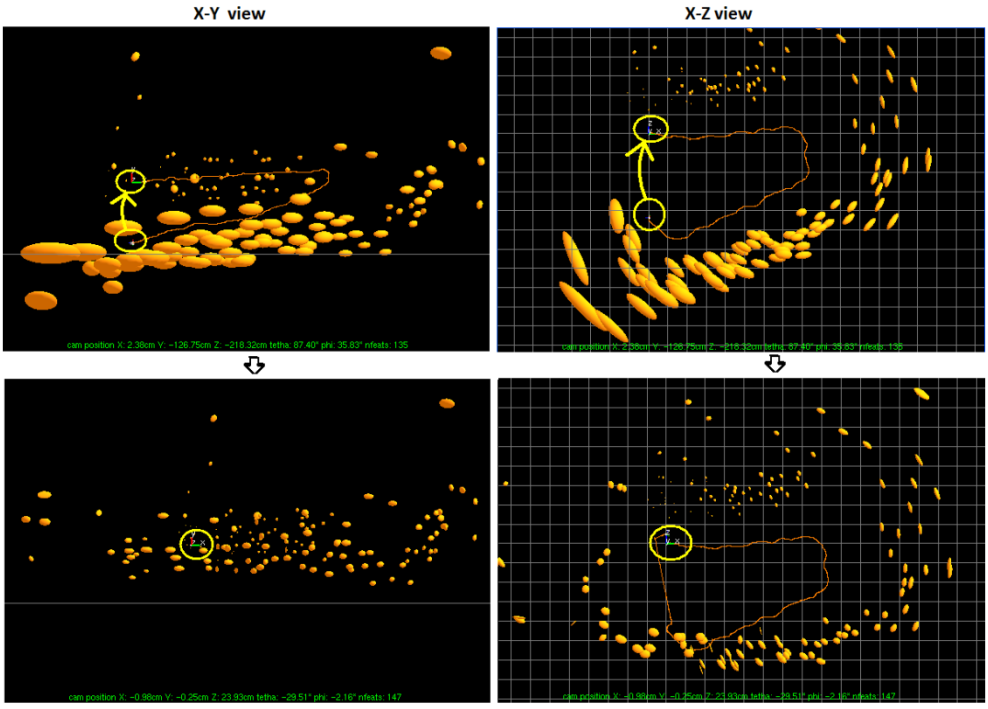


Figure 6.23 Detecting and closing the loop. In SLAM matching previously mapped features is the key in order to minimize drift. After a long travel and when an old feature has not been matched yet, the drift can be substantial; observe the drift at frame 1630 in both views X-Y and X-Z for the GS estimates (upper graphics). Fortunately, the GS can represent properly the propagation of uncertainty in both camera and features locations. This fact makes possible, when an old features is matched against a new measurement that the map and camera pose could be set right (lower graphics).

Also observe the uncertainty propagation for the measured features in the GS. (Compare the size of the first mapped features and the size of the last mapped ones).

At frame 1700 (plot h), the camera has returned nearby to its initial position and the GS has detected matches successfully and it closes the loop nicely. It can be observed how the map has been correctly built, the features uncertainty has been minimized, and the drift in the camera position has been fixed.

Figure 6.23 shows in two different views (X-Y and X-Z view) the moment when the loop is detected and closed. In section 6.3.1 we seen that for regular monocular SLAM methods, it often happen, that the propagation of uncertainty in both camera pose and map is not well represented by the estimation process, thus making difficult the loop detection-closure process. On the other hand, in the Distributed Scheme, the propagation

of uncertainty in the global map (Global SLAM) is well represented due to the Global SLAM process is decoupled from the Virtual Sensor process.

This later fact makes possible to close the loop in the global map when a match is found between the old mapped areas and the new measurements.

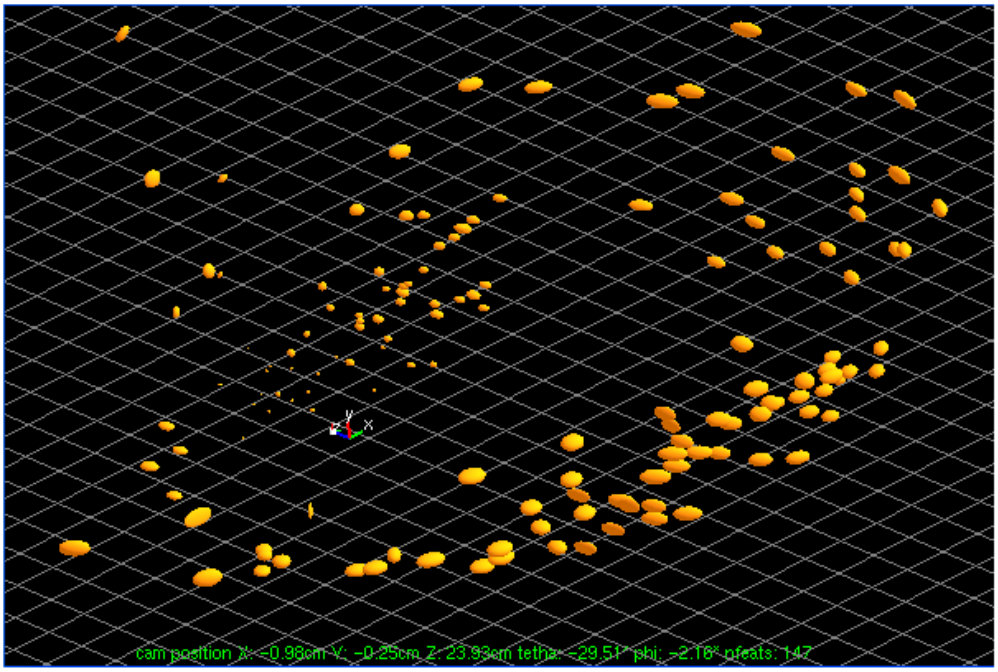


Figure 6.24, 3D view for the final map built by the Global SLAM (GS) sub-system.

At the end, about 360 features have been initialized in the Virtual Sensor (maintaining in the state an average of 20) on the one hand. On the other, 147 features have been initialized in the Global SLAM. These latter features contain a lot of useful information for data association (in the form of image descriptors). Figure 6.24 shows the 3D final map built by the Global SLAM subsystem.

One attribute of the Distributed approach is that it decouples the frequency operation of the input (camera) and the output (final camera position and map estimation). If only a single camera wants to be used as sensory input, a high frame rate operation is desirable. However, when the map grows maintaining the camera frequency operations, it becomes very difficult to keep the rate processing. In the distributed scheme, the operation frequency of the Global SLAM doesn't depend on the operation frequency of the Virtual Sensor. Therefore, if the Global SLAM map grows, it could be updated at slower rates in relation to the camera frame rate.

Monocular SLAM could profit from the use of different data association techniques, such as active search and image feature descriptors. However, there is a trade-off between the need of a high frame rate operation and the data association techniques required for addressing the whole problem. The proposed scheme handles these data association techniques naturally. The Virtual Sensor benefits from the use of the active search for matching features frame to frame, at high operation frequency. The process of matching features using image descriptors (useful for detecting loops after long camera trajectories or when there is a huge drift in the estimation) is not necessarily implemented with an active image search approach, because descriptors representing the current detected features can be matched with a data base of previously stored feature descriptors. This kind of matching approach is coherent with a general SLAM scenario (as the Global SLAM), where observations come from an abstract sensor, and the influx of observations doesn't depend on the estimation machinery.

6.4 Conclusions

In this chapter we address the problematic related to the 6-DOF monocular SLAM case, which possibly represents the harder variant of SLAM. First, it was explained how to modified a general Bearing-Only SLAM algorithm in order to be suited for a monocular camera context. In that sense the nonexistence of an odometry input and the ambiguity in scale are both important aspects to take into account.

It was seen that for most of the existing monocular SLAM implementations, there are two main challenges for addressing, in order to make possible that vision-based SLAM methods emulates (and ultimately surpass) the results in large-scale mapping achieved using laser range-finder sensors: Lack of robustness and Limited to room-sized domains.

Regarding to the lack of robustness issue, a novel method for delayed initialization of new features based on the inverse depth parametrization, is presented for monocular SLAM. The experimental results show that this method can be a good choice for implementing monocular SLAM systems. The method seems to be more robust respect to the un-delayed method, when initial metric reference points are used for scaling the map. In experiments the resulting camera trajectory estimate using the delayed method was similar to the estimate by the un-delayed method (when the un-delayed method was tuned). In aspects relating with features depth convergence the results were similar for both methods. Moreover, since the delayed method is more restrictive for adding new features, a reduced percentage of strong new features are added to the map (20-40%) respect to the un-delayed method, without losing the quality of the map.

In monocular SLAM, when the number of features in the system state increases, maintaining real time operation becomes very difficult and thus limiting to most of the existing of monocular SLAM methods to be applied to small environments. Moreover, it

has been seen that closing loops (minimizing drift in map and camera pose estimates) in real time, when camera's trajectory requires a vast number of natural landmarks in the map, could be a difficult task. In that sense, it is a clear trade-off between the need of a high frame rate operation and the data association techniques required for addressing the whole problem. In that sense a novel Distributed Framework, which handles in a natural manner the mentioned trade-off, is presented for addressing the problem of building and maintaining a global and consistent map of large environments at real time.

Our approach is to use a distributed framework where the key idea is to divide the whole task into two concurrent estimation processes. First a state of the art monocular SLAM method is modified as a complex virtual sensor that emulates typical sensors such as laser for range measurement and encoders for dead reckoning. Afterward, a classic SLAM method is plugged in for building and maintaining the final map.

In the experiments presented, real image sequences have been used. The C++ implementation runs at real time. Virtual Sensor and Global SLAM reside at different local network PCs communicated with the TCP/IP protocol. The experimental results are positive. In the example presented in this chapter, the map and camera trajectory estimated by the Virtual Sensor shows a huge drift, but it has been successfully minimized by the Global SLAM.

The modularity of the method allows other techniques for implementing both, Virtual Sensor and Global SLAM. The presented framework is also suitable for cooperative mapping contexts using more than one Virtual Sensor and a single Global SLAM connected by a network.

Chapter 7

Concluding Remarks

7.1 Conclusions

The importance of solving the Simultaneous Localization and Mapping (SLAM) problem in robotics can never be overstated. SLAM has been declared as the most fundamental problem that needs to be solved if we want to build truly autonomous mobile robots. SLAM is about a robot being able to concurrently track its position and construct a map of its surrounding environment. Solving each of the two problems independently is easier, i.e., a robot can trivially estimate its position given a complete map of its environment and it can also construct a map from sensor data if its position is accurately known. Solving both problems at the same time is a chicken and the egg type problem. So what makes SLAM so difficult and how are researchers trying to solve it?

Noisy sensors and actuators make SLAM a difficult problem. Every time the robot moves, it relies on its sensors to compute the distance traveled. Since there is no such thing as a noise-free sensor, there is always a small error associated with every move. These errors add up very quickly and the robot becomes lost. Correcting these errors is easy if a good map of the robot's operating environment is given; the robot can correct its estimated position by computing its relative position to landmarks that are present in the map and currently visible.

Statistical methods that have become popular in robotics give us powerful frameworks for handling uncertainty in sensor measurements and robot actions. These frameworks have allowed researchers to take large steps towards solving SLAM. For

example, the Extended Kalman Filter (EKF), which is an extension to the Kalman Filter (KF) for non-linear systems (which was used in this work) or the Monte Carlo methods such as Particle Filters (PFs).

The robot's sensors have a large impact on the algorithm used for SLAM. Early SLAM approaches focused on the use of range sensors as sonar rings or lasers. Nevertheless there are some disadvantages with the use of range sensors in SLAM: correspondence (data association) is difficult, they are expensive, and they are generally limited to 2D maps and computational overhead due to large number of features.

The aforementioned issues have propitiated that recent work is moving towards the use of cameras as the primary sensing modality. A camera is a projective sensor which has become more and more interesting for the robotic research community, because it yield a lot of information allowing reliable data association. Cameras are well adapted for embedded systems: they are light, cheap and power saving. Using vision, a robot can localize itself using common objects as landmarks.

On the other hand, at difference of range sensors (i.e. sonar or laser) which provides range and angular information, a camera is a projective sensor which measures the bearing of images features. Therefore depth information (range) cannot be obtained in a single frame. This fact has propitiated the emergence of a new family of SLAM methods: The Bearing-Only SLAM methods, which mainly relies in especial techniques for features system-initialization in order to enable the use of bearing sensors (i.e. cameras) in SLAM systems.

This thesis was focused on the study of the Bearing-Only SLAM problematic. Despite the fact that there has been a considerable progress in this area, currently there are two main problems with most existing monocular SLAM implementations: Lack of robustness and they are Limited to room-sized statics domains. In this work we present different methods and algorithms orient to contribute to the effort of make possible that bearing-based SLAM emulates (and ultimately surpass) the results in large-scale mapping achieved using laser range-finder sensors, aiming to build Bearing-Only SLAM systems with the potential of guiding autonomous robots in their exploration and operation in large and complex environments.

Several techniques have been proposed in literature for initializing features in Bearing-Only SLAM. Inverse Depth feature Parameterization increases the linearity of the measurement equation. In that sense, the Un-delayed Inverse Depth parameterization method has show to be an excellent option for Bearing-Only SLAM, because a standard EKF-SLAM framework can be straightforward applied to address the Bearing-Only SLAM problem. Nevertheless doing experiments we found that this kind of feature initialization has some drawbacks: It often happens, in the cases where there are near and distant landmarks that the estimated depth cannot converge to the real one, causing that the final map and vehicle trajectory are poorly estimated. It also often happens that some Kalman update steps the features depths become negatives causing even the divergence

of the filter. On the other hand, experimentally we found that making a manual tuning of the initial parameters (initial features depth and variance) the percentage of success increasing significantly. The last fact, motive us to propose a novel feature initialization technique called Delayed Inverse Depth Feature Initialization, where initial parameters are dynamically estimated priors to add a landmark as a new feature in the stochastic map. In Chapter 4 we introduce our features initialization method for Bearing-Only SLAM in a 2D context and assuming the availability of odometry.

In this case, we compared Un-delayed method with respect to Delayed method and the following contributions have been made:

- The robustness was increased. Making several simulations it was seen that the percentage of effectiveness (percentages of times a method makes congruent estimates of the map and robot location) was considerably increased using the Delayed Method.

- The percentage of features presenting negatives depths was reduced, since the Delayed Method realizes depths estimations priors to the state and covariance initialization phase and commonly, features are initialized very near to its ground truth with lower associated uncertainty.

Usually the Bearing-Only SLAM has been associated with the Vision-based SLAM systems, perhaps because Cameras are by far the most popular bearing sensor used in robotics. On the other hand the use of alternative sensorial capabilities as the hearing sense has been much less explored in SLAM. In this case, sound sources (artificial or natural) are used as landmarks for being included in the robot's map in order to localize it along the time.

Chapter 4 also focuses on the inclusion of the hearing sense in SLAM, attempting to localize (without a priori information of the sound source location) both, the robot position and the sound source, along the time while the robot is moving freely in its environment.

In this context, a real application for Bearing-Only SLAM based in our Delayed Initialization method is presented: A small differential drive robot capable of sense bearing information respect to an external sound source with modest angular acuity (±10°) is considered. This robot subsystem (called, Sound Sensor), capable of providing bearing information respect to one or several sound sources, is closely related (from the algorithmic point of view) with others bearing sensors (e.g. camera), in that sense a general Bearing-Only SLAM framework like the proposed for us, can be straightforward modified for working in a sound-based context. In this case, the following contributions have been made:

- It has been demonstrated that Bearing-Only SLAM systems, which are commonly based on cameras as primary sensors, can be straightforward extended for its application based on alternative bearing sensors.

- The algorithm simulations show that several sound sources improve the robustness and effectiveness of the method; nevertheless a real sound sensor capable of tracking and separating several sound sources requires a challenging implementation. In that sense, it has been demonstrated that a small mobile robot (with limited capabilities), capable of tracking a single sound source, can be used in order to test the SSLAM method with real data sensor.

- The experimental results show how the method is able to estimate the sound source position without prior knowledge of the environment; this information is subsequently used for estimating reasonably well the robot's trajectory. It can be appreciated how the error propagation of the encoders-based odometry is bounded. In that sense, these experimental results are very promising since the method is tested in a robot with very limited capabilities. The method could be applied in a robot equipped with a more complex sound sensor capable of measuring several natural audio sources.

One of the major motivations, in order to impulse the development of bearing-only methods, is related with the use of cameras as primary sensors. Cameras among other things, provides a huge amount of information useful for addressing the data association problem. In that sense, the correspondence problem, also known as the data association problem, is possibly the one of the hardest problem in robotic and also one of the most important problems to solve in SLAM. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world.

In chapter 5 several techniques are described for addressing the data association problem in a vision-based SLAM context. The followings are the findings and contributions stated in this chapter:

- We have described a method called ICAD for matching image features in a wide based-line using ICA descriptors. Basically the method shows good results, especially in terms of computational performance. Nevertheless the response is still a little sensitive respect to some changes like point of view. In essence, ICA descriptors represent an

intermediate alternative, having an intermediary performance in computational cost and robustness, respect to SIFT and SURF descriptors on the one hand and the Gray level patches on the other: ICA descriptors are more robust than Gray level patches but less than SIFT or SURF. ICA descriptors also have a lower computational cost than SIFT or SURF but a little more expensive than the Gray level patches.

- A study of the application of statistical methods (SVM and KNN) for capturing the variability of image feature appearance was presented. The experiments were performed over different variations of ICAD descriptors. The results are very promising, since they show that the proposed methodology increased the robustness of the descriptors. One characteristic of the proposed methodology is the modularity, so it can be extended for their application with descriptors like SIFT or SURF.

- It was seen that the application of SIFT or SURF descriptors to SLAM is not straightforward. In that sense, a simple but effective framework based on combining descriptors and saliency operators like the Harris corner detector was presented. The experiment show that a lower (but sufficient for SLAM) number, all correct, of matches were founded.

As computational power grows, an inexpensive camera can be used to perform range and appearance–based sensing simultaneously, by replacing typical sensors as laser and sonar rings for range measurement and encoders for dead reckoning. In this context, the 6-DOF monocular camera case (Monocular SLAM) possibly represents the harder variant of SLAM. Monocular SLAM is closely related to the structure-from-motion (SFM) problem of reconstructing scene geometry. SFM techniques have been used successfully for recovering camera position and scene structure. However, the SFM techniques coming from the vision community research have been formulated as off-line algorithms and required batch, simultaneous processing of all the images acquired in the sequence.

In chapter 6 we extend our general Bearing-Only SLAM algorithm, (introduced in chapter 4) for being used in a monocular SLAM context. This approach aims to contribute to the robustness of Monocular SLAM systems. In that sense, the experimental results show that this method can be a good choice for implementing monocular SLAM. In this case, we also compared Un-delayed method with respect to Delayed method and the following contributions have been made:

- When initial metric reference points are used for scaling the map, which is very important especially in a robotic context, the method seems to be more robust respect to the un-delayed method. Making several

experiments using a low cost hand-held camera, it was seen that the percentage of effectiveness was increased using the Delayed Method.

- In experiments the resulting camera trajectory estimate using the delayed method was similar to the estimate by the un-delayed method (when the un-delayed method was tuned). In aspects relating with features depth convergence the results were similar for both methods (Also tuning the un-delayed method).

- Since the delayed method is more restrictive for adding new features, lees but strong new features are added to the map (20-40% respect to the un-delayed method), without losing the quality of the map. This is a desirable aspect, because bigger environments can be mapped with the same number of features.

In monocular SLAM, when the number of features in the system state increases, maintaining real time operation becomes very difficult and thus limiting to most of the existing of monocular SLAM methods to be applied to small environments. Moreover, it has been seen that closing loops (minimizing drift in map and camera pose estimates), in real time, can be a difficult task, when camera's trajectory requires a vast number of natural landmarks in the map. In that sense, it is a clear trade-off between the need of a high frame rate operation and the data association techniques required for addressing the whole problem.

A novel Distributed Framework, called Distributed Monocular SLAM, which handles in a natural manner the mentioned trade-off, is also presented in chapter 6 for addressing the problem of building and maintaining a global and consistent map of large environments at real time.

Our approach is to use a distributed scheme, where the key idea is to divide the whole task into two concurrent estimation processes. First a state of the art monocular SLAM method (Called Virtual Sensor) is modified as a complex virtual sensor that emulates typical sensors such as laser for range measurement and encoders for dead reckoning. Afterward, a classic SLAM method (called Global SLAM) is plugged in for building and maintaining the final map.

The followings are contributions of this approach:

- In the past, a single estimation process (EKF, particle filter...) has been tried when solving the whole problem of single camera SLAM. With our distributed approach is proved that two different and concurrent processes can be used in order to address the monocular SLAM problem. Our C++ system implementation runs at real time and the Virtual Sensor

and Global SLAM reside in different local network PCs communicated with the TCP/IP protocol.

- A Distributed method is capable of closing loops after long trajectories. The experimental results are positive. In the example presented, the map and camera trajectory estimated by the Virtual Sensor shows a huge drift, but it has been successfully minimized by the Global SLAM.

- The modularity of the method allows other techniques for implementing both, Virtual Sensor and Global SLAM. In that sense the presented framework could be also suitable for cooperative mapping contexts using more than one Virtual Sensor and a single Global SLAM connected by a network.

7.2 Future Work and Discussion

The work covered in this thesis provides a number of areas of interest that may be worth further investigation. Among others, it may be interesting to consider the following lines for further research:

Bearing-Only SLAM based on alternative sensors.

As was mentioned before, cameras are by far the most popular bearing sensor and due to its underlying attributes have motivated the development of bearing-only SLAM methods. On the other hand the use of alternative bearing sensors has been much less explored for a SLAM context. Nevertheless, in this thesis, it has been demonstrated the viability of the inclusion of alternative bearing sensors in SLAM. In this case, a small mobile robot (with limited capabilities), capable of tracking a single sound source, was able to estimate the sound source position without prior knowledge of the environment and using this information in order to subsequently estimating reasonably well its movements. In that sense, there are a lot of potential for this kind of Bearing-Only systems. Two possible lines for further research could be:

- Simulations show that several sound sources improve the robustness and effectiveness of the SSLAM method, therefore instead of a simple sound sensor capable of detecting bearing from a single and artificial sound source, the method could be applied in a robot equipped with a more complex sound sensor capable of measuring and distinguish between several natural audio sources.

- Including the hearing sense in SLAM systems is a very promising topic for further research. On the other hand, approaches as the *Delayed Inverse Depth* SLAM method can be straightforward applied to SLAM systems based on other kind of bearing sensors. For example, it could be viable to develop a subsystem-sensor capable of detecting and tracking infrared light sources.

Data association techniques oriented to vision-based SLAM.

Cameras provide a huge amount of information useful for addressing the data association problem. Actually, this fact has motivated the raising of research in Bearing-Only SLAM. In this thesis it has been proposed different approaches of data association focusing on SLAM applications. Regarding to this theme a probable line for further research could be:

- It has been seen that learning the variability of image features using statistical methods can improve the performance of regular image-features-based data association techniques. In our experiments we apply statistical methods (SVM and KNN) for capturing the variability of image feature appearance over different variations of ICAD descriptors. On the other hand, one characteristic of the proposed methodology is the modularity, so it can be extended for their application with more robust state-of-the-art descriptors like SIFT or SURF and others statistical learning methods.

Monocular SLAM based on distributed systems.

Currently there is a huge amount of work to do in order to make possible that camera-based SLAM emulates (and ultimately surpass) the results in large-scale mapping achieved using laser range-finder sensors. In that sense the initial results for our Distributed SLAM system are very promising. However, is important to point out that since it is an early approach, a lot of work has to be done in order to fully justify the mathematical basis of the method. In fact, our Distributed SLAM was born as an engineering solution to a tradeoff present in monocular systems, more than a theoretical solution to the whole problem.

Actually, it could be said that our Distributed SLAM method is a little bit transgressor with some of the current paradigms in SLAM. For example a researcher founds our proposed method mathematically unsound due to decoupling of stochastic processes that are in reality strongly correlated “the transfer of information between the mono SLAM and the global SLAM is done in a way that violates the Kalman filtering hypothesis of uncorrelatedness between model and process noise”. He mentions that the

approach discards correlation information that is well known to play a key role in the consistency of the filter. Another researcher for example asks us “why the covariance of odometry is empirically tuned instead of obtained from the virtual sensor”.

In this regard, I agree on the one hand that more theoretical work has to be done in order to fully support the mathematics basis of the method; on the other hand we think that monocular SLAM is a very particular case of the general SLAM problem and could be treated in an alternative manner than traditional approaches. For example, in monocular SLAM methods, often happens after long camera trajectories that the uncertainty in the camera position (represented by the covariance matrix) is maintained stable over the whole sequence. However there always will be an implicit drift in estimations, among other things, due to the bias introduced to the system after several initializations of new features. Therefore, the covariance matrix does not reproduce the real error propagation, making practically impossible to detect and close loops (See section 6.3.1). Among other factors, this problem makes us think that monocular SLAM systems could be decoupled into different stochastic process. Hence the idea of considering a modified monocular method as a black box, capable to emulating typical sensors such as laser for range measurement and encoders for dead reckoning. Actually, if our Virtual Sensor were viewed as a real sensor, then our method does not violates the Kalman filtering hypothesis of uncorrelatedness between model and process noise; since the covariance representing the process noise in the virtual sensor is empirically tuned (like many of the real sensors) instead of obtained from the Virtual Sensor. A real sensor does not provide directly a measure of uncertainty; instead, uncertainty is statistically estimated comparing the output of the sensor and its ground truth.

Of course others researchers have given excellent comments regarding to the potential of the Distributed method, making possible for us to publish a couple of related papers in known publications.

Apart from work on the mathematical soundness of the method, two possible lines for further research could be:

- Due to the modularity of the method different techniques can be used for implementing both Virtual Sensor and Global SLAM. For example, a computationally more independent method respect to the number of features in the system can be used for implementing the Global SLAM subsystem.

- Cooperative monocular SLAM is a line of research very little explored yet. The Distributed monocular SLAM framework could be also suitable for a cooperative mapping contexts, for example using more than one Virtual Sensor and a single Global SLAM connected by a network.

Appendix A

Publications

The first experimental results regarding to the Sound-based SLAM system (SSLAM) as an application of the general Delayed Inverse-Depth Bearing-Only SLAM method (3DOF odometry-available context) was presented in:

- Rodrigo Munguia, Antoni Grau. (2008). **Single Sound Source SLAM.** *In 13th Iberoamerican Congress on Pattern Recognition CIARP*. Published in *LECTURE NOTES IN COMPUTER SCIENCE*, SPRINGER Press 5197 ( ): 70-77. ISSN: 0302-9743.

In addition to experiments presented in the paper above, among supplementary simulations of the Delayed Inverse-Depth features initialization method, a new set of experiments with our small mobile robot and the SSLAM method were presented in:

- Rodrigo Munguia, Antoni Grau,(2008). **Delayed Inverse-Depth Feature Initialization for Sound-Based SLAM.** *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation ETFA. Hamburg, Germany 2008*. ISBN: 978-1-4244-1505-2.

Concerning to the Data Association problematic, a novel method for extracting features descriptors from images called ICAD (based on Independent Component Analysis, ICA) was presented in the next paper:

- Rodrigo Munguia, Antoni Grau, Alberto Sanfeliu (2006). **Matching Images Features in a Wide Base Line with ICA Descriptors**. *Proceedings of the 18th IEEE International Conference on Pattern Recognition ICPR. Hong Kong 2006***.** IEEE Press, p. 159-163. ISBN/ISSN: 0-7695-2521-0.

Later, an early work on the use of statistical methods for capturing the variability of images features in order to increases the robustness of the matching process (using descriptors) when a video stream is available (i.e. vision-based SLAM) can be found in:

- Rodrigo Munguía, Antoni Grau. (2006). **Comparative Study of Statistical Methods for** *Image Feature Matching in a Wide Base Line.* Actes del AVR'06. ED. ESAII-IRI-IOC, p. 139-144. ISBN/ISSN: 84-7653-885-5.

The previous article was reworked and a new set of experiments were added in order to be published in:

- Rodrigo Munguia, Antoni Grau. (2006). **Learning Variability of Image Feature Appearance Using Statistical Methods.** *In 11th Iberoamerican Congress on Pattern Recognition CIARP.* Published in LECTURE NOTES IN COMPUTER SCIENCE, SPRINGER Press 4225 ( ): 716-725. ISSN: 0302-9743

Regarding to the monocular SLAM problematic, our first work presenting a novel method for initializing new features in monocular SLAM systems (6DOF, no-odometry context), "Delayed Inverse-Depth Feature Initialization " was presented in:

- Rodrigo Munguia, Antoni Grau (2007). **Delayed Feature Initialization for Inverse Depth Monocular SLAM.** *Proceedings of the 3rd* European *Conference on Mobile Robots* ECMR**,** Freiburg Germany 2007**.**

The previous paper was reworked in order to be published in:

- Rodrigo Munguia, Antoni Grau (2008) **Delayed Inverse Depth Monocular SLAM** *17th IFAC World Congress.* Seul, Corea 2008. ISBN: 978-3-902661-00-5.

Extra materials, as an extended survive on the subject, additional graphics and new set of experiments regarding to the Delayed Inverse-Depth monocular SLAM can be found in:

- Rodrigo Munguia, Antoni Grau,(2007). **Camera Localization and Mapping using Delayed Feature Initialization and Inverse Depth Parametrization.** *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation ETFA. Patras, Greece 2007*. IEEE Press, p. 981-988. ISBN: 1-4244-0826-1.

Our approach for adapting a monocular SLAM method as a visual-based odometry system by removing old features from the system state was presented in the paper below. In this paper our three points based technique, for establishing the scale of the map in monocular SLAM systems, was also presented.

- Rodrigo Munguia, Antoni Grau,(2007). **Monocular SLAM for Visual Odometry.** *Proceedings of the 4th IEEE International Symposium on Intelligent Signal Processing WISP. Alcala de Henares, Spain 2007*. IEEE Press, p. 443-448. ISBN: 1-4244-0829-6.

A long paper, summarizing all the results (previous four papers) regarding to the Delayed Inverse-Depth monocular SLAM method can be found in:

- Rodrigo Munguia, Antoni Grau (2007). **3D Visual Odometry based on Feature Maps**. *Proceedings of the 3rd* Congreso de *Computacion*, *Informatica*, *Biomedica y Electronica CONCIBE***,** Guadalajara México 2007. ISBN: 978-970-27-1243-5.

Early experimental results regarding to our Distributed framework for monocular SLAM systems, which aims to contribute to the goal of building and maintaining a global and consistent map of large environments at real time, was presented in:

- Rodrigo Munguia, Antoni Grau (2008) **Minimizing Drift in Monocular SLAM Real Time Systems** *IEEE International Symposium on Industrial Electronics ISIE*. Cambridge UK. 2008

Finally, all of the work concerning to the Distributed approach was reworked and extended in order to be accepted for its publication in the TIM journal:

- Rodrigo Munguia, Antoni Grau. (2008) **Closing Loops with a Virtual Sensor based on Monocular SLAM.** *To appear in the IEEE Journal on Transactions on Instruments and Measurements*.

Bibliography

[1] *Unified Inverse Depth Parametrization for Monocular SLAM.* **J.M.M. Montiel, Javier Civera, and Andrew J. Davison.** 2006. Robotics: Science and Systems Conference.

[2] *Sonar-based real-world mapping and navigation.* **Elfes., A.** 1987, IEEE Journal of Robotics and Automation.

[3] **Elfes, A.** *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* s.l. : PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[4] *Sensor fusion in certainty grids for mobile robots.* **Moravec, H. P.** 1988, AI Magazine.

[5] **S. Thrun, B. Wolfram, F. Dieter.** *Probabalistic Robotics.* s.l. : The MIT Press, 2005.

[6] *The vector field histogram – fast obstacle avoidance for mobile robots.* **J. Borenstein, Y. Koren.** 1991, IEEE Journal of Robotics and Automation.

[7] *The mobile robot Rhino.* **J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, S. Thrun.** 1995, AI Magazine.

[8] *Experiences with an interactive museum tour-guide robot.* **W. Burgard, A.B. Cremers, D. Fox, D. H¨ahnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun.** 1999, Artificial Intelligence.

[9] *Many robots make short work.* **D. Guzzoni, A. Cheyer, L. Julia, K. Konolige.** 1997, AI Magazine.

[10] *Probabilistic algorithms and the interactive museum tour-guide robot minerva.* **S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. H¨ahnel, C. Rosenberg, N. Roy.** 2000, International Journal of Robotics Research.

[11] *Place recognition in dynamic environments.* **B. Yamauchi, P. Langley.** 1997, Journal of Robotic Systems.

[12] *Position referencing and consistent world modeling for mobile robots.* **R. Chatila, J.P. Laumond.** 1985. IEEE International Conference on Robotics and Automation.

[13] *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations.* **B. Kuipers, Y.T. Byun.** 1991, Journal of Robotics and Autonomous Systems.

[14] **Choset, H.** *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph.* s.l. : PhD thesis, California Institute of Technology, 1996.

[15] *Sensor Based Planning: The Hierarhical Generalized Voronoi Graph.* **H. Choset, J.W. Burdick.** 1996. Workshop on Algorithmic Foundations of Robotics.

[16] *Error correction in mobile robot map learning.* **S. Engelson, D. McDermott.** 1992. IEEE International Conference on Robotics and Automation.

[17] *Topological mapping for mobile robots using a combination of sonar and vision sensing.* **D. Kortenkamp, T.Weymouth.** 1994. National Conference on Artificial Intelligence.

[18] *Learning to explore and build maps.* **D. Pierce, B. Kuipers.** 1994. National Conference on Artificial Intelligence.

[19] *Learning topological maps with weak local odometric information.* **H Shatkay, L. Kaelbling.** 1997. IJCAI.

[20] *Spatial learning for navigation in dynamic environments.* **B. Yamauchi, R. Beer.** 1996, IEEE Transactions on Systems, Man, and Cybernetics.

[21] *Robust world-modeling and navigation in a real world.* **Zimmer, U.R.** 1996, Neurocomputing.

[22] *Active visual navigation using non metric structure.* **P. Beardsley, I. Reid, A. Zisserman, D. Murray.** 1995. International Conference on Computer Vision.

[23] *Estimating uncertain spatial relationships in robotics.* **R. Smith, M. Self, P. Cheeseman,.** s.l. : Springer-Verlag, 1990, Autonomous Robot Vehicles.

[24] *On the representation of spatial uncertainty.* **R. Smith, P. Cheesman,.** 1987, Int. J. Robot. Res.,.

[25] *A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem.* **G. Dissanayake, H. Durrant-Whyte, T. Bailey.** 2000. Working notes of ICRA'2000Workshop W4: Mobile Robot Navigation and Mapping,.

[26] *A Bayesian algorithm for simultaneous localization and map building.* **H. Durrant-Whyte, S. Majumder, S. Thrun, M. de Battista, S. Scheding.** 2001. Symposium of Robotics Research.

[27] *A computationally efficient method for large-scale concurrent mapping and localization.* **J. Leonard, H. Feder.** 1999. International Symposium on Robotics Research.

[28] *A probabilistic approach to concurrent mapping and localization for mobile robots.* **S. Thrun, D. Fox, and W. Burgard.** 1998, Autonomous Robots, Vol. 5, págs. 253-271.

[29] **J.A. Castellanos, J.D. Tardos.** *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach.* Boston, MA. : KluwerAcademic Publishers, 2000.

[30] **M. Csorba.** *Simultaneous Localisation and Map Building, Ph.D. dissertation,.* 1997 : Univ. Oxford,.

[31] *A solution to the simultaneous localisation and mapping (SLAM) problem.* **G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, M. Csobra.** 2001, IEEE Transactions. Robotics. Automatomation.

[32] *Optimization of the simultaneous localization and map building algorithm for real time implementation.* **J. Guivant, E. Nebot.** 2001, IEEE Transaction of Robotic and Automation.

[33] **Newman, Paul Michael.** *EKF Based Navigation and SLAM.* s.l. : SLAM Summer School, 2006.

[34] *Autonomous underwater simultaneous localisation and map building.* **S.B. Williams, P. Newman, G. Dissanayake, Durrant-Whyte.** 2000. IEEE Int. Conf. Robot. Automat.

[35] *Online acquisition of compact volumetric maps with mobile robots.* **C. Martin, S. Thrun.** 2002. IEEE International Conference on Robotics and Automation.

[36] *EM, MCMC, and chain flipping for structure from motion with unknown correspondence.* **F. Dellaert, S.M. Seitz, C. Thorpe, S. Thrun.** 2000, Machine Learning.

[37] *Maximum likelihood from incomplete data via the EM algorithm.* **A.P. Dempster, A.N. Laird, D.B. Rubin.** 1977, Journal of the Royal Statistical Society.

[38] **G.J. McLachlan, T. Krishnan.** *The EM Algorithm and Extensions.* s.l. : Wiley Series in Probability and Statistics, 1997.

[39] *Visually realistic mapping of a planar environment with stereo.* **L. Iocchi, K. Konolige, M. Bajracharya.** 2000. International Symposium on Experimental Robotics.

[40] *Using EM to learn 3D models with mobile robots.* **Y. Liu, R. Emery, D. Chakrabarti, W. Burgard.** 2001. International Conference on Machine Learning.

[41] *Integration of range and image sensing for photorealistic 3D modeling.* **P.K. Allen, Ioannis Stamos.** 2000. IEEE International Conference on Robotics and Automation.

[42] *3D reconstruction of environments for virtual reconstruction.* **R. Bajcsy, G. Kamberova, Lucien Nocera.** 2000. IEEE Workshop on Applications of Computer Vision,.

[43] *Semiautomatic 3-D model extraction from uncalibrated 2-D camera views.* **S. Becker, M. Bove.** 1995. Symposium on Electronic Imaging,.

[44] *Three-dimensional reconstruction of points and lines with unknown correspondence across images.* **Y.-Q. Cheng, E. Riseman, X. Wang, R. Collins, A. Hanson.** 2000, International Journal of Computer Vision,.

[45] *Modeling and rendering architecture from photographs.* **P.E. Debevec, C.J. Taylor, and J. Malik.** 1996. International Conference on Computer Graphics and Interactive Techniques.

[46] *Sensor based creation of indoor virtual environment models.* **Boulanger., S.E. Hakim and P.** 1997. Internetional Conference on Virtual Systems and Multimedia.

[47] *Interactive construction of 3D models from panoramic mosaics.* **H. Shum, M. Han, R. Szeliski.** 1998. International Conference on Computer Vision and Pattern Recognition.

[48] **J. Borenstein, B. Everett, L. Feng.** *Navigating Mobile Robots: Systems and Techniques* . s.l. : Wellesley, 1996.

[49] **D. Kortenkamp, R.P. Bonasso, R. Murphy,.** *AI-based Mobile Robots: Case studies of successful robot systems.* s.l. : MIT Press., 1998.

[50] *Markov localization for mobile robots in dynamic environments.* **D. Fox, W. Burgard, S. Thrun.** 1999, Journal of Artificial Intelligence Research,.

[51] *A Stochastic Map for Uncertain Spatial Relationships.* **R. Smith, M. Self, and P. Cheeseman.** 1987. International Symposium of Robotics Research.

[52] *Uncertain geometry in robotics.* **Durrant-Whyte, H.F.** 1988, IEEE Trans.Robot. Automat., Vol. 4, págs. 23-31.

[53] *Building, registrating, and fusing noisy visual maps.* **N. Ayache, O. Faugeras,.** 1988, Int. J. Robot. Res.,.

[54] *World modeling and position estimation for a mobile robot using ultra-sonic ranging.* **Crowley, J.** 1989. IEEE Int. Conf. Robot. Automat.

[55] *Position referencing and consistent world modeling for mobile robots.* **R. Chatila, J.P. Laumond,.** 1985. IEEE Int. Conf. Robot.Automat.

[56] *Simultaneous map building and localisation for an autonomous mobile robot.* **J.J. Leonard, H.F. Durrant-Whyte,.** 1991. IEEE Int. Workshop Intell. Robots Syst.

[57] *Concurrent localization and map building for mobile robots using ultrasonic sensors.* **Renken, W.D.** 1993. IEEE Int. Workshop Intell. Robots Syst.

[58] *A new approach to simultaneous localisation and map building.* **M. Csorba, H.F. Durrant-Whyte.** 1996. SPIE Aerosense,.

[59] *A computational efficient method for large-scale concurrent mapping and localisation.* **J.J. Leonard, H.J.S. Feder,.** 2000. Robotics Research, The Ninth International Symposium.

[60] *Experiments in multisensor mobile robot localization and map building.* **J.A. Castellanos, J.M. Martnez, J. Neira, J.D. Tardós,.** 1998. IFAC Sym. Intell. Auton. Vehicles,.

[61] *Building a global map of the environment of a mobile robot: The importance of correlations.* **J.A. Castellanos, J.D. Tardós, G. Schmidt,.** 1997. IEEE Int. Conf. Robot. Automat.,.

[62] *Localization and map building using laser range sensors in outdoor applications.* **J. Guivant, E.M. Nebot, S. Baiker,.** 2000, J. Robot. Syst.

[63] *Feature-based mapping in real, large scale environments using an ultrasonic arra.* **K.S. Chong, L. Kleeman,.** 1999 : s.n., Int. J. Robot. Res.

[64] *Experimental comparison of techniques for localization and mapping using a bearing-only sensor.* **M. Deans, M. Hebert,.** 2000. Int. Symp. Experimental Robot.

[65] *Mobile Robot Localization and Map Building: A Multisensosr Fusion Approach.* **Tardos, J.A. Castellanos and J.D.** s.l. : Kluwer Academic Publishers, 1999.

[66] *A solution to the simultaneous localization and map building (slam) problem.* **M.G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Corba.** 2001, IEEE Transactions on RObotics and Automation, Vol. 17, págs. 229-241.

[67] *Robust mapping and localization in indoor environments using sonar data.* **J. Tardos, J. Neira, P. Newman, and J. Leonard.** 2002, International Journal of Robotics Research, Vol. 4.

[68] *Simultaneous localization and mapping with sparse extended information filters.* **S. Thrun, Y. Liu, D. Koller, A. Ng, Z.Ghaharamani, and H. Durrant-White.** 2004, International Journal of Robotics Research, Vol. 23, págs. 690-717.

[69] *Simultaneous localisation and mapping with sparse extended information filters.* **S. Thrun, Y. Liu, D. Koller, A. Ng, H. Durrant-Whyte,.** 2004, Int. Journal on Robotics.

[70] *Optimization of the simultaneous localization and map-building algorithm for real-time implementation.* **Nebot, J.E. Guivant and E.M.** 2001, IEEE Transactions on Robotic and Automatomation.

[71] *Data association in stochastic mapping using the joint compatibility test.* **J. Neira, J.D. Tardós,.** 2001, IEEE Transactions on Robotic and Automation.

[72] *A counter example to the theory of simultaneous localization and map building.* **S.J. Julier, J.K. Uhlmann.** 2001. IEEE International Conference on Robotics and Automation.

[73] *FastSLAm: Factored Solution to the Simultaneous Localization and Mapping Problem.* **M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit.** 2002. National Conference on Artificial Intelligence AAAI.

[74] *Bayesian map learning in dynamic environments.* **Murphy, K.** 1999. Advances in Neural Information Processing Systems.

[75] *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping.* **S. Thrun, W. Bugard, D. Fox,.** 2000. International Conference on Robotics and Automation.

[76] *Consistency of the FastSLAM algorithm.* **T. Bailey, J. Nieto, E. Nebot,.** 2006. IEEE International Conference on Robotics and Automation.

[77] *Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges.* **M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit.** 2003. Int. Joint Conf. Artif. Intell.

[78] *Simultaneous Localization and Mapping Tutorial I.* **Durrant, Tim Baley and Hugh.** 2006, IEEE Robotics & Automation Magazine.

[79] *Simultaneous Localization and Mapping SLAM II.* **Durrant, Tim Baley and Hugh.** 2006, IEEE Robotics & Automations Magazine.

[80] *Contour tracking bys stochastic propagation of conditional density.* **Blake, M. Isard and A.** 1996. 4th European Conference on Computer Vision,.

[81] **Lucas, G.W.** A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators. [En línea]
http://rossum.sourceforge.net/papers/DiffSteer/.

[82] **J. Borenstein, H. R. Everett, and L. Feng.** *Where Am I? - Systems and Methods for Mobile Robot Positioning.* 1996.

[83] *Experimental Odometry Calibration of the Mobile Robot Khepera II Based on the Least-Squares Technique.* **Antonelli, G. , Chiaverini, S.** 2005. IEEE International Conference on Robotics and Automation ICRA.

[84] *Auto-calibration of systematic odometry errors in mobile robots.* **Bak M., Larsen T. D., Andersen N. A.** 1999. Mobile robots. Conference .

[85] *Fast and Easy Systematic and Stochastic Odometry Calibration.* **Kelly, Alonzo.** 2004. International Conference on Robots and Systems IROS .

[86] **Latombe.** *Robot Motion Planning.* s.l. : Kluwer Academic Publishers, 1991.

[87] *Landmark-Based Robot Navigation.* **A. Lazanas, Latombe.** 1992. Tenth National Conference on AI.

[88] **Nourbakhsh., R. Siegwart and I.R.** *Introduction to Autonomous Mobile Robots.* London : The MIT Press, 2004.

[89] *Sequential Localisation and map-building for real-time computer vision and robotics.* **Andrew J. Davison, Nobuyuki Kita.** s.l. : Elsevier, 2001, Robotics and Autonomous Systems.

[90] **A. Kehagias, J.Djugash and S. Singh.** *Range-only SLAM with interpolated Range Data.* Robotics Institute, Carnegie Mallon University. 2006. Tech. CMU-RI-TR-06-26.

[91] *Pure Range-Only Sub-Sea SLAM.* **Leonard, P. Newman amd J.** 2003. IEEE International Conference on RObotics and Automation ICRA.

[92] *Range-only SLAm for robots operating cooperatively with sensor networks.* **Djugash, J. Singh, S. Kantor, G.** 2006. IEEE International Conference on Robotics and Automation, ICRA.

[93] *Mapping partially observable features from multiple uncertain vantage points.* **J. Leonard, R. Rokoski, P. Newman, and M. Bosse.** 2002, International Journal of RObotics Research.

[94] *A Pure Probabilistic Approach to Range-Only SLAM.* **J.L. Blanco, J. Gonzalez, J.A. Fernandez.** 2008. IEEE International Conference on Robotics and Automation, ICRA.

[95] **Vidal, Teresa A.** *Visual Navigation in Unknown Environments.* Institut de Robotica i Informatica Industrial, UPC. 2007. PHD thesis.

[96] *Preemptive RANSAC for live structure and motion estimation.* **Nister, David.** 2003. IEEE International Conference on Computer Vision ICCV.

[97] *Bundle adjustment, a modern synthesis.* **B. Triggs P. MacLauchlan, R. Hartley and A. Fitzgibbon.** s.l. : Springer-Verlag, 2000, Vision Algorithms: Theory & Practice.

[98] *Real-time 3-D motion and structure from point features: a front-end system for vision-based control and interaction.* **H. Jin, P. Favaro and S. Soatto.** 2000. IEEE international Conference on COmputer Vision and Pattern Recognition.

[99] *3D reconstruction of complex structures with bundle adjutmet: an incremental approach.* **Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd.** 2006. International Conference on Robotics and Automation.

[100] *SLAM via Variable Reduction from Constraint Maps.* **Konolige, Kurt.** 2005. IEEE. International Conference on Robotics and Automation, ICRA.

[101] *Recursive 3-d motion estimation from a monocular image sequence.* **T.J. Broida, S. Chandrasshekhar, and R. Chellappa.** 1990, IEEE Trans. on Aerospace and Electronic Systems, págs. 639-656.

[102] *Experimental comparison of techniques for localization and mapping using a bearings only sensor.* **Herbert, Matthew Deans and Martial.** 2000. International Symposium on Experimental Robotics, ISER.

[103] *Online motion estimation from image and inertial measurements.* **Singh, Dennis Strelow and Sanjiv.** 2003. Workshop on Integration of Vision and Inertial Sensors, INERVIS.

[104] *Constrained initialisation for Bearing-Only SLAM.* **Bailey, Tim.** 2003. IEEE International Conference on Robotics and Automation, ICRA.

[105] *Bearing-only SLAM in Indoor Enviroments.* **Dissanayake, N.M. Kwok and G.** 2003Bearing-only SLAM in Indoor Enviroments.

[106] *An effcient multiple hypothesis filter for bearing-only SLAM.* **Dissanyake, N.M. Kwok and G.** 2004. International Conference on Intelligent Robots and Systems, IROS.

[107] *Bearing-only SLAM Using a SPRT BAsed Gaussian Sum Filter.* **M.N Kwork, G. Dissanayake.** 2005. IEEE International Conference on Robotics and Automation, ICRA.

[108] *Bearing-only tracking using a set of range-parametrised extended kalman filters.* **Peach., N.** 1995, IEEE Proccedings on COntrol Theory Applications, Vol. 142, págs. 73-80.

[109] *Real-time simultaneous localisation and mapping with a single camera.* **Davison, A.J.** 2003. International Conference on Computer Vision, ICCV.

[110] *Real-Time 3D SLAM with Wide-Angle Vision.* **A.J. Davison, Yolanda Gonzalez and Nobuyuki Kita.** 2004. IFAC Symposium on Intelligent Autonomous Vehicles.

[111] *Exploiting Distinguishable Image Features in Robotics Mapping and Localization.* **Patric Jensfelt, John Folkesson, Danica Kragic, and henrik I. Christensen.** 2006. European Robotics Symposium, EUROS-06.

[112] *A visual front-end for simultaneous localization and mapping.* **L. Goncavles, E. di Bernardo, D. Benson, M. Svedman.** 2005. IEEE International Conference on Robotics and Automation, ICRA.

[113] *Un-delayed Initialization in Bearing only SLAM.* **Joan Sola, Michel devy, Andre Monin and Thomas Lemaire.** 2005. IEEE International Conference on Intelligent Robots and Systems, IROS.

[114] *A practical Bearing-Only SLAM algorithm.* **Thomas Lemaire, Simon Lacroix and Joan Solá.** 2005. IEEE International Conference on Intelligent Robots and Systems, IROS.

[115] *Scalable Monocular SLAM.* **Drummond, Ethan Eade and Tom.** 2006. IEEE Computer Vision and Pattern Recognition, CVPR.

[116] *Robust sound source localization using a microfone aaray ona a mobile robot.* **J. Valin, F. Michaud, J. Rouat, D. Letourneau.** 2003. International COnference on Intelligent Robots ans Systems IROS.

[117] *Auditory Robotic Tracking of Sound Sources using Hybrid Cross-Correlation and Recurrent Network.* **J. Murray, H. Erwin, S. Wermter.** 2005. International COnference on Intelligent Robots ans Systems IROS.

[118] *A Biomimetic Apparatus for Sound-source Localization.* **Amir Handzel, Sean Andersson, Martha Gebremichael, P.S. Krishnaprasad.** 2003. IEEE Conference on Decision and COntrol.

[119] *Robot phonotaxis with dynamic sound-source localization.* **Sean Andersson, Amir Handzel. Krishnaprasad.** 2004. IEEE International Conference on Robotics and Automation ICRA.

[120] *Algorithms for acoustic localization based on microphone array in service robotics.* **E. Mumolo, M. Nolich, G. Vercelli.** 2003, Robotics and Autonomous Systems, Vol. 42, págs. 69-88.

[121] *Automatic identification of sound source position employing neural networks and rough sets.* **Czyzewski, Andrzej.** 2003, Pattern Recognition Letters, Vol. 24, págs. 921-933.

[122] *Realtime sound source localization and separation ofr robot audition.* **K. Nakadai, H. Okuno, H. Kitano.** 2002. IEEE International Conference on Spoken Language Processing.

[123] *Advanced Sensorial System for an Acoustic LPS.* **J. Ureña, A. Hernandez, A. Jimenez, J.M. Villadangos, M. Mazo, J.C. Garcia, J.J. Garcia, F.J. Alvarez, C. de Marziani, M.C. Perez, J.A. Jimenez, A.R. Jimenez, F. Seco.** 2006, Journal of Microprocessors and Microsystems, pág. 9.

[124] **Thrun, S.** *Robotic Mapping.* s.l. : CMU-CS-02-111, 2000.

[125] *Landmark-based navigation and the adquisition of environmental models.* **E.M. Riseman, A.R. Beveridge, J.R. Kumar, H. Swahney.** 1997, In Visual Navigation: From Biological Systems to Unmanned Ground Vehicles, págs. 317-374.

[126] *A combined corner and edge detector.* **C. Harris, M. Stephens.** 1988. Alvey Vision COnference.

[127] *Indexing based on scale invariant interest points.* **K. Mikolajczyk, C. Schmid.** 2001. International Conference on Computer Vision.

[128] *A new class of corner finder.* **Smith, S.** 1992. British Machine Vision Conference.

[129] *Good features to track.* **J. Shi, C. Tomasi.** 1994. IEEE Conference on Vision and Pattern Recognition (CVPR).

[130] *Interest point detectors for visual slam.* **O. Martinez, A. Gil, M. Ballesta, O. Reinoso.** 2007. Conference of the Spanish Association for Artificial Intelligence (CAEPIA).

[131] *Qualitative image based localization in indoor enviroments.* **J. Kosecka, L. Zhou, P. Barber, Z. Duric.** 2003. IEEE Conference on COmputer Vision and Pattern Recognition.

[132] *Object recognition from local scal-invariant features.* **Lowe, D.G.** 1999. International Conference on Computer Vision.

[133] *Vision-based mobile robot localization and mapping using scale-invariant features.* **J. Little, S. Se, D. Lowe.** 2001. IEEE International Conference on Robotics and Automation (ICRA).

[134] *Global localization using distinctive visual features.* **J. Little, S. Se, D.Lowe.** 2002. IEEE International Conference on Intelligent Robots and Systems.

[135] *Vision-based SLAM using the rao-blackwellised particle filter.* **R. Sim, P. Elinas, M. Griffin, J. Little.** 2005. IJCAI Workshop on Reasoning with Uncertanty in Robotics.

[136] *SURF: Speeded up robuts features.* **Heber Bay, Tinne Tuytelaars, Luc Van Gool.** 2006. European Conference on Computer Vision.

[137] *An affine invariant interet point detector.* **K. Mikolajczyk, C. Shmid.** 2002. ECCV.

[138] *Matching Widely Views based on Affine Invariant Regions.* **T. Tuytelaars, L. Van Gool.** 2004, Computer Vision.

[139] *Viewpoint invariant texture matching and wide baseline stereo.* **F. Schalizky, A. Zisseman.** 2001. ICCV.

[140] *Multiple view features descriptors from image sequence via Kernel principal component analysis.* **J. Meltzer, H. Yang, R. Grupta, S. Soatto.** 2004. ECCV.

[141] *A performance evaluation of local descriptors.* **K. Mikolajczyk, C. Schmid.** 2005, Transactions on Pattern Analysis and Machine Intelligence.

[142] *SIFT, SURF and Seasons: Long-term Outdoor Localization Using Local Features.* **C. Valgren, A. Lilienthal.** 2007, European Conference on Mobile Robots (ECMR).

[143] *Independent component analysis, A new concept?* **Common, P.** 1994, Signal Processing, Vol. 36, págs. 237-314.

[144] **Jolliffe, I. T.** *Principal Component Analysis.* s.l. : Springer Verlag, 1986.

[145] *A fast fixed-point algorithm for independent component analysis.* **A. Hyvarinen, E. Oja.** 1997, Neural Computation.

[146] *An Iterative Image Registration Technique with an Application to Stereo Vision.* **D. Lucas, T. Kanade.** 1981. International Joint Conference on Artificial Intelligence.

[147] **C. Tomasi, T. Kanade.** *Detection and Tracking of Point Features.* Carnegie Mellon University. 1991. Technical Report. CMU-CS-91-132.

[148] **David Mount, Sunil Arya.** ANN: A library for Approximate Nearest Neighbor Searching. [En línea] 2006. www.cs.umd.edu/~mount/ANN/.

[149] *A training algorithm for optimal margin classifiers.* **B.E. Boser, I. Guyon, V. Vapnik.** 1992. Worshop on Computational Learning Theory.

[150] *Turbulence and the dynamics of coherent structures, Part 1.* **L. Sirovich, R. Oja.** 1987, Quartely of Applied Mathematics, Vol. 45, págs. 561-571.

[151] **Chih-Chung Chang, Chih-Jen Lin.** LIBSVM: a library for support vector machines. [En línea] 2001. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[152] *Good Image Features for Bearing-only SLAM.* **Xiang Wang, Hong Zhang.** 2006. International Conference on Intelligent Robots and Systems .

[153] *A Semi-Direct Approach to Structure from Motion.* **H. Jin, P. Favaro, S. Soatto.** 6, 2003, The visual Computer, Vol. 19, págs. 377-394.

[154] *Automatic Camera Recovery for CLosed or Open Image Sequences.* **A.W. Fitzgibbon, A. Zisserman.** 1998. European Conference on Computer Vision.

[155] *Robust 3-D Motion tracking from Stereo Images: A Model-Less Method.* **K. Yin, H. Kin, Siu Wong, M. Chang.** 3, 2008, Transactions on Intrumentation and Measurement, Vol. 57, págs. 622-630.

[156] *MonoSLAM: Real-Time Single Camera SLAM.* **Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, Olivier Stasse.** 2007, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, págs. 1052-1067.

[157] *Dimensionless Monocular SLAM.* **J. Civera, Andrew Davison, J. Montiel.** 2007. Iberian Conference of Pattern Recognition and Image Analysis.

[158] *Real-time SLAm Relocalisation.* **B. Williams, G Klein, I. Reid.** 2007. International Conference on COmputer Vision, ICCV.

[159] *Locally planar path features for real-time structure from motion.* **N. Molton, I. Reid, A. Davison.** 2004. British Machine Vision Conference.

[160] *Real-time and Robust Monocular SLAM using Predictive Multi-Resolution Descriptors.* **D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, A. Calway.** 2006. International Symposium on Visual Computing.

[161] *Real-Time Monocular SLAM with Straight Lines.* **P. Smith, I. Reid, A. Davison.** 2006. BMVC.

[162] *Inverse Depth to Depth Conversion for Monocular SLAM.* **J. Civera, A. Davison, J. Montiel.** 2007. IEEE International Conference on Robotics and Automation, ICRA.

[163] Quaternions and spatial rotation. *Wikipedia.* [En línea] http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation.

[164] *MFm: 3-D Motion from 2-D Motion Causally Integrated over Time.* **A. Chiuso, P. Favaro, H. Jin, S. Soatto.** 2000. European Conference on Computer Vision, ECCV.

[165] *Mobile Robot Localisation using Active Vision.* **A. Davison, D. Murray.** 1998. European Conference on Computer Vision.

[166] *Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks.* **Stephen Se, David Lowe, Jim Little.** 2002, International Journal of Robotics Research.

[167] *Distinctive Image Features from Scale-Invriant Keypoints.* **Lowe, D.** 2, 2004, International Journal of Computer Vision, Vol. 60, págs. 91-110.

[168] *Mapping Large Loops with a Single Hand-Held Camera.* **L. Clemente, A. Davison, Ian Reid, J. Neira, J. Tardos.** 2007. European Conference on COmputer Vision.

[169] *SIFT, SURF and Seasons: Long-trem Outdoor Localization Using Local Features.* **C. Vlargren, A. Lilienthal.** 2007. European Conference on Mobile Robots.

[170] *Simultaneous map building and localization for an autonomous mobile robot.* **Leonard, John J, Durrant-Whyte, Hugh F.** 1991. IEEE Int. Workshop on Intelligent Robots and Systems. págs. 1442-1447.