Universidade do Porto

Faculdade de Engenharia

Programa Doutoral em Engenharia Eletrotécnica e de Computadores

Cooperative Vehicle Localization in Networked Systems

Marcelo Borges Nogueira

Porto, Julho de 2013

Universidade do Porto
Faculdade de Engenharia
Departamento de Engenharia Eletrotécnica e de Computadores

COOPERATIVE VEHICLE LOCALIZATION IN NETWORKED SYSTEMS

Marcelo Borges Nogueira

Dissertação submetida ao Departamento de Engenharia Eletrotécnica e de Computadores do Faculdade de Engenharia da Universidade do Porto, como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências.

Supervisor: Prof. Dr. Fernando Manuel Ferreira Lobo Pereira

Porto, Julho de 2013

UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Aprovada em Julho de 2013 pela comissão examinadora, formada pelos seguintes membros:

Prof. Dr. Fernando Manuel Ferreira Lobo Pereira
Departamento de Engenharia Electrotécnica e de Computadores da Universidade do Porto

---

Prof. Dr. António Pedro Rodrigues Aguiar
Departamento de Engenharia Electrotécnica e de Computadores da Universidade do Porto

---

Prof. Dr. Rui Paulo Pinto da Rocha
Departamento de Ciência de Computadores da Universidade de Coimbra

---

Prof. Dr. Antonio Manuel Santos Pascoal
Dep. de Eng. Electrotécnica e de Computadores do Inst. Sup. Técnico da Univ. Técnica de Lisboa

"An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed ... would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes."

Pierre Simon Laplace

Abstract

This thesis concerns cooperative localization of Autonomous Underwater Vehicles (AUVs). Due to the strict communication and sensing constraints of the underwater environment, localization of AUVs remains an important challenge. We introduce an approach that contributes to the current state-of-the-art. We show that most of the existing methods do not apply to our problem. Then, we introduce two types of SLAM-like algorithms, respectively, based on Extended Kalman Filter and Particle Filter to solve the localization problem with a group of AUVs equipped with poor navigation systems in an unknown environment. Without the aid of a global positioning system, the systems error is not bounded, but can still be greatly reduced, as it is revealed by simulation and experimental results. If the involved AUVs can emerge once in a while to update their pose by GPS, we can bound the error. We also present a nonlinear observability analysis of the system.

The main goals of this thesis are:

- Design estimation methods for multi-vehicles that does not require a high bandwidth communication.
- Analyse how the convergence of the Simultaneous Localization and Mapping algorithms behaves with a dynamic environment/map in a Networked Control System.
- Optimize the performance of the system, while fulfilling requirements, by combining total observation and highly uncertain sensors with partial observation and less uncertain sensors.

Resumo

Esta tese trata do problema de localização cooperativa de veículos submarinos autônomos (AUVs). Devido às limitações de comunicação impostas pelo ambiente submarino, localização cooperativa para AUVs é uma tarefa difícil. Neste trabalho apresentamos o estado da arte nesta área e analisamos as soluções existentes para problemas relacionados. Mostramos que a maioria dos métodos existentes não podem ser aplicados ao nosso problema. Em seguida, propomos a construção de algoritmos, similares à localização e mapeamento simultâneos (SLAM), organizados em dois grupos, baseados no Filtro de Kalman Estendido e no Filtro de Partículas, para resolver o problema de localização de um grupo de veículos submarinos, equipados com instrumentos de navegação de baixo custo, operando em um ambiente desconhecido. Sem o auxílio de um sistema de posicionamento global, o erro do sistema cresce sem limites. Contudo, é possível reduzi-lo bastante, como mostrado em resultados de simulações e experimentos. Caso os AUVs envolvidos tenham a possibilidade de emergir de tempos em tempos de forma a adquirir sinal de GPS e atualizar sua posição, é possível limitar o erro do sistema. Também apresentamos um estudo não linear da observabilidade do sistema.

Os principais objetivos desta tese são:

- Desenvolver métodos de estimação para múltiplos veículos que não exijam uma comunicação de banda larga.
- Analisar como se comporta a convergência do SLAM com um mapa dinâmico em um sistema de controle via rede (NCS).
- Otimizar a performance do sistema, satisfazendo suas restrições e requerimentos, combinando observações totais de sensores com incertezas altas com observações parciais de sensores mais precisos.

Acknowledgements

I would like to express my gratitude to some people that helped me through out the journey to finish this work. In first place, to my wife, Meika, who accompanied me to live in a foreign country. She made me feel at home. To my parents, Roussel and Angela, my sister, Ethel, and my mother and father in law, Mrs Akemi and Mr Abelardo, whom even being far away, were always "around", giving their support.

To Prof. Fernando, for the supervision and cheerful conversations. To Sujit, for all the advices and contributions. To Prof. Paulo Lopes, for a great idea. To LSTS people and Joao for their help, collaboration and friendship and to Eng. Paulo Lopes for being so helpfull at any time.

I also would like to thank my friends/new family that I have made in Porto: Adriano and Beth (nice trips), Alessandro, Ava, Eduardo (my wingman at parties), Fefe, Filipe and Katha (thanks for introducing me into the world of beer), Heryka, Karla (she taught me how to cook rabbit), Lucival and Cynthia (gave me a glimpse of how a family works), Mathias (Mr. Picanha), Marilia, Maria, Nuno (thanks for the hospitality and for showing so much of Portugal), Orlando and Andrea (thank you very much for eating my burned beans without much complaint), Thiago, Thiagão, Veve, Rose, my "oldest son" Romulo (without his help I would not be able to handle this work), and Will (for some proof reading).

To ALBAN, for the financial help. To FEUP and Electrical and Computer Engineering Department, for the opportunity.

Contents

List of Figures

List of Tables

Acronyms

AUV Autonomous Underwater Vehicle

CAUV Central Autonomous Underwater Vehicle

CCFAUV Current Communicating FAUV

CEKOL Cooperative Localization using EKF based on Localization

CEKOLM Cooperative Localization using EKF based on Localization and Mapping

CNA Communication and Navigation Aid-AUV

CPF Cooperative Localization using PF

DCEKOLM Decentralized CEKOLM

EKF Extended Kalman Filter

FAUV Feature Autonomous Underwater Vehicle

GPS Global Positioning System

INS Inertial Navigation System

KF Kalman Filter

LBL Long Baseline

NCS Networked Control System

pdf Probability Density Function

PF Particle Filter

RFID Radio Frequency Identification

REKF Regular EKF

SEKF SLAM EKF

SLAM Simultaneous Localization and Mapping

SIS Sequential Importance Sampling

SIR Sampling Importance Resampling

TOF Time of Flight

Chapter 1

Introduction

This thesis concerns the problem of cooperative navigation for Autonomous Underwater Vehicles (AUV). Localization is a core issue in navigation and a particularly formidable challenge in the underwater domain. To address it, we use concepts of Simultaneous Localization and Mapping (SLAM). Typical localization requirements imply the need to combine exteroceptive and proprioceptive data. The integrative nature of the physical devices providing the interoceptive data implies that uncertainty increases monotonically with time. On the other hand, exteroceptive data depends both on the wealth of features exhibited by the environment and the associated propagation of sensed data for passive features, and on the available communications capabilities for active features. Localization in the underwater milieu is, in general, a particularly challenging problem due to the difficulty in extracting exteroceptive data: the environment is, in general, relatively poor in features, and both communication and propagation of sensing data is difficult. Electromagnetic propagation is limited to extremely short distances and acoustic propagation exhibits poor reliability, very small bandwidth, and low propagation speed. These difficulties and the current technological limitations are pointing the underwater robotics towards the use of multi-vehicle system. We investigate the problem of how to overcome these obstacles in order to ensure reasonable localization requirements

by controlling the motion of multiple AUVs, some of which play the role of features. The determination of the localization of each one of these controlled mobile features depends on the respective interoceptive data gathered in certain time intervals, on the knowledge of their localization at the beginning of each one of these time intervals, on the data shared among the AUVs, and, of course, on their relative controlled motions. Thus, multiple AUV systems encompass important challenges. One of these consists in handling the severe constraints on sensing and on communications which emerge from the strict limitations on available power, communication bandwidth, reliability, and delays. Another important challenge concerns results on coordination and control theory to support the system's design enabling to strike a desirable trade-off between exploration and exploitation activities that optimizes the overall system's resources. The cooperative localization schemes conceived in this thesis use SLAM techniques in order to coordinate the motion of the AUVs subject to the above mentioned constraints so that the desired relative localization requirements are achieved. The associated research effort also encompasses the investigation of the following issues which were motivated by the extraction of guidelines for the design of multiple AUVs based systems: spatial distribution optimizing the cooperative localization quality, nonlinear observability, and the convergence of the proposed SLAM based cooperative schemes. The conclusions are illustrated with simulation results as well as with data obtained from experiments.

In order to better characterize the addressed challenges, we present some pertinent general considerations about navigation and SLAM in the first subsection, (1.1). Subsection 1.2 details the objectives of the thesis, and outlines the methodology used to achieve those objectives. A brief description of the main contributions is given in subsection 1.3. Finally, in subsection 1.4, we present the organization of the thesis.

1.1 General Considerations

Robotic systems can, by and large, be considered as organized in essentially two groups - manipulators and free moving vehicles, also known as mobile robots, [21], - as well as a combination of these two groups. All of them are able to move around their working space, but usually mobile robots are not constrained to fixed points in a predetermined working space, and, thus, are endowed with much wider mobility possibilities. In order to move around the environment, the robot must navigate. Navigation is one of the main areas of robotics, which addresses the processes enabling a robot to move from an initial set to a final set in the space of configurations - also known as the space of poses, i.e., position and orientation - of a given environment, possibly filled with obstacles, in an as efficient as possible manner while avoiding collisions with obstacles. One must point out that the large versatility of the associated motion planning and control problems enables the consideration of extremely elaborated navigation schemes which may also involve constraints on linear and/or angular velocities and accelerations. However, in this thesis, we will not consider such problems in order to keep the main addressed challenges simple to grasp. In order to navigate, the robot might, in general, require the following capabilities:

- Determine where it is (its current pose with respect to a global reference frame), which is called Localization. Given the fact that measurements are plagued with noise and models with uncertainties, it is understood here and in what follows, that the term "determine" has to be made precise by specifying a characterization of the uncertainty. An upper bound on the uncertainty of the robot's pose is one of the typical requirements specified for navigation.

- Identify and localize the features in its surroundings pertinent to motion relatively to its position with a certain level of precision, called Mapping. Obviously, if a sufficiently precise map is available *a priori*, then navigation does not requires mapping. However, if this is not the case or the environment is variable, then the navigation task must also build the required map. This can be viewed as a virtual reconstruction of the environment using sensor data which incorporates information relevant to the robot motion or any other of its activities.
- Plan a route from the initial to the final configuration while satisfying all the initial, intermediate and final constraints. This is known as Path Planning.

The first two problems are interrelated, since: (i) knowing its absolute pose with high accuracy is essential for the robot to construct its surrounding environment by placing the observed features of the world into an absolute map; and (ii) if an accurate description of the environment exists, then the robot can localize itself by matching its observations with the mapped features of the world, [75]. When there is no a priori map of the environment and the robot does not have the capacity to localize itself with the required level of precision, both problems - localization and mapping - will have to be jointly solved. That is, as the robot moves around, it improves the reconstruction of the environment map, and, as the map is being built, the robot uses it together with its interoceptive data in order to improve its localization. This process is called Simultaneous Localization and Mapping, or SLAM.

In order to be able to observe the features of the environment as well as its internal data (state), a robot needs be equipped with sensors. There are two kinds of sensors used for navigation: (i) “dead reckoning" or proprioceptive sensors (odometers, gyros, accelerometers, etc.), supplying measurements from the robot’s internal variables; and (ii) external or exteroceptive sensors (GPS, Vision, etc.), providing

the robot with measurements enabling the characterization of the features of the world. Of course, it is possible to use only proprioceptive sensors to attempt to solve the localization problem. However, independently of the quality of the used sensors, the uncertainty in the measured variables grows without bound due to the integrative nature of these sensors, and, thus, the obtained estimates usually do not meet the localization uncertainty requirements. Therefore, estimates must be periodically reset by using information from external sensors with adequately low uncertainty, like, for example, range and/or bearing measurements to known features in the world, or Global Positioning System (GPS) data. For air or ground vehicles, for example, GPS data with differential corrections (DGPS) can provide very precise and inexpensive measurements of geodesic coordinates, [34].

From the above, it is evident that the underwater environment poses particularly difficult challenges in the field of Navigation, and the motivation to address such challenging problems is clear.

An increasingly important class of problems for which the challenges addressed in this thesis are of utmost importance concerns the ocean exploration and the management of its resources. The ocean is largely unknown, and this, together with the serious sustainability challenges that human kind is facing, has been making it the subject of an increased attention in the past few decades. Approximately 72% of the Earth's surface (an area of some 361 million square kilometers) is covered by ocean. The study of the oceans is intimately linked to understanding global climate changes, potential global warming and related biosphere concerns, [106]. The future of humanity is deeply related to the bodies of water of the planet, and to the maintenance of their bio-diversity. This is why researchers are actively studying this important source of life – water – in all of its forms, [23]. Nevertheless, the Ocean is a hazardous environment for humans. Most of the oceanic environment has remained unexplored in part because of our own physiological limitations. Placing

people into the marine environment has always been a dangerous endeavour. One way to explore the oceans and avoid the dangers to humans is to use uninhabited - either remotely operated or autonomous - vehicles, enabling the human operator to stay in less hazardous locations. Even so, the scope of manned submersibles and remotely operated vehicles is limited to a few applications because of very high operational costs, operator fatigue and safety issues. That is why the demand for advanced underwater robot technologies is growing fast, [110]. AUVs are unmanned and untethered submarines that provide marine researchers with a long-range and low-cost solution with which to gather oceanographic data. Common applications for deploying AUVs include, to name just a few, oceanographic sampling, bathymetric profiling, underwater systems inspection, and military mine counter measure (MCM) operations, [90]. In many of these missions, it is critical that the vehicle position is known precisely and in real time.

While submerged, the vehicle cannot use GPS data and, typically, can not rely on optical instruments - cameras and laser range finders - in order to reset the estimates of its navigation data. By surfacing, the AUV is able to use GPS data to update his position. However, surfacing frequently in order to maintain the quality of the navigation data is not a desirable motion or behavioral constraint for most of the applications of AUV systems as this implies a poor usage of valuable resources. Thus, for most of applications, range sensors - based on the acoustic time of flight measurements to landmarks - are often the only external sensors available for navigation purposes.

One way to use range measurements for localization is to employ static beacons in known positions to form a Long Baseline (LBL) configuration that will enable the estimation of the vehicle's position by means of triangulation. However, this approach naturally limits *a priori* the operation area, [34], and this is an unacceptable constraint for many mission scenarios for which the geographical bounds can

not be delimited beforehand. To overcame this, instead of static beacons, a group of AUVs can cooperate to provide a localization service supporting the execution of the assigned mission. One way to achieve this consists in each one of a subset or all of the vehicles play the role of landmark for the others, thus creating a Moving LBL, [98]. The difficulty of this approach is that, now, landmarks have some uncertainty associated with their position, and this data has to be shared among at least some of the vehicles so that it can be taken into account in the computation of the navigation data estimates by each one of the vehicles. Thus, communication between the involved vehicles is required. There are some methods in the literature ([79], [77]) for land and aerial vehicles that combine this information from all the vehicles and use filtering techniques to estimate their pose. This combination of information requires a substantial communication among the vehicles. However, communication between AUVs using the the current state-of-the-art underwater acoustic modems feature very low bandwidth and very high unreliability. This severely limits the data that can be shared among the vehicles. Alexander Bahr, [11], proposed a method for cooperative localization of AUVs that does not require high bandwidth. His approach makes use of a class of AUVs capable of highly accurate self localization. However, as will be discussed in Chapter 7, as the uncertainty of the system (uncertainty both in vehicle position and range measurement) increases, his approach degenerates and does not yield good results.

Because of the low bandwidth in underwater communication, the amount of data shared by vehicles to help estimate their pose should be as low as possible. In estimation theory, there are basically two types of methods: the ones that make some assumptions about the *a posteriori* distribution, and the ones that consider no particular shape for the uncertainty distribution. In contexts in which random variables are assumed Gaussian, estimation is provided by the Kalman Filter for linear systems or by the Extended Kalman Filter for nonlinear systems. In this

case, the probabilistic distribution is fully characterized by two parameters - mean (vector) and covariance (matrix) - and it is not difficult for a vehicle to share its data with the others. However, it might not be practical or even possible for all the vehicles to share all the data all the time. So, we have to come up with a method that uses information by a subset of vehicles, and not all of them. There is still another difficulty with the EKF approach: it is known that it can fail to provide optimal estimates for highly nonlinear systems, [16].

On the other hand, the Particle Filter (PF) approach is suitable for nonlinear systems, since it does not require any assumption on the uncertainty distribution. The distribution is not parameterized, but described as a set of several points that represent the desired distribution. However, this brings an even greater problem: even considering just one vehicle, it is impossible to send all this data to another AUV. So, how can this information be shared with the other vehicles? There is still the problem that the vehicles must communicate with each other, thus forming a Networked System, and the communication difficulties arising in sustaining the network directly affects the problem of control and estimation. Thus, by studying the SLAM problem with a dynamic map, we investigate estimation problems in Networked Systems.

Once the cooperative localization problem is successfully solved, then one may address the vehicles formation control problem. A key question is: what should be the relative position between the vehicles to better achieve their goal? We investigate topologies in the formation control that bring advantages to the estimation process.

In the SLAM scenario, the issue of how to settle the trade off between exploration and exploitation, i.e., the execution of the mission main goal becomes inescapable. In order to execute the goal of the mission, the robot must have information about the working environment. If the main goal is not just mapping, then how should the resources of the robot be allocated between exploring the environment by revisiting

landmarks to create a good map, and executing the missions main goal? If the map is not sufficiently good, thus causing poor localization, the robot may not be able to accomplish the main goal satisfactorily. By the other hand, there is no point in the robots spending most of their resources creating a high quality map, leaving the main goal as a secondary task that may not even be achieved. This idea can be generalized for a much larger set of autonomous robotic systems.

1.2 Objectives and Approaches

The main objective of this thesis is to develop methods based on the EKF and PF to solve, in real time, the localization problem with a group of AUVs equipped with possibly poor navigation systems and limited sensing and communication capabilities in an unknown, often feature-poor, environment. The method must be robust to the process and measurement noises and a critical point is that it must take into account the communication constraints of the underwater environment. That is, the approach must consider a limited bandwidth and communication failures, such as data packets loss and transmission delays. Our approach relies on the investigation of SLAM schemes for dynamic maps in a Networked Control System. Moreover, since in our framework, some of the features are controlled AUVs, the generation of available information depends on the relative position of which the vehicles are controlled. Thus, the problem of defining the control of an autonomous system by combining components that (i) optimize the performance in pursuing its goal with the available information and (ii) gather more information about the environment to optimize the overall performance in pursuing its goal, can be regarded as a general abstract paradigm of main objective of the thesis.

To accomplish this task, we consider the following sub-goals:

- Cooperative localization methods based on the EKF and PF with low bandwidth communication and analyse its convergence properties when considering

perfect and faulty communications.

- Vehicles' spatial formation control topologies and communications protocols in order to optimize the developed estimation methods.
- Architectures for autonomous systems that must consider the trade-off between the efforts for cooperative estimation and actuation (fulfillment of the mission's primary goal) by analyzing formation control and communications topologies.

In order to achieve the proposed goals, we will use methods of analysis, filtering and estimation, notably the KF and the PF, control theory and optimization methods.

1.3 Contribution

The main contribution of this thesis includes a set of algorithms to perform the relative localization of multi-vehicle systems with limitations on communications capabilities. These algorithms make use of the Extended Kalman Filter and can be divided in two groups:

1. Algorithms based on Localization - centralized and decentralized versions.
2. Algorithms based on Localization and Mapping - centralized and decentralized versions.

These algorithms are compared to the current state-of-the-art and shown to have key advantages such as being able to work properly and robustly subject to slow and faulty communications. The algorithms are also shown to have better performance and able to keep a consistent cross-correlation matrix (algorithms based on Localization and Mapping with Extended Kalman Filter) which provides the correct level of confidence of relative positioning. In order to prove their efficacy, we performed an observability analysis of the proposed algorithms. We also presented an

optimization method to determine the spacial distribution of the group of vehicles in order to give the best possible relative positioning.

Although less informative than global positioning, relative positioning suffices to meet the localization requirements of a wide range of applications. In this thesis, we present as an example of such application, plume tracing achieved via an extremum seeking navigation scheme physically realized by a team of AUVs performing the simplex algorithm. Since the vehicles have some associated uncertainty on their position, we conducted a study of the convergence properties of simplex method with uncertain vertices.

The contributions of this thesis represents a step forward towards the systematization of the design of multiple vehicle systems with emphasis in ocean observation, monitoring surveillance, and other applications for which cooperative navigation of underwater vehicles contributes to the desired requirements.

1.4 Organization

This thesis is organized as follows. In Chapter 2, the estimation problem is introduced by considering two classical solutions - the Kalman Filter and the Particle Filter -, and the issues concerning their formulation for Networked Control Systems, notably the strong dependence on communication, are discussed. In Chapter 3, the SLAM problem is presented and shown how the estimation methods addressed in the previous chapter can be used to solve it. In Chapter 4, the benefits of cooperative navigation are discussed, the detailed formulation of the problem is presented and some of the key existing approaches to solve it are addressed. Also in this chapter, we will argue why the existing methods are not appropriate for the proposed cooperative AUV localization scenario - a group of AUVs equipped with all with the same kind of dead reckoning and external sensors - and present in detail the main goals of the thesis. Chapter 5 presents and discusses the solutions found for the stated

cooperative localization problem, and includes details of the various developed algorithms. Chapter 6 deals with optimizations techniques for the proposed algorithms, and shows a typical real application scenario in which the proposed algorithms are of interest. The proposed solutions are compared with the closest approach found in the literature, as shown in Chapter 7, by using both simulations and experimental data. In Chapter 8, we present an observability analysis of the system in what concerns thee exteroceptive information, and, in the light of this analysis, we discuss the convergence of the proposed cooperative navigation algorithms. Finally, some conclusions are drawn and future work outlined in Chapter 9.

Chapter 2

Preliminary Concepts

In this Chapter we will present, for the sake of completeness, some concepts that will play a key role in the developments of this thesis. In section 2.1 we present the Estimation problem and discuss two classical solutions: the Kalman Filter, presented in section 2.1.1, and the Particle Filter, discussed in section 2.1.3. In Section 2.2 we discuss some aspects of networked control systems.

2.1 Estimation Theory

Estimation is the process by which we infer the value of a quantity of interest, x, by processing data that, in some way, depends on x, [66]. Consider a discrete time system modeled by the Equation (2.1), where $x(k) \in \mathbb{R}^n$ (or x_k) is the state vector, $u(k) \in \mathbb{R}^m$ is the control signal at time instant k and $y(k) \in \mathbb{R}^o$ is the output measurement, or observation. Assuming that the current state $x(k)$ condenses all the information about the past of the system, if we know the initial system state $x(0) = x_0$, it is possible to use the equations to predict exactly the state of the system at any future time instant.

$$\begin{aligned} x(k+1) &= f(x(k), u(k))) \\ y(k) &= h(x(k), u(k)) \end{aligned} \tag{2.1}$$

Now consider that the systems is contaminated by some noise, as shown in Equation (2.2), where $q(k) \in \mathbb{R}^p$ and $r(k) \in \mathbb{R}^o$ are the process and the measurement noises respectively.

$$\begin{aligned} x(k+1) &= f(x(k), u(k), q(k)) \\ y(k) &= h(x(k), u(k)) + r(k) \end{aligned} \tag{2.2}$$

Since there are some noise sources in Equation (2.2) that cannot be measured, even if we knew with high accuracy the initial sate of the system $x(0)$, we can not tell for certain the evolution of the system's state. We expect to see this uncertainty in input transformed into uncertainty in the output. However, we can say what is the most likely state of the system (expected value of x, $E\{x\}$) or what is the probability that the system is in a given state at some moment. We can achieve this by computing the probability density function (pdf) of the system's state. In this case, we can use the information given by the observations, $y(k)$, to improve the estimate of the pdf of the system's state at a given time instant, that is, the pdf of x conditioned on the observations. This is represented by the Equation $p(x_k|Y^T)$, where $Y^T = y(1:k) = y_{1:k}$ represents all the observations made until to time k and x_k represents the state of the system at time instant k.

From the Bayes rule, it follows that

$$p(x_k|Y^T) = \frac{p(x_k, Y^T)}{p(Y^T)}, \tag{2.3}$$

and also that

$$p(Y^T|x_k) = \frac{p(Y^T, x_k)}{p(x_k)}. \tag{2.4}$$

From Equations (2.3) and (2.4) we can conclude that

$$p(x_k|Y^T)p(Y^T) = p(Y^T|x_k)p(x_k). \tag{2.5}$$

Now, we will derive the Bayesian recursive state estimator. Assume that the observations at the current time are independent of past states, that is

$$p(Y^T|x_k) = p(Y^{T-1}|x_k)p(y_k|x_k),$$

where y_k is a single observation at time instant k. By substituting into 2.5, we get

$$p(x_k|Y^T)p(Y^T) = p(Y^{T-1}|x_k)p(y_k|x_k)p(x_k). \tag{2.6}$$

By again using the Bayes rule for the term $p(Y^{T-1}|x_k)$ and substituting in Equation (2.6) we obtain

$$\begin{aligned} p(x_k|Y^T)p(Y^T) &= \frac{p(x_k|Y^{T-1})p(Y^{T-1})}{p(x_k)}p(y_k|x_k)p(x_k), \\ &= p(x_k|Y^{T-1})p(Y^{T-1})p(y_k|x_k), \\ \text{and} \quad \text{thus} \quad p(x_k|Y^T) &= \frac{p(x_k|Y^{T-1})p(Y^{T-1})p(y_k|x_k)}{p(Y^T)}. \end{aligned} \tag{2.7}$$

By using once again the Bayes rule

$$\begin{aligned} p(y_k|Y^{T-1}) &= \frac{p(y_k, Y^{T-1})}{p(Y^{T-1})}, \\ &= \frac{p(Y^T)}{p(Y^{T-1})}, \\ \text{and} \quad \text{thus} \quad \frac{1}{p(y_k|Y^{T-1})} &= \frac{p(Y^{T-1})}{p(Y^T)}. \end{aligned} \tag{2.8}$$

By substituting this expression in Equation (2.7), we finally get

$$p(x_k|Y^T) = \frac{p(y_k|x_k)p(x_k|Y^{T-1})}{p(y_k|Y^{T-1})}. \tag{2.9}$$

The term $p(x_k|Y^{T-1})$ can be computed by the formula (Theorem of total probability)

$$p(x_k|Y^{T-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y^{T-1})dx_{k-1}. \tag{2.10}$$

Equations (2.9) and (2.10) constitute the recursive Bayesian estimator. Since the denominator of (2.9) does not depend on x, it can be considered just a normalization factor, η. Further simplification of these equations is possible if the Markov assumption holds, that is, the state x at time k is independent of past measurements other then y_{k-1}. This gives us

$$p(x_k|y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{k-1})dx_{k-1}, \tag{2.11}$$

$$p(x_k|y_k) = \frac{p(y_k|x_k)p(x_k|y_{k-1})}{\eta}, \tag{2.12}$$

where η is the normalization factor (this term can be easily calculated after the first step, so that the probability of two complementary events add up to the unity or by using the theorem of total probability, [94]). Terms $p(x_k|x_{k-1})$ and $p(y_k|x_k)$ are derived from the system transition model and observation model, respectively. Together with the initial estimated state (and its associated uncertainty), Equations (2.11) and (2.12) define a recursive estimator for the state of a partially observable system, i.e., systems where the measurements does not fully determine the state, [95]. Notice that there are no assumptions on the form of the distribution $p(x_k|y_k)$. However, implementing the exact solution to the Bayes estimator is not trivial, specially when one is concerned with efficiency. There are several techniques to solve it, and, in the next sections, we will introduce two classical solutions, namely, the Kalman Filter (and, for nonlinear systems, the Extended Kalman filter) and the Particle Filter.

Estimation theory has a wide application in tracking and navigation. In robotics, estimation is often used to perform localization and/or mapping. We will also show how these solutions can be applied specifically to solve the SLAM problem.

2.1.1 Kalman Filter

The Kalman Filter (KF) combines system's noisy measurements over time in order to obtain state estimates that are more precise than estimates based on a single measurement, [55]. It is a recursive procedure developed by Rudolf Kalman in [42], yielding optimal estimate values for linear systems contaminated with Gaussian white noise.

Consider a discrete system modeled by the Equation (2.13), where $x(k) \in \mathbb{R}^n$ is the state vector (robot pose - position and orientation - for example), $u(k) \in \mathbb{R}^m$ is the control signal at time instant k. $y(k) \in \mathbb{R}^o$ is the output measurement, or observation, $q(k) \in \mathbb{R}^n$ and $r(k) \in \mathbb{R}^o$ are the process and the measurement noises (both Gaussian, with zero mean and uncorrelated), respectively, and $A(k) \in \mathbb{R}^{n \times n}$, $B(k) \in \mathbb{R}^{n \times m}$, $C(k) \in \mathbb{R}^{o \times n}$ are the system transition, control and observation matrices, respectively.

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + q(k) \\ y(k) &= C(k)x(k) + r(k) \end{aligned} \tag{2.13}$$

Since the system (2.13) is linear and the process noise is Gaussian, the pdf of x, $p(x)$, is also going to have a Gaussian distribution, [35]. This kind of distribution, in the one-dimensional case, can be fully described by its expected value $E\{x\}$ and its variance σ^2. In the multivariate case, where $x = [x_1\, x_2 \cdots x_n]^T$, it is described by the expected value $E\{x\}$ and its covariance matrix $P = E\{(x - E\{x\})(x - E\{x\})^T\}$.

Without any observation, the expected value and covariance matrix can be propagated using only the system's model. This is called prediction. The propagation

of the predicted expected value of x, $\hat{x} = E\{x\}$, is given by:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + q(k), \\ E\{x(k+1)\} &= E\{A(k)x(k) + B(k)u(k) + q(k)\}, \\ \hat{x}(k+1) &= A(k)E\{x(k)\} + B(k)u(k) + E\{q(k)\}, \\ &= A(k)\hat{x}(k) + B(k)u(k), \end{aligned} \tag{2.14}$$

since $q(k)$ is a Gaussian noise with zero mean, that is, $E\{q(k)\} = 0$.

Similarly, we use the system's model to propagate the covariance matrix:

$$\begin{aligned} P(k) &= E\{[x(k) - E\{x(k)\}][x(k) - E\{x(k)\}]^T\} \\ &= A(k-1)E\{[x(k-1) - E\{x(k-1)\}] \\ &\qquad [x(k-1) - E\{x(k-1)\}]^T\}A(k-1)^T \\ &\quad + E\{q(k-1)q(k-1)^T\} \\ &= A(k-1)P(k-1)A(k-1)^T + Q(k-1). \end{aligned} \tag{2.15}$$

If there is an observation available, we can use it to improve our prediction, that is, we can find a new estimated value and covariance matrix that are more accurate than the ones obtained by the prediction phase. The optimal linear estimate based on the observation $y(k)$ is a linear function of the *a priori* estimate (predicted from Equation (2.14), and from now on, represented by $\hat{x}_p$), and the measurement $y(k)$, [35], that is:

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}_p(k+1) + K(k)\left(y(k) - C\hat{x}_p(k)\right), \\ &= A(k)\hat{x}_p(k) + B(k)u(k) + K(k)\left(y(k) - C(k)\hat{x}_p(k)\right). \end{aligned}$$

The optimal value of $K(k)$ that minimizes the square error of the estimated state, that is, $E\{(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T\}$, is given by, [94],

$$K(k) = P_p(k)C(k)^T\left(C(k)P_p(k)C(k)^T + R(k)\right)^{-1}, \tag{2.16}$$

where $P_p(k)$ is the predicted covariance matrix at time instant k, given by Equation (2.15).

Summarizing, the KF equations, [43], are given by

$$\begin{aligned} e(k) &= y(k) - y_p(k), \\ L(k) &= C(k)P_p(k)C^T(k) + R(k). \end{aligned}$$

Prediciton:

$$\begin{aligned} \hat{x}_p(k+1) &= A(k)\hat{x}(k) + B(k)u(k), \\ P_p(k+1) &= A(k)P(k)A^T(k) + Q(k), \\ \hat{y}_p(k+1) &= C(k)\hat{x}_p(k+1). \end{aligned} \tag{2.17}$$

Update:

$$\begin{aligned} K(k) &= P_p(k)C^T(k)L(k)^{-1}, \\ \hat{x}(k) &= \hat{x}_p(k) + K(k)e(k), \\ P(k) &= P_p(k) - P_p(k)C^T(k)L^{-1}(k)C(k)P_p(k), \end{aligned}$$

where $e(k)$ is called the innovation, $K(k)$ is the Kalman gain, $\hat{x}_p(k)$ and $\hat{x}(k)$ are the predicted and the filtered state estimates, $y(k)$ and $\hat{y}_p(k)$ are the observation and predicted observation estimate, and $P(k)$ and $P_p(k)$ are, respectively, the filtered and the predicted error covariance matrix. By assuming that $q(k)$ and $r(k)$ are uncorrelated, matrices $Q(k)$, and $R(k)$ are given by

$$E\left\{ \begin{bmatrix} q(k) \\ r(k) \end{bmatrix} \begin{bmatrix} q^T(k) & r^T(k) \end{bmatrix} \right\} = \begin{bmatrix} Q(k) & 0 \\ 0 & R(k) \end{bmatrix}.$$

The initial conditions $x_p(0)$ and $P_p(0)$ are needed for the algorithm to work. The update of the current estimate takes place only if an observation is available. Otherwise, we put $\hat{x}(k) = \hat{x}_p(k)$ and $P(k) = P_p(k)$. Notice that the KF equations give optimal solutions only for linear systems.

The KF is a widely used technique to solve localization problem. If a robot is moving in a given region and a map of this environment (known features or

landmarks) is available, then measurements (also called observations) like range and/or bearing (those are actually nonlinear measurements, and in the next Section we will see how to use the KF with nonlinear models) to the detected features can be used to feed a KF to improve the current estimate of the configuration (pose and orientation) of the robot. In a 2D world representation, this component of the state of the robot is given by the vector $X = [x,\, y,\, \theta]^T$ (in this case we use a capital letter to differentiate the whole state vector X from its x coordinate component). Conversely, in a mapping problem, the robot uses observations to determine the location of features on the environment relatively to the robot. If the robot knows its pose, this information can be used to determine the location of the landmarks in the environment. In this case, the state vector would be composed by the landmarks parameters, $X = [x_1,\, x_2 \cdots x_n]$, where x_i are the parameters of the i^{th} feature.

2.1.2 Extended Kalman Filter

We saw in section 2.1.1 that the KF is an estimation technique developed for linear systems. However, in the real world, most of the systems have nonlinear model. So, how can the KF be applied to these nonlinear models?

For non-linear systems, state vector in the next time instant is a nonlinear function of the current state vector and of the control signal, and the observation can also be a non linear function of the state variable, as shown in Equation (2.18), one can use the Extended Kalman Filter (EKF).

$$\begin{aligned} x(k+1) &= f(x(k), u(k), q(k), k), \\ y(k) &= h(x(k), u(k), k) + r(k). \end{aligned} \tag{2.18}$$

The EKF uses linearizations of the functions f, and h as shown in Equation (2.19) below. Here, $D_x f$, $D_q f$, and $D_x h$ denote the Jacobians of f with respect to

x, and to q, and of h with respect to x, respectively. The EKF equations are, thus, as follows:

$$\begin{aligned} e(k) &= y(k) - h(x(k), u(k), k), \\ L(k) &= D_x h P_p(k) D_x h^T + R(k). \end{aligned}$$

Prediciton:

$$\begin{aligned} \hat{x}_p(k+1) &= f(x(k), u(k), k), \\ P_p(k+1) &= D_x f P(k) D_x f^T + D_q f Q(k) D_q f^T. \end{aligned} \tag{2.19}$$

Update:

$$\begin{aligned} K(k) &= P_p(k) D_x h^T L(k)^{-1}, \\ \hat{x}(k) &= \hat{x}_p(k) + K(k) e(k), \\ P(k) &= P_p(k) - K(k) L(k)^T. \end{aligned}$$

For simplicity, the arguments $x(k)$, and $u(k)$ of f, and of h have been omitted.

The Jacobian $D_x f$ of a function$f : R^n \to R^m$ with respect to x can be computed as follows:

$$D_x f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \tag{2.20}$$

It is a common practice to consider that the control signal is corrupted by noise, $u(k) = u_n(k) + q(k)$, where $u_n(k)$, is a nominal control, and $q(k)$ is the input noise. In this case, we have $D_q f = D_u f$, the Jacobian of$f$ with respect to u, [66].

The EKF does not give the optimal solution, since it approximates the nonlinear system by its linearization around the current estimate, nevertheless it still exhibits reasonable performance in many important classes of systems.

Obviously, for the KF and the EKF to work properly (and, as a matter of fact, for any other estimation technique), measurements must yield sufficient information about the state of the system. If it is possible to completely determine the state of

system given the control signals $u(t)$, and the measurements $y(t)$, then the system is said to be observable, [6]. It is not necessary, however, that the measurements fully determine the state x. The Kalman Filter might be designed to provide estimates with partial measurements. However, if the system is not observable, the uncertainty associated with the unobservable component of the state will grow without bounds.

2.1.3 Particle Filter

In the previous section, we saw how to solve the the Bayes Filter estimator assuming that the pdf of x, $p(x_k|Y^T)$, is Gaussian. However, this assumption might not hold for some systems. There is class of methods to solve efficiently the Bayes filter which do not require the specification of the pdf of x, which might not even be analytical. They are called Monte Carlo methods. Monte Carlo methods are a class of algorithms that rely on repeated random sampling in order to generate estimates that are used when it is not possible to compute an exact result with a deterministic algorithm, [105]. Among these, the Particle Filter (PF) methods, also known as Sequential Monte Carlo methods, are the most popular.

The key idea of the PF methods consists in using a discrete approximation of $p(x_k|Y^T)$. We represent $p(x_k|Y^T)$ by a set of m weighted samples, $\{x_k^i, w_k^i\}_{i=1\cdots m}$, distributed according to $p(x_k|Y^T)$, as shown in Equation (2.21), where x_k^i represents the i^{th} particle, w_k^i is its weight or importance, and δ is the Dirac delta function. The weights are normalized so that $\sum_{i=1}^{m} w^i = 1$.

$$p(x_k|Y^T) \approx \hat{p}(x_k|Y^T) = \sum_{i=1}^{m} w^i \delta(x_k - x_k^i) \tag{2.21}$$

As the number of samples, also called particles, increase, the approximating pdf $\hat{p}(x_k|Y^T)$ becomes closer to the true optimal Bayesian estimate $p(x_k|Y^T)$, [17]. The weights of the particles w^i are chosen via a general technique called Importance Sampling.

Importance Sampling solves the problem of estimating properties of a probability density $p(x)$ using samples generated by another distribution. The basic idea is to associate with each particle a weight in addition to its position, [44]. Suppose that we have a known, possibly different, probability density function $\pi(x)$, designated importance density, from which we can easily draw samples x^i. In this scenario, we can write, [44]:

$$\hat{p}(x_k|Y^T) = \sum_{i=1}^{m} \frac{p(x_k^i)}{\pi(x_k^i)} \delta(x_k - x_k^i). \tag{2.22}$$

By choosing an appropriate function $\pi(x)$ such that it can be decomposed according to

$$\begin{aligned} \pi(x_{0:k}|y_{1:k}) &= \pi(x_{0:k-1}|y_{1:k-1})\pi(x_k|x_{0:k-1}, y_{1:k}), \\ &= \pi(x_{0:k-1}|y_{1:k-1})\pi(x_k|x_{k-1}, y_k), \end{aligned} \tag{2.23}$$

we derive the update equations for the weights (below, $w_k^i = w^i(k)$) :

$$w_k^i = w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{k-1}^i, y_k)}. \tag{2.24}$$

The performance of PF methods is crucially dependent on the choice of the Importance Density Function, [3]. The optimal importance density function choice, π_{opt}, is the one that keeps the weights of the particles as uniform as possible (i.e., minimizes the variance of the weights). This optimal function is given by $\pi_{opt} = p(x_k|x_{k-1}^i, y_k)$, [3, 44]. But this choice is usually not used since sampling from π_{opt} is not easy. A more convenient choice is to choose the importance density function to be

$$\pi(x_k^i|x_{k-1}^i, y_k) = p(x_k^i|x_{k-1}^i). \tag{2.25}$$

By substituting (2.25) in the Equation (2.24) we get

$$w_k^i = w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{p(x_k^i|x_{k-1}^i)} = w_{k-1}^i p(y_k|x_k^i). \tag{2.26}$$

Now, if we have the system model shown in Equation (2.2), reproduced here for convenience,

$$\begin{aligned} x(k+1) &= f(x(k), u(k), q(k)), \\ y(k) &= h(x(k), u(k)) + r(k), \end{aligned}$$

and the process and output noises $q(k)$ and $r(k)$ are characterized, we can use the model for each particle, by randomly generating noise samples, to predict their trajectories, and use Equation (2.26) to update the weights for each particle. The estimated system's state vector can be given as the weighted mean of the particles, i.e., (2.27), or simply by the best particle (the one with the highest weight). This Particle Filtering method is known as SIS (Sequential Importance Sampling).

$$\hat{x}_k = \sum_{i=1}^{m} w_k^i x_k^i \tag{2.27}$$

The application of the SIS algorithm to solve localization for mobile robots is straight forward. Just consider the pose of the robot as the system state, and use observations of known features on the environment made by the robot. Each particle corresponds to a possible vehicle pose, and, as the robot makes observations of the environment, these possible poses will be updated to represent an increasingly accurate estimate of the true robot pose.

Notice that we do not impose any assumptions about the noise or state vector distribution, and, moreover, there is no need to perform linearization, as required by the EKF. This makes the PF a very simple and accurate method. However, if an insufficient number of particles is employed, the SIS method may run into some difficulties. The number of particles must be sufficient to capture the pdf $p(x_k|Y^T)$, [39]. But even if the number of particles is sufficiently large, there is another problem, called the degeneracy problem. After a few iterations of the PF algorithm as presented above, the weights of all the particles, except one, may take

Algorithm 2.1 Particle Filter SIR algorithm

1. Initialize N particles (randomly trough the state space, or based on some previous information about the initial condition of the system)
2. while true
   (a) for each particle i
      i. Apply the motion model to estimate particle predicted state $x^i(t) = f(x_V^i(t-1), u(t-1), q^i(t-1))$, where $q^i(t-1)$ is a sample from the process noise distribution
      ii. Update the weight $w^i(t) = w^i(t-1)p(y(t)|x^i(t))$ based on the observation $y(t)$
   (b) endfor
   (c) Normalize the weights
   (d) if $N_{ef} < \alpha$, resample the particles based on $\hat{p}(x_t|Y^T)$
   (e) Compute the estimate state vector $\hat{x}(t)$
   (f) $t = t + 1$
3. endwhile

on very low values. So, even with a large number of particles, since they are not well distributed, the PF algorithm may fail to capture well the posterior distribution. This situation can be verified by Equation (2.28), as shown in [5]. Notice that small values of N_{ef} indicates degeneracy.

$$N_{ef} = \frac{1}{\sum_{i=1}^{m}(w_k^i)^2} \tag{2.28}$$

The degeneracy problem can be solved by what we call Resampling, or the Sampling Importance Resampling (SIR) method. The basic idea of Resampling is to eliminate particles that have small weights and to concentrate on particles with large weights. A new set of m samples is generated sampling the discrete distribution $\hat{p}(x_k|Y^T)$, and all the particles are equally weighted with $w_k^i = 1/N$. The Particle Filter SIR algorithm is summarized in algorithm 2.1.

Although the Resampling reduces the effects of the degeneracy problem, it introduces other practical problems. The main problem is that particles with high weights are statistically selected many times, and, in the case of low process noise, the distribution will have many repeated points leading to a loss of diversity among the particles. This is known as sample impoverishment and can be counteracted by

methods such as Resample-Move or regularization, [44, 5].

Recently, a large number of algorithms and applications based on PF methods have appeared in the literature to solve many applications in statistics and related fields (for a survey see [26]). However, few of these methods have been proved to converge rigorously. In [22], the authors prove the convergence of a general sequential Monte Carlo (SMC) method which includes most of the important features present in current SMC methods. This provides a solid theoretical backing for the validity of all the algorithms that can be obtained as particular cases of it.

2.1.4 Comparison of the Methods

As discussed earlier, the (regular or extended) Kalman Filter assumes that the estimated state is described by a Gaussian distribution. However, there are cases where this is clearly not the case. Consider for example a vehicle which is moving through a corridor which is known (by the vehicle) to have two doors. Suppose the vehicle initially does not know where along the corridor it is located. This could be represented by a Gaussian distribution with any estimated position and infinite variance, corresponding to maximum uncertainty about its position, as shown in Figure 2.1 (a). Once the vehicle senses a door, it increases its confidence of being close to a door (both of them), producing two peaks, as shown in Figure 2.1 (b).

In the case discussed above, the robot detects the door, but does not know which door it is detecting. In such situations, for which data association is hard and leads to conflicting hypothesis, the PF presents some advantage over the KF.

If both the peaks are Gaussian, there is no way to represent this kind of distribution using the regular (or the extended) KF. The best hypothesis would be a Gaussian distribution centered at the middle point between the doors. We could also use an extension of the KF (or EKF), called multi-hypothesis KF, that uses mixtures or sums of Gaussians, [94]. However, if the peaks are not Gaussian and

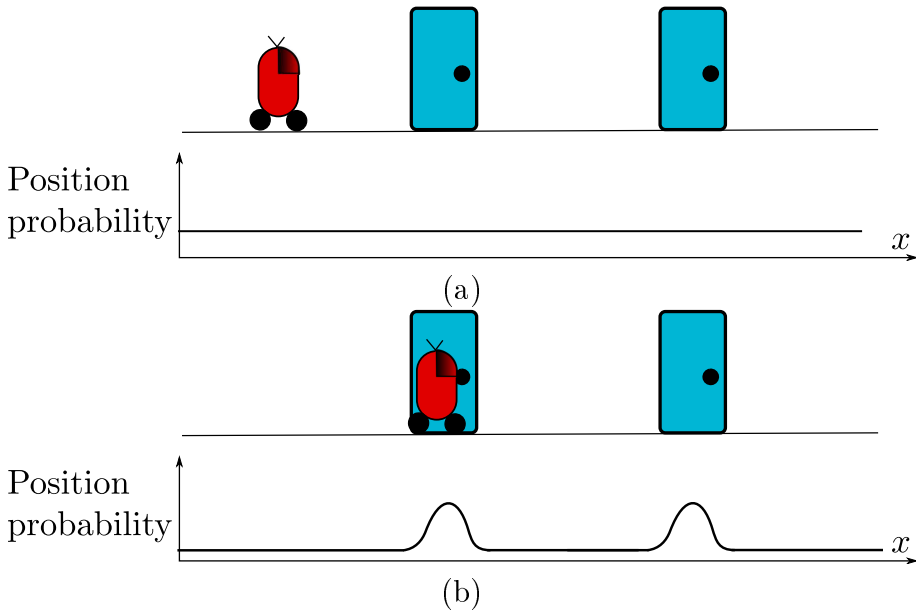


Figure 2.1: Probabilistic localization of a robot: a) the robot is initially completely unsure about its position; b) after sensing a door, the robot increases its probability of being close to a door represented on his map.

assume any other kind of distribution, of if the sensors have a non Gaussian error measurements, the PF methods come to advantage, since they do not assume any specific form for the distribution. Of course this comes at the cost of representing the distribution by a set of particles, instead of simply two multivariate parameters like in the Gaussian case: mean and variance. One way to reduce this added complexity of the PF is to use less particles if the pdf is of low complexity, and increase the number of particles as the pdf becomes more complex. These are called adaptative PF, [94].

Since these two methods, namely, the KF and the PF, are the most popular methods used in nowadays robotic applications, having each one its advantages and disadvantages, we decided to use both of them as a starting point for the development of the cooperative localization algorithm proposed on this thesis.

2.2 Networked Control Systems

In this section we provide an overview of issues, approaches and recent developments for Networked Control Systems which are more pertinent for the work in this thesis.

In the widest sense, a Networked System can be regarded as a collection of dynamic subsystems whose evolution relies on shared information either via sensing or communication, and a Networked Control System (NCS) can be defined as a networked system that be controlled. Control enters not only at the subsystem level, pretty much like in conventional control systems, but also at the information sharing process level, and, moreover, in the interdependence between these. These subsystems, also designated by agents, which may be physically distributed in the environment, have to communicate with each other in order to accomplish their goal.

Furthermore, the case in which control loops are closed through a real-time communications network is of particular relevance. Thus, a typical NCS may involve the following four basic ingredients:

- Sensors to acquire information;
- Controllers to compute control commands, and decisions;
- Actuators to perform the control commands; and
- Communication network to enable exchange of information.

The fact that NCS can be regarded as a collection of distributed nodes, each one controlled in order to achieve a global goal or optimizing a given collective performance function based both on local information, and on some shared information, enables the design of control systems with not only characteristics superior to those of conventional "monolithic" control systems, but also with unique functionalities. These, with emphasis to networked autonomous vehicle systems, are badly needed to address the critical challenges that human kind is facing to ensure the sustainability

of its presence on the Earth. Motivation for a formal framework to design systems for a wide variety of applications, such as, oceanographic observation, climate change research, environmental monitoring, natural resources management, security, surveillance, among others. Obviously, the potential of applications also covers a wide range of industries such as space and terrestrial exploration, access in hazardous environments, factory automation, remote diagnostics and troubleshooting, experimental facilities, domestic robots, aircraft, automobiles, manufacturing plant monitoring, health monitoring support systems, nursing homes, and an enormous variety of tele-operation contexts.

Key factors, such as, versatile and sophisticated data gathering requirements, high degree of survivability and persistence, superior resilience and robustness in attaining goals with adequate performance levels, and economic sustainability constitute important drivers for the development of such control design framework.

Moreover, the recent dramatic developments in virtual all the required underlying technologies - communications, computation, electronics, sensors and actuators - provide an environment in which the envisaged systems can actually be developed and deployed in operational scenarios. These technological advances fueled research of micro sensor integrated systems which, in turn, enabled large scale integration. Such systems combine the computing, sensing, energy source and radio technology which permit them to communicate with each other over an wireless link. When distributed over large areas, these elements form up a wireless network which can perform a variety of tasks that range from environmental monitoring and military surveillance, to navigation and control of a moving vehicle, [82].

In the past, the two areas of control and communication were disassociated with each other. While engineers designing controllers assumed that there were no error or delays in data transmission, the issues that were considered in the design of

communication networks concerned channel capacity, data packets loss and transmission delays without any consideration for the purpose of the transmitted data, [92]. With the rise of NCS it becomes clear that communication and control are tightly coupled and cannot be addressed independently. A communication network to exchange information is required whenever sensors, plants and controllers are located at different physical locations. For example, the problem of controlling a vehicle by using a network of sensors measuring its current position and velocity that send their measurements to the controller via wireless links. The sensor network provides observed data that are used to estimate the state of the controlled system, and this estimate is used for control. Thus, the quality of communication - data loss and delay - plays a key role in the controller, [92].

To materialize the tremendous potential of NCS, novel exciting challenges emerge that open many new critical research avenues and revives some older ones. The insertion of a communication network in the feedback control loop makes the analysis and design of an NCS much more complex. Issues such as reliability and security of communications, bandwidth allocation, time-driven and event-driven sampling strategies, data communication protocols, fault detection and fault tolerant control strategies, real-time information collection and efficient processing of sensor data, as well as their implications in control design, have, among other research topics, been the subject of a vast body literature.

The sensor network provides observed data that are used to estimate the state of the controlled system, and this estimate is then used for control. It is well know that this communication is not perfect, and assessing the impact of data loss or delay in the controller performance has been a key research issue. Depending on the application, time-delays could impose severe degradation on the system performance. Just to cite a few examples, a Gain Scheduler Middleware (GSM) methodology was proposed in [97] to alleviate the time-delay effect, and, in [62] a Smith predictor,

a Kalman filter and an energy regulator were designed to perform tele-operation in a NCS through the Internet. Many other researchers provided solutions using concepts from several control areas such as robust control, optimal stochastic control, model predictive control, fuzzy logic etc. Moreover, a most critical and important issue surrounding the design of distributed NCSs with the successively increasing complexity is to meet the requirements on system reliability and dependability, while guaranteeing a high system performance over a wide operating range, [101]. This makes network based fault detection and diagnosis techniques, which are essential to monitor the system performance, receive more and more attention.

In [80], Ling Shi *et al* argues the need of developing new control paradigms for large networks of wireless sensors and actuators in order to efficiently utilize system resources. As we can see, the communications problems introduced by the network directly affects the problem of control and estimation. Communication links introduce many problems, such as random delays, data loss and data corruption that might lead to performance degradation or even loss of stability, [38]. Graph theoretic methods have been addressed in [57] for networked multi-agent systems. Also, in [31] and [30], Fax and Murray show that the topology of the communication network is related to the stability of formation control for a network of multiple vehicles.

The problem addressed on this thesis can be considered an estimation problem in a sensor network. If you consider a centralized solution, one of the AUVs on the system will play the role of the controller/estimator, and the sensor network would be composed of the others AUVs. In a decentralized case, all the AUVs would have to share information with one another in order to accomplish the state estimation. In either case, the challenges introduced by the communication network have to be dealt with, and these become even more acute in the underwater environment where communications are very difficult.

A typical optimal controller for a given linear system encompass a Kalman Filter

(KF) to estimate the state and a state feedback controller. The KF uses the sensor measurements to compute the minimum mean square error estimate of the control system state and this state estimate is then used to compute the control command, [51]. This is well established in control theory in the instance of no information loss.

However, in a NCS, where different components of the control system communicate over a wireless network, the problems introduced by the communication network must be considered in the state estimator design, [40]. Ling Shi *et al* [80] shows that the optimal estimator over a sensor tree is given by a KF with a certain structure and that the loss or delay of information can directly affect the performance of the KF, even leading to an unbounded error state covariance. Stankovic *et al* proposed an algorithm for a distributed state estimation in a multi-agent network in [89]. Each agent could perform a state estimation with a local KF, and to determine the resulting state estimate, they used a dynamic consensus strategy between the agents. They showed that, under general conditions concerning local resources and the network topology, the proposed algorithm was asymptotic stable given a proper choice of the consensus gains. In [88], Stankovic et al extrapolate their previous work and study the distributed state estimation in a multi-agent network assuming intermittent observations and communication faults. The problem of state estimation involving a limited communication bandwidth was introduced in [108, 109], where they derived a new upper bound for the average estimation error and analyze how convergence properties of some coder-estimator algorithms are related to communication data rate and the rate of change of the state.

In [82], Sinopoli et al addressed the problem of Kalman filtering with intermittent observations, or information loss across the network. To study the statistical convergence properties of the estimation error covariance, they modeled the arrival of observations as a random process where the packet losses are Bernoulli-distributed, and showed that, if the observations rate of arrival is greater than a certain critical

value, the state estimation error can grow without bound with time. In their work is also presented upper and lower bounds on this expected state error covariance. In [2], Almstro *et al* argues that in reality packet losses tend to be correlated, so the Bernoulli loss process is not representative for wireless communication in industrial environments. So they represent the packet loss distribution by a two-state Markov model due to Gilbert and Elliot. They study how the estimator performance depends on loss probability and loss burst length, and to derive conditions on these parameters that guarantee that the mean-square error remains bounded. Other researchers model the KF with missing observations as a jump linear systems [68][20][54], which are stochastic hybrid systems characterized by linear dynamics and discrete regime transitions modeled as Markov chains.

Liu and Goldsmith, in [51], found similar results of [82], but considered the case where there are several sensors. Contrary to the idea of Sinopoli *et al*, where an observation is received in full or completely lost, Liu and Goldsmith analyzed the problem where there are partial observation losses, that is, packets arriving from different sensors are dropped independently. In [58], the authors argue that previous works on Kalman filtering with intermittent observation losses deal with the asymptotic behavior of the expected value of the error covariance. They, instead, consider a probabilistic statement of the error covariance, and show that the error covariance is bounded above, by a value M, with a high probability p, being the relationship between M and p determined.

Clearly, the research literature reveals a trend pointing towards an increasingly tighter coupling between estimation and control. This brings in a new dimension on the top of all sectorial challenges in that the scope of both the control and estimation problem formulations is enlarged. While the amount of data or information available for control might in itself be considered part of the control strategy to

be generated by the overall control system, the estimation problem has to be formulated so that control requirements are taken into account. The developments of this thesis reveal to some extent the need of the interplay between estimation and control when designing a NCS in the context of cooperative navigation. Moreover, this interplay is extremely critical given the very strict constraints of the context that we are considering.

In this Chapter we presented two of the most used techniques to perform estimation, namely, the Kalman Filter and the Particle Filter. We also shown how they relate to networked control when applied over networked systems. In the next Chapter we will show how the estimation techniques presented are used to solve the simultaneous localization and mapping problem.

Chapter 3

Simultaneous Localization and Mapping

In this Chapter, we start by giving an overview - main issues and major work - of the Simultaneous Localization and Mapping (SLAM) problem. Then, in the second and third sections, we investigate the relations of SLAM with the Networked Estimation Problem and how estimation techniques introduced previously can be used to solve the SLAM problem. In Section 3.4 we consider the case where the SLAM technique is used with dynamic maps.

3.1 Overview

The SLAM problem, also known as Concurrent Localization and Mapping problem, asks whether an autonomous robot placed in a unknown location in an unknown environment is able to incrementally build a map of this environment and, at the same time, use this map no navigate trough it. This problem has been of much interest in the robotics community since the seminal papers [83] and [27]. The SLAM problem has been formulated and solved in may different forms, and has been implemented in a wide range of domains, such as indoor and outdoor robots, underwater and airborne systems, [28]. A SLAM algorithm builds a consistent estimate of both

environment map and vehicle trajectory using noisy proprioceptive and some exteroceptive information, [12], while building a correlation matrix between landmarks which plays a fundamental role in the process.

It was shown in [83] that, as a robot moves through an unknown environment taking measurements of landmarks, these measurements are necessarily correlated with one another due to the common error in the vehicle pose estimation. This led to the conclusion that a consistent solution of the SLAM problem requires a joint state variable composed by the vehicle pose and the landmark locations. This implies that the dimension of the state vector increases linearly with the number of landmarks. By then, it was assumed that the map error would not converge, but exhibit a random walk behavior, i.e., unbounded error. These assumptions lead researchers to use approximations by minimizing or even eliminating the correlations between landmarks, [48, 76].

Later on, in 1996, Durrant-Whyte et al coined the term SLAM for a structure that is commonly used today and showed convergence results, [29]. Since this breakthrough, there has been many developments in the area, and the SLAM scheme has been solved by using several different techniques, such as probabilistic approaches, [32, 86], kinematic links, [73], Extended Kalman Filters, [25, 28, 66] and Particle Filters, [59, 60].

In [25], the authors prove the convergence properties of the EKF SLAM and also show that the correlations between landmarks play a major role in the convergence of the SLAM scheme. These correlations are exactly what guarantees that the SLAM scheme converges for an accurate relative map of the environment. They argue that, as landmarks are re-observed, their correlations grow to unity, thus forming a rigid relative map.

In the past few years, research in SLAM has been focusing in, basically, three

large areas, [12]: computational complexity, data association, and environment representation.

As we already saw, to be consistent, the SLAM scheme has to use a state variable, which is composed not only by the vehicle's pose, but also by all the landmarks found in the environment. If the number of landmarks is too big, this can bring tractability difficulties, since the computational complexity scales quadratically with the number of landmarks in a map. However, the problem formulation has a peculiar structure: the process model only affects the vehicle pose and the observation model, and, in most cases, only makes reference to a single vehicle-landmark pair (vehicle observing a single feature at a time). This enables researchers to develop a wide range of techniques exploiting this special structure, in order to limit the computational complexity of the SLAM algorithm. Some of the techniques used to reduce the algorithm complexity are state augmentation, [107], and partitioned updates (which uses the special structure of the SLAM algorithm), [37], and sub-mapping (which breaks a map into regions with local coordinate systems and arranged in a hierarchical manner), [36].

One of the major problems of SLAM is the so-called data association problem, i.e., the correct association of a given observed feature with one in the current map estimate, and, thus, already present in the state vector. The association of observations to landmarks is specially fragile in the EKF formulation of the SLAM scheme, [63], and, thus, incorrect associations may lead to inconsistencies in the estimated map or to the divergence of the SLAM scheme, causing the complete failure of the algorithm. Data association is particularly important when a vehicle returns to a previously mapped region after a long excursion, the so-called 'loop-closure' problem, [12].

The basic environment representation considers that the world is represented by a set of simple discrete landmarks described by geometric primitives such as points,

lines or circles. However, if one is working with more complex and unstructured environments, such as the marine or atmospheric milieus, this may no longer be a good option. In [67], the authors show how to implement a EKF-SLAM using a scan-matching that allows the definition of landmarks without resorting to geometric feature models. In [12], the authors also consider the problem of partial observability of landmarks, such as the one obtained by range-only or bearing-only sensors, which require several measurements in order to determine the landmark location, [65, 70].

3.2 Extended Kalman Filter SLAM

We saw in Section 2.1 that if a positioning system is not available, a classical technique for a vehicle to perform localization is to use a KF (or its nonlinear counterpart EKF) along with a map of the environment. If such a map is not available, then SLAM is an option for navigation. A robot can localize itself (determine its pose) by using a map of landmarks and on-board sensors which provide relative localization of these landmarks; conversely, if the robot knows a sufficiently accurate estimate of its localization, then it can build a map of the landmarks (i.e., determine their spatial coordinates). Performing both tasks simultaneously constitutes the Simultaneous Localization and Mapping (SLAM) problem, [7].

It is possible to use an EKF to solve a SLAM problem with non-linear system models, [25, 66]. In this case, the state vector $X(k)$ is composed by the vehicle pose, $X_v(k)$ (the capital letter is used to avoid confusion with the x coordinate of the vehicle x_v), and by the map feature parameters, that is

$$X(k) = [X_v(k)\ X_{f1}(k)\ X_{f2}(k)\ \ldots\ X_{fn}(k)]^T,$$

where $X_{fi}(k)$ are the state parameters of the i^{th} feature at time instant k. The system will be described by

$$\begin{aligned} X_v(k+1) &= f_v(X_v(k), u(k), k) + q_v(k), \\ X_{fi}(k+1) &= X_{fi}(k), \end{aligned} \tag{3.1}$$

where f_v describes the motion of the vehicle, and, since the features are static, their position remain unchanged over time.

Notice that, since the map is static, the features are not contaminated by noise. Only the motion of the vehicle motion model is affected by the process noise $q_v(k)$. The process noise matrix Q is given by Equation (3.2) and the Jacobian of the function f w.r.t. X by Equation (3.3). Q_v is the process noise of the vehicle motion function f_v and $D_X f_v$ is the Jacobian of f_v. Thus, we have

$$Q = \begin{bmatrix} Q_v & 0 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} \tag{3.2}$$

and

$$D_X f = \begin{bmatrix} D_X f_v & 0 & \cdots & 0 \\ 0 & I & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & I \end{bmatrix} \tag{3.3}$$

Now, the uncertainty matrix is

$$P = \begin{bmatrix} P_{vv} & P_{vf1} & P_{vf2} & \cdots & P_{vfn} \\ P_{f1v} & P_{f1f1} & P_{f1f2} & \cdots & P_{f1fn} \\ P_{f2v} & P_{f2f1} & P_{f2f2} & \cdots & P_{f2fn} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{fnv} & P_{fnf1} & P_{fnf2} & \cdots & P_{fnfn} \end{bmatrix} \tag{3.4}$$

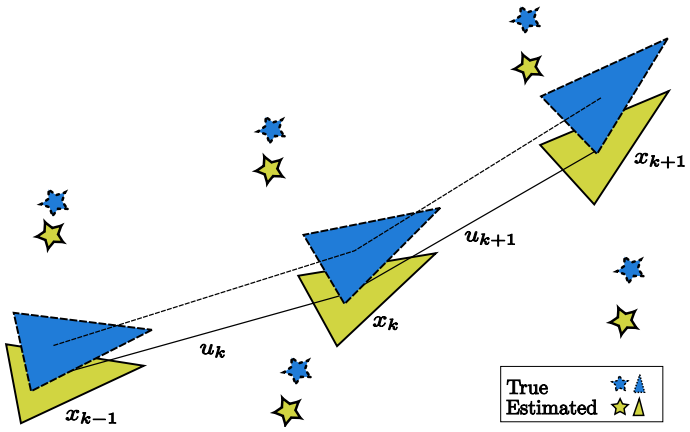


Figure 3.1: The SLAM algorithm. The constructed final map (and, thus, the estimated robot trajectory) is an accurate relative map. The error displayed between the estimated and true location of each feature is due to the initial uncertainty of the robot.

represents not only the uncertainty in the robot pose P_{vv}, but also the uncertainty in the map features parameters (P_{fi} for feature i) and also the correlations between the robot's pose and features, which are the non-diagonal terms of the matrix P.

As shown by Dissanayake *et al* in [25], the structure of the SLAM scheme is critically dependent on the maintenance of the complete knowledge of the cross correlation between landmark estimates. As the vehicle travels around the environment and observes features, the correlation of the errors in their estimates increases until, in the limit as the number of observations increase, the errors of the estimate of all features become fully correlated. At this point the map of relative locations of the features is known with absolute precision. This means that, given the real position of any feature, the whole map can be computed with zero error. It is also shown that the absolute error of the relative map reaches a lower bound determined only by the error that existed when the first feature was observed.

In Figure 3.1 we can see an example of the map and vehicles trajectory estimated by the SLAM algorithm. Notice that the estimated map/trajectory differs from the true one by some translation and rotation. This is due to the initial uncertainty in vehicle's pose as it observes the first features. Thus, the error in landmarks are

highly correlated. This means that, even if the true position of the landmarks are not known, the relative location between them may be known with high accuracy. In the EKF implementation this is represented by the off-diagonal terms of the covariance matrix P. As the vehicle observes features, the correlation between them increase monotonically. This means that, when a landmark is re-observed and its position updated, all other landmarks that are correlated with it are updated too, even if they are not observed directly by the vehicle. All landmarks end up forming a network linked by relative locations whose correlation values increase as observations are made. In the limit, a rigid accurate relative map is obtained, with some absolute error depending on the vehicle initial error, as show in Figure 3.1. This also means that the uncertainty of the landmarks (diagonal terms P_{fi} of the covariance matrix P corresponding to landmarks) decrease monotonically to a lower bound as observations are made. This can be seen in Equations (2.19), 3.2 and 3.3. Since the features are static and there is no feature process error Q_f, the P_{fi} terms of P do not increase in the prediction phase of the EKF. On the update phase, if a landmark is observed, its uncertainty is decreased as show in Equation (2.19). This way, the error on the estimated position of the vehicle relative to the map is bounded only by the quality of the map and relative measurement sensor, [28]. Observe that these results have been proved for the linear case and for Gaussian noises only, [25]. The EKF SLAM scheme is presented in the Algorithm 3.1.

Both maritime and aviation applications require navigation in a critical way. It is not surprising that navigation is an old science. According to [49], the problem of position determination has been of considerable interest over the last 4000 years, as the basic process of distance measurement, correlation, and triangulation was known to the Phoenicians. So, why robust and reliable autonomous mobile robot navigation remains such a difficult problem? In [49], the authors argue that the reason is clear: it is not the navigation process *per se* that is a problem, it is the

Algorithm 3.1 EKF SLAM algorithm

1. Initialize State Vector and Error Covariance Matrix
2. while true
   (a) Use proprioceptive information and the prediction equations to estimate $\hat{X}_p(k)$. For the vehicles next pose $\hat{X}_{v_p}(k) = f(\hat{X}_v(k-1), u(k-1), q(k-1))$, where $q(k-1) = 0$ (its mean value). Map estimates stays unchanged so $\hat{X}_{fi}(k) = \hat{X}_{fi}(k-1)$.
   (b) Compute the Jacobians $D_X f$ and $D_u f$, evaluated at $\hat{X}(k-1)$, and use predict equations to get the uncertainty matrix $P_{v_p}(k) = D_X f P(k-1) D_X f^T - D_u f Q(k-1) D_u f^T$.
   (c) If an observation from the i^{th} landmark is available
      i. Compute the Jacobian $D_X h$, evaluated at $\hat{X}_p(k)$, the innovation $e(k)$ and the Kalman gain $K(k)$.
      ii. Use update equations to update the state vector $\hat{X}(k) = \hat{X}_p(k) + K(k)e(k)$ (not only the vehicles pose $\hat{X}_v(k)$ and the observed landmark $\hat{X}_{fi}(k)$ will be update, but all the map, because of the correlations created between landmarks).
      iii. Use update equations to compute the uncertainty matrix $P(k) = P_p(k) - K(k)LK(k)^T$ (updates not only P_v and P_{fi}, but also other terms of P to account for the correlations).
   (d) else
      i. Make $\hat{X}(k) = \hat{X}_p(k)$ and $P(k) = P_p(k)$.
   (e) endif
   (f) $k = k + 1$
3. endwhile

reliable acquisition or extraction of information about navigation landmarks, from sensory information, and the automatic association or mapping of these with the ones in some navigation map that makes the autonomous navigation problem so difficult. If one uses artificial features, such as beacons, that can communicate to the vehicle and identify themselves, then the association problem is trivially solved.

A special case of SLAM is when only range measurements to landmarks are available. If there are n features in the environment, and the vehicle observes the feature i at time t, the observation equation is given by:

$$h(X) = \sqrt{(x_v - x_{fi})^2 + (y_v - y_{fi})^2},$$

where (x_v, y_v) and (x_{fi}, y_{fi}) are the position of the vehicle and of the i^{th} landmark in a 2D global reference frame, respectively. In this case, a single measurement does not contain enough information to determine the location of a landmark. If

there is no a prior position of the landmarks, this partial observability requires the fusion (processing) of several observations from different vehicle positions (in a 2D scenario at least 3 non-collinear measurements) in order to determine the landmark location, [65, 70]. But, once the positions of the landmarks have been initialized, or their positions are known a priori, then SLAM uses each observation to improve the estimate of the positions of both the vehicle and landmark. The partial observability when only range sensors are used brings up some other issues that will be discussed in more detail in Chapter 8.

To illustrate the presentation of the EKF SLAM scheme, we present now a simple, and yet important, example widely use in navigation and control problems in the area of mobile robotics.

Let us consider a 2D unicycle model for the vehicle motion. The state vector is composed by the position of the vehicle, (x, y), on the plane, and its orientation, θ, the angle formed by the vehicle's local reference frame relative to the global reference frame.

$$X_v = [x_v \; y_v \; \theta_v]^T \tag{3.5}$$

The most common unicycle model found in the literature is shown in equation 3.6, where v is the forward linear velocity of the vehicle and v_θ the angular speed.

$$\begin{bmatrix} \dot{x}_v \\ \dot{y}_v \\ \dot{\theta}_v \end{bmatrix} = \begin{bmatrix} v\cos(\theta_v) \\ v\sin(\theta_v) \\ v_\theta \end{bmatrix} \tag{3.6}$$

Here, we will consider that it is also possible for the vehicle to move laterally, such that the velocity will have two components (sway and surge). In this way, the control input is given by

$$u(k) = [v_x(k) \; v_y(k) \; v_\Theta(k)]^T,$$

where v_x and v_y are, respectively, the velocity along the x and the y axes of the vehicle's local reference frame and v_Θ is the turn rate of the vehicle at each time step. We consider that the control signal $u(k) = u_n(k) + q(k)$, where $u_n(k)$ is a nominal control which is corrupted by noise $q(k)$. The process is represented in Figure3.2. In order to consider the sway component of the velocity, we can represent the system in polar coordinates as

$$x_v(t) = \alpha(t)\cos\phi(t), \tag{3.7}$$

$$y_v(t) = \alpha(t)\sin\phi(t), \tag{3.8}$$

where $\alpha(t)$ represents the distance from vehicle's position $(x_v(t), y_v(t))$ to the origin and $\phi(t)$ represents the angle between $\alpha(t)$ and the x axis of the global reference frame. By differentiating both equations we get

$$\begin{aligned}\dot{x}(t) &= \dot{\alpha}(t)\cos\phi(t) - \alpha(t)\sin\phi(t)\dot{\theta}(t),\\ \dot{y}(t) &= \dot{\alpha}(t)\sin\phi(t) + \alpha(t)\cos\phi(t)\dot{\theta}(t).\end{aligned}$$

The velocities in the x and y axis of the global reference frame can be computed by $v^x = \dot{\alpha}(t)\cos\phi$ and $v^y = \alpha\dot{\theta}(t)\sin\phi$. Converting them to the velocities in the vehicle's frame we get $v_x(t)\cos\theta_v = v^x$ and $v_y(t)\sin\theta_v = v^y$, and finally we get the following model (dropping the time index t):

$$\begin{bmatrix} \dot{x_v} \\ \dot{y_v} \\ \dot{\theta_v} \end{bmatrix} = \begin{bmatrix} v_x\cos\theta_v - v_y\sin\theta_v \\ v_x\sin\theta_v + v_y\cos\theta_v \\ v_\theta \end{bmatrix}. \tag{3.9}$$

The discrete motion function of the vehicle is given by Equation (3.10), where, in this case, the velocities actually represent the displacement at each time step.

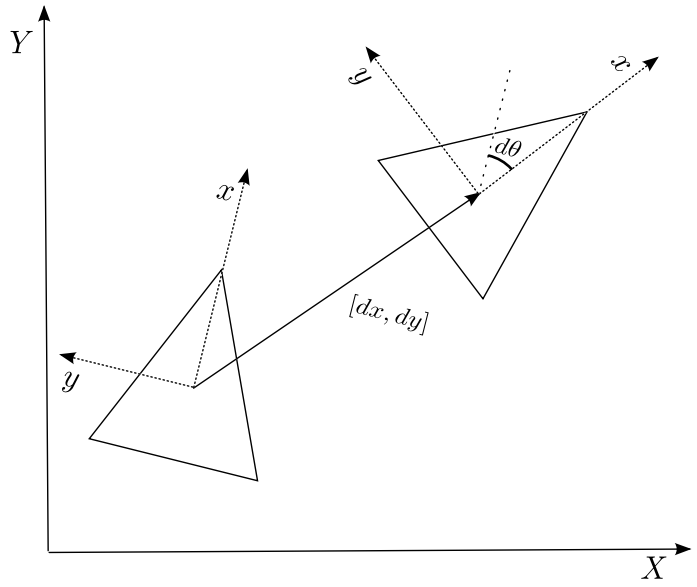


Figure 3.2: Unicycle model

$$
\begin{aligned}
X_v(k+1) &= f_v(X_v(k), u(k)) \qquad (3.10)\\
\begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} &= \begin{bmatrix} x_v(k) + v_x(k)\cos\theta_v(k) - v_y(k)\sin\theta_v(k) \\ y_v(k) + v_x(k)\sin\theta_v(k) + v_y(k)\cos\theta_v(k) \\ \theta_v(k) + v_\theta(k) \end{bmatrix}.
\end{aligned}
$$

The Jacobians of the vehicle motion function f_v w.r.t. the state and control variables are, respectively,

$$
D_{X_v} f = \begin{bmatrix} 1 & 0 & -v_x(k)\sin\theta_v(k) - v_y(k)\cos\theta_v(k) \\ 0 & 1 & v_x(k)\cos\theta_v(k) - v_y(k)\sin\theta_v(k) \\ 0 & 0 & 1 \end{bmatrix}, \qquad (3.11)
$$

and

$$
D_u f = \begin{bmatrix} \cos\theta_v(k) & -\sin\theta_v(k) & 0 \\ \sin\theta_v(k) & \cos\theta_v(k) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (3.12)
$$

Now, suppose that the environment is populated by a set of n discrete landmarks, or features, whose locations in the plane can be described by a set of parameters. In

this case, these parameters represent the location of the feature in the plane, that is, the i^{th} feature at time instant k is place at $X_{fi}(k) = (x_{fi}(k), y_{fi}(k))$. The set of all landmarks compose the map state vector $X_f = [X_{f1} X_{f2} \cdots X_{fn}]^T$. Since the features are static, their motion function f_f is given by $X_f(k+1) = f_f(X_f(k)) = X_f(k)$. So we may drop the the time index and represent $X_f(t)$ by X_f.

The Jacobians of the feature motion function are given by

$$D_{X_f} f = I_{2n \times 2n}, \tag{3.13}$$

$$D_u f = 0_{2n \times 2n}. \tag{3.14}$$

By lining up the state of the vehicle and the state of the features in a single vector X, we obtain

$$X(k+1) = \begin{bmatrix} X_v(k+1) \\ X_f(k+1) \end{bmatrix} = f(X(k), u(k)) = \begin{bmatrix} f_v(X_v, u(k)) \\ X_f(k) \end{bmatrix}. \tag{3.15}$$

By combining Equations (3.11)-(3.14) we get the Jacobians of the combined function f

$$D_X f = \begin{bmatrix} D_{X_v} f_v & 0 \\ 0 & I_{2n \times 2n} \end{bmatrix}, \tag{3.16}$$

$$D_u f = \begin{bmatrix} D_u f_v & 0 \\ 0 & 0_{2n \times 2n} \end{bmatrix}. \tag{3.17}$$

If the robot is equipped with a range and bearing sensor, that, at each time step, observes one of the features present in the environment, the observation model will be given by Equation (3.18).

$$h(X(k)) = \begin{bmatrix} r \\ \theta_o \end{bmatrix} = \begin{bmatrix} \sqrt{(x_v(k) - x_{fi})^2 + (y_v(k) - y_{fi})^2} \\ \arctan(\frac{y_{fi} - y_v(k)}{x_{fi} - x_v(k)}) - \theta_v(k) \end{bmatrix} \tag{3.18}$$

The Jacobian of the observation function w.r.t. the pose of the vehicle, $D_{X_v}h$, and w.r.t the observed feature i, $D_{X_{fi}}h$, are given by Equations (3.19) and (3.20).

$$D_{X_v}h = \begin{bmatrix} \frac{x_v(k)-x_i}{r} & \frac{y_v(k)-y_i}{r} & 0 \\ -\frac{y_v(k)-y_i}{r^2} & \frac{x_v(k)-x_i}{r^2} & -1 \end{bmatrix} \tag{3.19}$$

$$D_{X_{fi}}h = -D_{X_v}h \tag{3.20}$$

The Jacobian w.r.t. all other features are zero. So the combined Jacobian of the function h is given by

$$D_X h = [D_{X_v}h\, 0 \cdots 0\, \nabla D_{X_{fi}}h\, 0 \cdots 0] \tag{3.21}$$

$$D_X h = \begin{bmatrix} D_{X_v}h & 0 & \cdots & 0 & \nabla D_{X_{fi}}h & 0 & \ldots & 0 \end{bmatrix} \tag{3.22}$$

Now, if we have the initial conditions $X_p(0)$ and $P_p(0)$, and the matrix $Q(k)$ and $R(k)$ introduced earlier in the description of the EKF scheme, this can be used - see the EKF Equations (2.19) - to build a map of the environment and, at the same time, produce an estimate of the robot localization relative to this map. Now, observe that having an initial condition $X_p(0)$ means not only to have a guess for the vehicle's initial position, but also a guess for the map of the environment. This is not consistent with the philosophy of SLAM! We are supposed to construct the map from a totally unknown environment. The way out of this is to start the state vector with an arbitrary pose of the robot only, that is, $X(0) = X_v(0)$. Then, as the robot moves trough the environment and observes features, initial position estimates are generated for them, and these are added to the state vector.

3.3 Particle Filter SLAM

We saw in Section 2.1 that a Particle Filter (PF) can be used to perform localization for nonlinear systems. But what if we want to used it for SLAM? We already

saw that, in SLAM scheme, the state vector is composed of the vehicle's pose and map feature parameters. We could use a PF in the same way as we used for the localization problem, where each particle now will represent not only a possible robot pose, but also a possible configuration of each parameter of each feature of the entire environment map. However, this high dimensional state space of SLAM makes the application of PF algorithms computationally infeasible, [28]. The number of particles necessary to cover all the state vector space must be really high, and its computation in real time would be impossible.

One way to overcome this difficulty is to use a Rao-Blackwellised particle filter, in which the joint state variable is partitioned according to the product rule

$$p(x_1, x_2) = p(x_1)p(x_2|x_1). \tag{3.23}$$

If the term $p(x_2|x_1)$ can be described analytically, only $p(x_1)$ needs to be sampled. The way to use this propriety in the context of SLAM is to use the observation as if the robot's path was somehow known exactly. In this case, the measurements of each landmark would be independent. Thus, the exact knowledge of the path of the robot, enables the decomposition of the computation of the landmark locations into independent estimation problems, one for each landmark,[59]. This independence happens only if you consider the state vector as a trajectory of the vehicle $X_v^T = x_v(0:k) = x_{v_{0:k}}$ rather then a single pose $x(k) = x_k$. So, in Equation (3.23), the term x_1 would represent the trajectory of the robot, and the term x_2 the map associated with that trajectory. In this way, the joint state will be factorized into two components: the vehicle's trajectory and the map. The vehicle's distribution will be represented by a set of weighted samples as in the PF approach, and the map distribution can be described analytically by a set of independent Gaussians as in the EKF approach, as shown in Equation (3.24).

$$p(X_v^T, x_{fi}(k)|Y^T) = p(X_v^T|Y^T)\prod_{j=1}^{n} p(x_{fj}(k)|X_v^T, Y^T) \tag{3.24}$$

This means that the problem can be decomposed into independent estimation problems: the robots path posterior, and the location of each landmark. Equation (3.25) shows how one particle looks like

$$x^i = \{X_v^{T^i},\ p(x_{f1}|X_v^{T^i}, Y^T), \cdots,\ p(x_{fn}|X_v^{T^i}, Y^T)\}. \tag{3.25}$$

Here, $X_v^{T^i}$ represents the trajectory of the robot for particle i, and $p(x_{fj}|X_v^{T^i}, Y^T)$ represents the pdf for the j^{th} feature associated with the i^{th} particle.

The map update is done as in a mapping problem, that is, we consider that the robot pose is known (the pose of the particle to which this map is attached) and uses EKF equations to update the posterior for the observed feature. Unobserved features remain unchanged. This is known as the FASTSLAM algorithm, proposed by Montemerlo *et al* in [59].

The path of the vehicle is updated by using a regular PF SIR filter. Although the formulation considers the state space of the vehicle as a trajectory, when generating the particles at time instant k, we only need to use the particles at time instant $k-1$ (propagate the last pose of the vehicle). This makes the size of the particles independent of time, [59]. The scheme is summarized in Algorithm 3.2.

There are two proposed FASTSLAM algorithms: (1.0) and (2.0) [59, 60]. They differ only in the form of the chosen distribution. While in the (1.0) version, the distribution is just the motion model, in the (2.0) version the observations are included, and, as discussed in Section 2.1.3, this makes this last version much more accurate. There are well known theoretical problems with the statistical accuracy of the FASTSLAM algorithms, [28], although in practice they are capable of generating accurate maps, [60].

Algorithm 3.2 Rao-Blackwellised Particle Filter SLAM algorithm

1. Initialize N particles (randomly trough the state space, or based on some previous information about the initial condition of the system). Notice that each particle

$$x^i = \{X_v^{T^i}, p(x_{fj}|X_v^{T^i}, Y^T)_{j=1}^n\}$$

represents a robot trajectory X_v^T and an associated distribution for each j^{th} map feature $p(x_{fj}|X_v^{T^i}, Y^T)$.

2. While true
   (a) For each particle i
      i. Apply the motion model to estimate particle vehicle component

$$x_v^i(k) = f(x_v^i(k-1), u(k-1), q^i(k-1)),$$

      where $q^i(k-1)$ is a sample from the process noise distribution
      ii. Update the weight $w^i(k) = w^i(k-1)p(y(k)|x^i(k))$ based on the observation $y(k)$
      iii. Use EKF update equations to update the j^{th} observed feature posterior $p(x_{fj}(k)|X_v^{T^i}, Y^T)$ based on the pose of the i^{th} particle (the particle associated with it) and on the observation.
   (b) Endfor
   (c) Normalize the weights
   (d) If $N_{ef} < \alpha$, resample the particles based on $\hat{p}(x_k|Y^T)$
   (e) Compute the estimate state vector $\hat{x}(k)$
   (f) Let $k = k + 1$
3. Endwhile

Wenyan Hu *et al,* in [39], proposed a Rao-Blackwellised PF in which the distribution of the landmarks are also represented by particles, instead of using a Gaussian distribution. The advantage of this method is that there is no assumption on the posterior distribution of the landmarks, and both the robot's path and the map can be non-linearly approximated by particles. This makes the algorithm computational complexity higher, although, according to Wenyan Hu, it is still applicable in real-time. This is done by representing the robot's path by a set of particles in the usual way and, attached to each robot particle, there is a set of particles representing the landmarks. With this approach, the correlations between the estimated pose of the robot and the positions of the landmarks are fully accounted for, and the algorithm complexity only grows linearly with the number of map features.

3.4 SLAM With Dynamic Map

In the previous Section, we described the SLAM algorithm with static features. What if we consider the SLAM formulation with non static features? First of all, we have to define what is a map with moving features. Usually, when we talk about mapping, in general, we refer to relatively fixed features in the environment. So, what do we mean exactly by saying that the features are capable of moving? Do they move in a completely unknown way, or do they follow some kind of pattern? Or maybe move in some formation? Are they able to move at any time? Do we possess any kind of control about the motion this features describe? What kind of velocities does the features have, when compared with the vehicle? We need to answer these questions in order to better define the application of the SLAM with moving features algorithm presented in this work.

In cartography, the term “mapping" means constructing a representation of an area, i.e., a symbolic depiction highlighting relationships between elements of that space such as objects, regions, and themes, [104]. Usually, the notion of a map

requires persistence over time (e.g., stationary) as it should preserve the ability of motion reference. However, in a SLAM with moving features, the map M is no longer represented solely by a spatial component x, but needs also a time component t, so that $M = M(x,t)$. It is well known that, in the conventional SLAM with static features, the map converges to the true map with some offset, [12, 28]. This means that an accurate relative map of the environment is generated differing from the real one only by a fixed homogeneous transformation. Now, such a guarantee does not exist in the case of SLAM with dynamic features.

In the static map SLAM case, since there is no motion of the features, the system can be modeled as $X_{fi}(k+1) = X_{fi}(k)$, where fi is the i^{th} feature. There is no process noise contaminating the features, only the vehicle motion model is contaminated by a noise, as show in Equation (3.2). To say that a map is dynamic means that the features are able to have a dynamic behavior such that $X_{fi}(k+1) = f_i(X_{fi}(k),\, u_{fi}(k), k) + q_{fi}(k)$, being the matrix Q now given by

$$Q = \begin{bmatrix} Q_v & 0 & \cdots & 0 \\ 0 & Q_{f1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & Q_{fn} \end{bmatrix}, \tag{3.26}$$

where u_{fi}, f_i and Q_{fi} are the control, the dynamics and the process noise covariance matrix of the i^{th} feature, respectively. We consider the case for which all these parameters are known.

We are considering a SLAM scheme with moving features motivated by the multiple AUV systems application scenarios with underwater environments poor in natural features. To address this difficulty, some artificial features are added to the environment. These features might be static or mobile. We already observed that static features limit the operation area of the vehicle, so it is of interest that, at least some of the features, are mobile. What can we say about the motion of

these features? If their goal is only to help the localization of the vehicle, then we can infer that these mobile features must have the ability to move along with the vehicle, keeping a certain minimum distance. Thus, they must move in velocities similar to the vehicle, and must be controllable in some way. If not directly by the vehicle itself, they must at least have some information about where the vehicle is, so that it is possible for them to plan they trajectories in order to move along with the vehicle if necessary. The features could move keeping a certain formation, for example. The other way to look at the problem is to think that the moving features do not have the single goal of supporting the localization other vehicles, but are, in themselves, vehicles operating in the same environment, possibly in a cooperative mission. In this context, vehicles interact with one another in a reciprocal manner in order to improve their localization by sensing and communicating among them. In this case, keeping a formation might no longer be an option.

In [25], the authors prove the convergence of the static map in a EKF SLAM algorithm based on the principle that there is no process noise injected on the features dynamics. This means that, when a feature is not observed during a time step of the algorithm, the prediction equations of the KF are used to predict its next pose and pose uncertainty. But, since there is no process noise, the covariance matrix remains constant. On the other hand, when a feature is observed, the update equation either decreases its uncertainty or leaves it unchanged. This guarantees that the covariance matrix will converge to a minimum value, which depends on the initial covariance of the vehicle location estimate. It is also shown that, in the limit, as the number of observations increases, the features estimates became fully correlated. In this way, a relatively precise map of the environment is constructed.

In the dynamic map SLAM, since the features are contaminated by noise, there is no guarantee that the map will converge. When a feature is not observed, it is moving and accumulating error, so its uncertainty grows. Even when it is observed,

although the uncertainty may possibly decrease, the error of the map might not be bounded. To make the system observable, at least one of the vehicles must have some absolute positioning capabilities, as shown in [77].

But even if the system is observable, a feature is not guaranteed to be observed each time step. Actually, in most SLAM algorithms, the vehicle observes at maximum one single feature at each time step, and it may spend several time steps without observing any feature. This is specially true in outside environments, where the sensor used by the vehicle has a limited range, or when the observation relies on the communication between the vehicle and the environment features subject to communication constraints. In these cases, it is impossible to know exactly when a feature will be observed.

Considering this scenario, we notice that the system can be modeled as a Networked Control System, as discussed in Section 2.2, where the observations are considered to be communications between agents, with a certain probability of success. The fact that the vehicle does not observe a feature at some time can be regarded as a communication failure.

In the next Chapter, we will formally present the main problem addressed in this thesis, that is, the cooperative localization of AUVs, and show some of the solutions proposed in the literature.

Chapter 4

Cooperative Navigation for AUVs: Motivation and Problems

In this Chapter, Section 4.1 will examine the scenario in which several AUVs execute cooperative missions. We motivate this class of missions by showing why systems with multiple AUV based systems are of interest in important classes of applications. We also present the problems that arise with this configuration, and show the main existing solutions for related problems with a focus on the localization problem in Section 4.2. Finally, in Section 4.3, we give a brief discussion about SLAM in a cooperative scenario.

4.1 Advantages of Cooperative Missions for AUVs

Underwater multi-agent systems have a great potential to address the formidable challenges arising in the data gathering and intervention in the ocean. We argued in Chapter 1 that the role of oceanography is addressing critical societal issues, such as environmental sustainability and climate change, no strange to its prominent emergence. Unfortunately, we still know little about the vast bodies of water constituting the majority of our planet and, hence, the pressure to acquire the required huge amounts of data from the marine environment is enormous. Systems based on controlled mobile sensor platforms emerge as one of the most promising technologies

to reduce the overall time and cost of acquiring data over large volumes of water.

We have already seen in Chapter 1 that LBL limits the operation area and demands a lot of effort for a single AUV. For some classes of missions, multiple AUVs may be used, each one operating in an area disjoint from that of the others. In this case, independent sets of navigation beacons have to be installed. However, in this context, the logistical complexity of the mission increases rapidly, [84] That is why SLAM schemes for underwater robotics becomes interesting.

In the SLAM problem, the map usually consists of natural or artificial features found in the environment. However, the underwater environment is, in general, very poor in natural features, even close to the sea floor. Thus, a solution often adopted consists in deploying artificial features in the area of operation, like moored floating devices in the water column or beacons sitting at the sea floor (static features). This set up is similar to LBL and, therefore, not only limits the operation area, but also requires a substantial deployment effort before operation, especially in deep water, [11]. What if other AUVs were used as features? In this context, we can have several AUVs on the water, each one executing its own mission (which might be part of a bigger, cooperative mission), and also using one another navigation data and estimated ranges between them in order to improve the estimate of their own positions. In [102], the authors present a platform-independent acoustic communication system that enables underwater vehicles and/or surface craft to simultaneously exchange data and calculate inter-node ranges with accuracy in the order of 1 meter. This scenario, in which several AUVs work cooperatively to accomplish a given mission exhibits very interesting and promising features in what concerns fulfilling sampling requirements - model adapted, spatial and temporal distribution patterns, etc. - in a robust, and an operationally and economically efficient fashion. For example, searching a particular area can be done faster using multiple AUVs which might be simpler, cheaper and easier to operate. Another good scenario are dangerous

missions in hostile environments, in which the survival rate or endurance of an AUV might be low, and, thus, it might be better to deploy several inexpensive, with less capabilities and, possibly prone to failure AUVs, instead of a single AUV which would have to be much more expensive in order to accommodate the equivalent capabilities.

When a group of robots involved in a mission is composed of different platforms carrying different proprioceptive and exteroceptive sensors, or executing different trajectories in a given region of the environment, the quality of the localization estimates will vary significantly across the individual members at a given time. If the members can sense and communicate with each one another at all times, then, by using this information, every member of the group would have less uncertainty in its own position estimate than that of the robot with the best localization information, [77].

In multi-vehicles missions, the robots must act cooperatively in order to achieve one goal in a more effective and efficient way. This involves several tasks, of which we single out: (i) localization of each robot relative to the world and relative to one another; (ii) coordination of a common control strategy so that, by accomplishing its own specific goal, each robot contributes effectively to the successful completion of the whole mission; (iii) path planing for each vehicle that takes into account the positions of the others to avoid collisions, the overlap of explored areas, and the navigation requirements that can be achieved only with cooperation; (iv) management of the overall communications network so that a sufficiently robust connectivity is guaranteed to satisfy the mission execution and the inherent data flow. All this requires communication between robots which constitutes a formidable constraint in the underwater environment. In this thesis, we will focus on the problem of cooperative localization having in mind that the minimality of the shared data is a strong requirement imposed by the strict communication constraints.

4.2 Cooperative Localization

Although a lot of effort has been spent on addressing the problem of single vehicle localization by casting it in a state estimation probabilistic framework, localization for the case of a team of robots, though, is a still relatively new field.

Kurazume *et al,* [45], and Rekleitis, [75], proposed a method that divide ground operating robots into two groups, A and B. While group A remains stationary and acts as a set of landmarks, vehicles of group B use these landmarks to support their navigation in the execution of the mission activities. Then, these two groups or part of them may switch roles in order to achieve the overall mission goals. This is a good strategy for ground robots. However, for an AUVs it is hard to stay stationary or behave in a equivalent manner. Moreover, given the severe limitation of resources, it is desirable that AUVs are able to execute their own specific tasks without significant disruption due to additional activities required for the localization of the others AUVS.

Alexander Bahr *et al,* [11], proposed a solution, called Cooperative Navigation Algorithm (CONA), using multiple vehicles. A subgroup of the AUVs involved in the mission were called Communication and Navigation Aid-AUVs (CNAs). The CNAs could maintain an accurate estimate of their positions through sophisticated DVL, INS sensors or even GPS when surfaced. The CNAs would communicate, one at a time, with the AUV. The data communicated consists in the CNA's estimated position, $x(k)$, $y(k)$, and $z(k)$, and pose uncertainty (covariance matrix) to the AUV, being range between the vehicles computed based on the time of flight. The amount of information shared is small, and, thus, suited for the strict bandwidth communication constraints of the underwater environment. Since depth can be accurately measured with a pressure sensor, the AUV can use its depth and the depth received from the CNA to project the CNA's position into a 2D plane and,

in this way, reduce the cooperative localization from a 3D to a 2D problem. In between communications with CNAs, the AUV keeps track of his traveled distance with data from its proprioceptive sensors. If the AUV communicated at time k_1 with a CNA, and, then, communicated again later (with another or even the same CNA) at time k_2, the AUV builds the matrix $d_{k_1,k_2} = [dx_{k_1,k_2}, dy_{k_1,k_2}]^T$, where $d_{k_1k_2}$ represents the distance traveled by the AUV in between communication times k_1 and k_2 (this distance is obtained from proprioceptive measurements), as well as the covariance matrix P_{k_1,k_2} associated with that measurement. At time k_1, when the AUV communicates with the first CNA (CNA_1), which is computed to be at a distance r_1 from the AUV, the circle c_1 with radius r_1 centered at the position of CNA_1 at time k_1 defines all possible positions of the AUV at time k_1. Similarly, at time k_2, there will be a circle c_2 of radius r_2 centered at the position of CNA_2 at time k_2 which defines all possible positions of the AUV at time k_2. By shifting the center of c_1 by d_{k_1,k_2} the two circles will intersect, at most, at 2 points. One of them is chosen to be the estimated position of the AUV at time k_2. The CONA algorithm is illustrated in Figure 4.1. It is possible to use not only the last communication, but the last n communications, and we can compute up to $2n$ possible solutions. To choose which one of the possible positions is going to be the estimated AUV position, Alexander Bahr et al uses a statistical method. He proposes a cost function which computes the cost (inverse of likelihood) of the AUV having traveled from the position at time k_1 to the position at time k_2 and chooses the most likely position. In Chapter 7, we will use CONA as a comparison for the the developed methods in this thesis.

This is also the idea underlying the Moving Long Base-Line in [98]. Their method takes into account the small bandwidth of the underwater acoustic communications and requires good pose estimation for the CNAs and low noise measurements by the AUVs. The estimate of this uncertainty of the CNAs position is taken into account

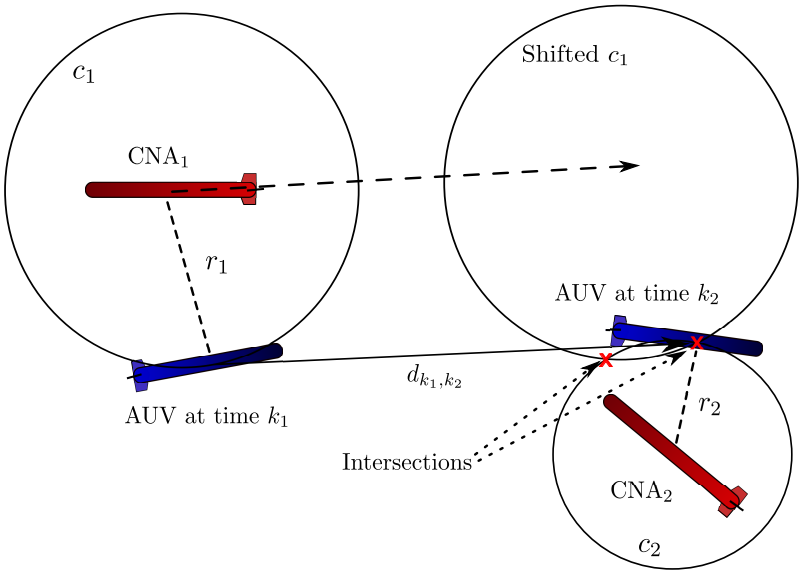


Figure 4.1: Cooperative Navigation Algorithm: the AUV position is given by the intersection of circles

when using range information.

The approach we propose improves these results by suppressing the CNAs, that is, all the AUVs involved have similar and poor Navigation systems, and by allowing relatively high process and measurement noises.

In [77], the authors present what they called collective localization, in which each robot collects data regarding its own motion and share information during the update cycles. This exchange of information is only necessary when two vehicles measure their relative position, but it is required that all the vehicles successfully communicate with one another whenever there is an observation. They show that, by exchanging only individual estimates of pose and covariance, inter-vehicle correlation can still be maintained. In their observability analysis, they disregard the fact that the robots do not observe each other at each time step, and, as discussed in Section 2.2, this can lead to instability. In [13], the authors propose a scheme where some part of the information (proprioceptive measurements) is fused in a decentralized manner across all the vehicles. When a pair of vehicles observe each other, obtaining exteroceptive information, it is fused in a centralized way by sending data to a central

server. For this purpose the authors use the information-form of the Kalman filter.

Most properties of SLAM are proved for environments with still features, although SLAM can still be used for moving features, [25, 100]. Without the aid of statical features, error in the pose estimate of the vehicles will not be bounded, but can still be greatly reduced, [79]. To be able to update the state vector in the standard SLAM algorithm with moving features, all the features will have to send their control signal to a central processing unity at every step of the algorithm, to compose the overall control signal $u(k) = [u_v(k)\, u_{f1}(k)\, u_{f2}(k) \ldots\, u_{fn}(k)]^T$, where $u_v(k)$ and $u_{fi}(k)$ are the control signal of the Vehicle and of the i^{th} moving feature and at time instant k, respectively. This operation requires high bandwidth communication between the AUV and the features. This is the idea used in [79], which is applied to land or air vehicles that can rely on fast communication.

In the underwater environment, acoustic communications is probably the most viable type of communication available. Unfortunately, it is slow and unreliable. Some development programs investigated and evaluated other technologies such as laser communication at short range, and relatively noise free communications over larger ranges using RF current density techniques. However, the strong attenuation of electromagnetic waves in the underwater milieu implies that radio or optical communication can be used only for distances of a few meters. For longer range distances, we are restricted to acoustic modems. In the past few years there has been significant advances in acoustic communications and, nowadays, relatively low error rate communications is possible over ranges of Kms at bit rates of a few Kbps, [14]. Moreover, since sound propagation is dependent on temperature, salinity and depth in the water column (pressure), the acoustic communication channel is unreliable and its performance is hard to predict, [11].

Another issue is that range calculations based on underwater communication (TOF) is contaminated by several noise sources, some of which usually are not

Gaussian. One source of error is the one mentioned just above: the fluctuation of the speed of sound in the water. Multi-path is another error source, when the signal, instead of traveling directly from the source to the destination, reflects on the ocean surface or floor first. Both these sources are related to environmental conditions, which do not change rapidly, and can be non-Gaussian, [70]. Since KFs yield optimal solutions for Gaussian noise only, outliers measurements must be removed previously. Similar argument implies that this also holds for EKFs. In a real-life application, a technique to remove the outliers, such as the one presented in [70], must be used. The approach proposed here to identify outliers in range data is based on modeling measurements by a graph, and, by applying graph partitioning algorithms to identify sets of consistent measurements. Here, two measurements are considered consistent if they can be explained by a landmark at some particular location (with some tolerance). Then, each measurement is associated with the dead reckoning pose of the vehicle. In this way, an undirected graph of pairwise consistent measurements is constructed, that is, measurements represented by vertices and consistent measurements are connected by an edge. In this way, it is possible to identify the outliers by dividing the graph into two sets of vertices by cutting edges such that the inliers are in one set and the outliers are in the other. To do this, a function that measures the quality of a cut (with inliers separated from outliers) is maximized. Unfortunately, this method cannot be directly applied in our problem since the features are not static.

By assuming that the outliers in the range estimation based on TOF of communications are correctly identified, and by choosing a EKF with the cost of non-optimal solutions, as in [79], then there is still the problem of all AUVs communicating their control signals to a central AUV, what, typically, would be extremely difficult to implement. To avoid sharing so much information, we could simply consider the AUVs as known features in the environment, communicating with each other one at

a time. However, this scheme leads to bad results as will be discussed in detail in Chapter 5. There is also the difficulty of the non-linearity of the system, which may lead to an unsatisfactory performance of the EKF.

A better way to deal with non-linearities is to use Particle Filters (PF). We saw in Chapter 3 that a Rao-Blackwellised is ideal address the SLAM problem. The FASTSLAM solutions proposed by Montemerlo et all in [59, 60] have the important advantage over the EKF approaches of not requiring the Gaussian assumption on the posterior probability functions. Another advantage is that while the computational complexity of EKF algorithms grows quadratically with the number of landmarks in the environment, the complexity of PF increases logarithmically. However, for underwater cooperative localization, the main problem is not only the number of features (in our framework, this is the number of vehicles involved in the mission), but also the low bandwidth communication channel. Even if the group of AUVs involved were large, in any case, they would probably have to be split up in smaller subgroups due to communication constraints. Besides, the FASTSLAM approach considers features characterized by Gaussian distributions, what, in our case, is not reasonable, since they are other vehicles with nonlinear motion models.

The method proposed by Wenyan Hu *et al* in [39], presented in Chapter 2, seems to be more appropriate. It represents the posterior distributions of the landmarks as a set of particles as well, and by using a Rao-Blackwellised PF, the complexity of the algorithm is reduced. However, there is still the communications issue. It is very difficult for a given AUV, say AUV1, to communicate the whole set of particles that represents its pose distribution to another AUV, say AUV2, so that AUV2 could use this information combined with its own, and the range between them, to improve the estimates of the poses of both of them. Moreover, once these computations are completed, AUV2 still will have to send back to AUV1 its new estimate posterior distribution, which, again, is a large set of particles.

The fact that vehicles must communicate with each other in order to perform the Cooperative Localization characterizes the system as a Networked System. This fact has several implications on control and estimation algorithms acting on this kind of system, as discussed in Section 2.2.

4.3 Cooperative SLAM

In order to a multi robot team be able to coordinate while navigating autonomously within an area, all robots must be able to determine their positions with respect to a common reference frame. When a sufficiently accurate global positioning system is not available, the robots of the team can still improve their localization accuracy, particularly if the estimates were obtained with dead-reckoning only, by using relative position measurements, as we saw in Section 4.2. However, performing only Cooperative Localization using relative position measurements causes the uncertainty of the robots' position estimates to increase with time. One way to bound the error is to localize while concurrently building a map of the environment, that is, to perform SLAM. This formulation of doing cooperative localization and SLAM at the same time is what is called Cooperative Simultaneous Localization And Mapping (C-SLAM) that has recently attracted the interest of many researchers. The number of potential applications that require robots to perform C-SLAM is growing continuously, as autonomous vehicles are more and more employed for tasks ranging from planetary exploration and environmental monitoring, to construction and transportation, [61].

In [112], the authors use indirect communication to share knowledge and use a gradient-like local search to direct robots towards interesting areas. In order to share a common reference frame among robots, they use a feature based SLAM approach, where features are RFIDs tags deployed by each robot for building a network of reachable locations that represent a topological map of the environment.

Observations are RFID range readings. Maps generated by each robot are merged only at the end of the mission into a global topological map by performing the union of the local maps based on the association of RFIDs.

In [99], the authors address a form of C-SLAM in which only one vehicle is responsible for maintaining estimates of the map and poses for each robot. They organize the robots in two groups: slaves (robots with limited set of sensors, both proprioceptive and exteroceptive), and masters (robots with acceptable dead reckoning capabilities and accurate exteroceptive sensors). One robot belonging to the master group, called SLAM robot, is chosen to perform the SLAM algorithm (they use the feature-based formulation) using information shared by all others. The state vector is then composed by the vehicles pose history and environment features. At each time step, all the robots send their control input and observations to the SLAM robot, which performs the prediction and update equations.

In [61], C-SLAM is considered within the Stochastic Mapping framework. Mourikis et al assume that the mobile robots move continuously and randomly in a planar environment, while recording measurements of the relative positions of other robots in the team, and of point landmarks that exist in the environment. They describe the exteroceptive measurements at each time step in a graph, called Relative Position Measurement Graph (RPMG), and obtain analytical upper bounds for the positioning uncertainty. The derived bounds provide descriptions of the asymptotic positioning performance of a team of robots in a mapping task as a function of the characteristics of the proprioceptive and exteroceptive sensors of the robots, and of the RPMG.

Bryson et al, in [15], propose a cooperative path-planning algorithm for multi-vehicle SLAM that uses information-based measures to maximize the accuracy of a feature map which is constructed from terrain observation made by each vehicle. They use a distributed formulation for the SLAM algorithm, in which the processing

occurs across multiple vehicles using information filtering. The locally built map information is then shared via a central node for communications. Each vehicle also communicates to the central node potential actions that it can take, and the expected information that it can contribute towards the central feature map for each action. The central node then computes which combination of actions will result in the highest information gain in the feature map, and sends appropriate trajectory commands to each of the vehicles.

The main challenge in a decentralized Cooperative Localization algorithm concerns communications constraints which make it hard to create and update covariance information between vehicles as they communicate with each other. This deficit of covariance information between the vehicles may lead to a repetitive use of the same evidence and to an overconfidence of the robots pose as discussed in [33]. In this case it it possible to use the algorithm introduced in [41], where the authors show a consistent, suboptimal, way to fuse estimates with unknown covariance. This is the idea used by Li et al in [50]. In [10], the authors proposed a solution to this difficulty by keeping a table of measurements and by preventing the use of any measurement more than once. To accomplish this, each vehicle runs a bank of filters (if there are n vehicles involved in the system, each vehicle will have up to 2^n filters). By tracking the origins of the measurements and preventing the use of any of the measurements more than once, the multiple estimates of the bank of filters are combined in a consistent manner, yielding conservative, suboptimal, covariance estimates. However, this approach still demands a very high communication rate, and might be unpractical in real situations.

In this chapter, we presented several methods discussed in the literature to perform Cooperative Localization using EKF and PF based estimators. However, we saw that those methods are not appropriate for our AUV Cooperative Localization

mainly due to the strict communications constraints imposed by underwater environment. In the next chapter, we will propose some solutions to this problem and analyse the properties of these solutions.

Chapter 5

Proposed Solutions to Cooperative Localization for AUVs

In the previous Chapter, we presented the problem of Cooperative Navigation, commented on a number of approaches proposed in the current state-of-the-art, and conclude that these are not appropriate to address the key challenges arising for AUVs based systems. In this Chapter, we propose methods to solve this problem which takes into account the communication constraints discussed in Chapter 4. In Section 5.1, we briefly discuss the addressed problem and organize it in several tasks to be solved. In the ensuing section, the similarities between the cooperative localization scheme and the SLAM structure are presented. Finally, in Sections 5.3 and 5.4, we present the two proposed algorithms based, respectively, on the EKF and the PF estimators.

5.1 An AUVs Cooperative Localization Scheme

In this Chapter, we address one of the main contributions of this thesis, i.e., methods for AUVs cooperative localization. As shown in Chapter 4, the main challenges encountered in the underwater environment are:

- Difficulties in the extraction of natural features, and, consequently, in the execution of a SLAM algorithm.

- Impossibility of the AUVs remaining stationary due to the fact that controllability is lost when the velocity is below a certain value. Notice, that if the thrusters are switched-off, then underwater currents and the typical positive buoyancy (for security reason, the buoyancy of AUVs are usually adjusted to be slightly positive so that the vehicle will surface when powerless), the AUV will drift away from its position.
- Severe communication constraints which consist in: (i) very low bandwidth - implying that only a small amount of data can be shared in a given time interval -, transmission delays - due to the slow propagation of sound in the water -, and (iii) poor reliability which manifests in frequent losses of data packets, and multi-paths.

With these considerations in mind, we develop methods based on estimation techniques for cooperative localization involving n AUVs based essentially in two classes of procedures: Localization algorithms, and SLAM schemes. These will be further developed in Sections 5.3 and 5.4.

The methods presented in Chapter 4 either rely on fast and/or faultless communication, or do not keep a consistent cross correlation matrix, leading to overconfidence or conservative covariance estimates. The goal of this work is to develop algorithms for cooperative localization in the presence of slow and faulty communications. Another issue that arises in the cooperative localization setup considered in this thesis, and further detailed in Chapter 6, concerns finding the spatial distribution of the vehicles that yields the best possible information from the observations and optimizes the vehicles uncertainty.

The research effort in this thesis towards the development of cooperative localization methods for AUVs based systems encompasses the following tasks:

1. Investigation and design of estimation methods for multi-vehicle that requires a as low as possible bandwidth for acoustic communication.

2. Analysis of the system's observability and stability in the case of perfect communications.
3. Analysis of the convergence of the SLAM algorithm with a dynamic map when slow and faulty communications are considered.
4. Investigation of spatial formation and communications topologies optimizing the estimation methods developed in task 1.

5.2 Cooperative Localization and SLAM

Now, we address the first task presented in Section 5.1, and analyse how the problem of developing estimation methods for multiple vehicles relates to the SLAM problem.

We already saw that, in order to perform localization for underwater vehicles, static artificial features, like beacons, limits the operation area and requires a substantial deployment effort. We consider the scenario shown in Figure 5.1 in which several vehicles cooperate in such a way that, while executing its own mission, each AUV uses navigation data from the other AUVs to improve its own pose estimate, and, at the same time, generates data to support their navigation. Given the strict constraints on onboard resources, this concept of having several AUVs cooperating in order to fulfill a given collective mission is very attractive.

For simplicity, let us consider a 2D unicycle model - which was presented in Section 3.2 and, for convenience, repeated here in Equation (5.2) - for all the vehicles involved in the system. Observe that, by using pressure sensors, it is possible to reduce the cooperative localization from a 3D to a 2D problem [8]). The state vector $X_i \in \mathbb{R}^3$ for vehicle i is composed by the position (x_i, y_i) on the plane and the orientation θ_i relative to the global reference frame, as show in Equation (5.1).

$$X_i = [x_i \; y_i \; \theta_i]^T \tag{5.1}$$

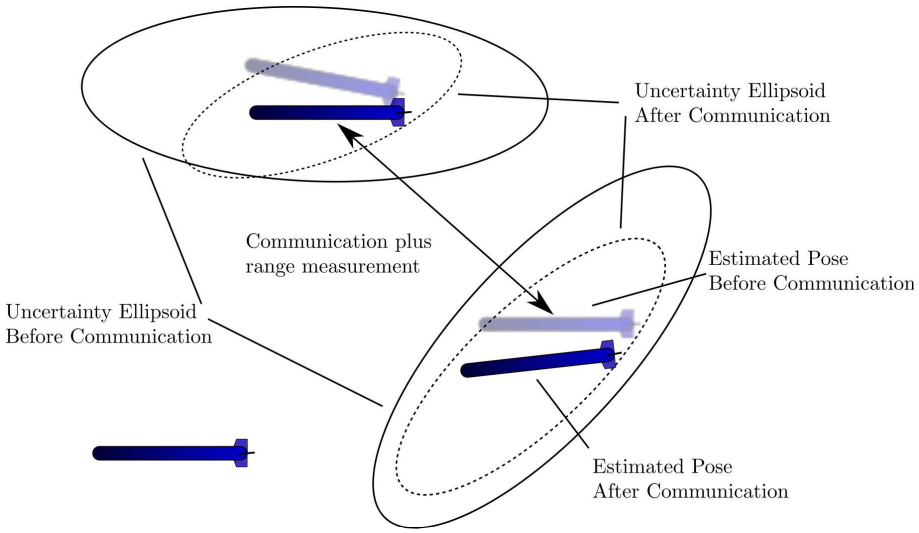


Figure 5.1: Cooperative Localization: vehicles communicate when possible, and, after sharing information, and fusing it with range measurements, they will generate new pose estimates with a lower uncertainty.

The control input is

$$u_i^{Vi}(k) = [v_x^{Vi}(k)\, v_y^{Vi}(k)\, v_\Theta^{Vi}(k)]^T,$$

where v_x^{Vi} and v_y^{Vi} are, respectively, the velocity in the x and y axes of the vehicles local reference frame $\{Vi\}$, and v_Θ^{Vi} is the angular velocity of the vehicle. As before, we consider that the control signal $u_i^{Vi}(k) = u_{in}^{Vi}(k) + q_i(k)$, where $u_{in}^{Vi}(k)$ is the nominal control of the vehicle i which is corrupted by noise $q_i(k) = [q_x(k)\, q_y(k)\, q_\theta(k)]$. The discrete time model of the vehicle's motion is given by Equation (5.2).

$$\begin{aligned} X_i(k+1) &= f_i(X_i(k), u_i^V(k)) \\ \begin{bmatrix} x_i(k+1) \\ y_i(k+1) \\ \theta_i(k+1) \end{bmatrix} &= \begin{bmatrix} x_i(k) + v_x^{Vi}(k)cos(\theta_i(k)) - v_y^{Vi}(k)sin(\theta_i(k)) \\ y_i(k) + v_x^{Vi}(k)sin(\theta_i(k)) + v_y^{Vi}(k)cos(\theta_i(k)) \\ \theta_i(k) + v_\theta^{Vi}(k) \end{bmatrix} \end{aligned} \tag{5.2}$$

If we put together the state vector of all the n vehicles into a single augmented state vector we will get $X \in \mathbb{R}^{3n}$ such that

$$X = [X_1\, X_2\, \ldots\, X_n]^T.$$

This mimics the SLAM structure presented in Section 3.2, but now, instead of one vehicle and several static features, we will have a group of moving vehicles. So, this

scenario constitutes a problem similar to the SLAM problem, with moving features as presented in section 3.4.

All the developed algorithms start by considering one AUV as a central processor unit (from now on called CAUV, whose state vector will be denoted by X_C), and the others AUVs as features on the environment (called FAUVs, denoted by X_{Fi},$i = 1, 2, ..., n$), which communicate with the CAUV. Hence, the state vector of the system will be composed of the CAUV and n FAUVs (total of $n + 1$ vehicles):

$$X = [X_C \, X_{F1} \, \ldots \, X_{Fn}]^T.$$

If it is possible to synchronize clocks, then any pair of AUVs can estimate the range to each other by computing the communication time of flight (TOF). Otherwise, the TOF can be estimated by the round-trip ping time (nowadays, there are several commercial underwater modems that give the distance from the communication pair based on the message's TOF). Therefore, the observation represents range information only. Since each FAUV is able to transmit its own current position estimate, there is always a priori knowledge of the landmark location. This eliminates the issues due to the partial observability of the range only sensor discussed in Chapter 4. While no observation is available, each AUV uses an EKF and the generated proprioceptive information - for example, via an Inertial Navigation System (INS) - to estimate its current position and update its error covariance matrix.

Remember that, as mentioned in Chapter 4, the implementation of the standard SLAM algorithm with moving features, at every step of the algorithm, all the FAUVS are required to send their control signals to the CAUV, in order to compose the control signal $u(k) = [u_C(k) \, u_{F1}(k) \, u_{F2}(k) \, \ldots \, u_{Fn}(k)]^T$, where $u_C(k)$ and $u_{Fi}(k)$ are the control signal of the CAUV and of the i^{th} FAUV at time instant k, respectively. However, this is prevented by the low acoustic communication bandwidth. Actually, several algorithm steps are already taken just for one FAUV to send his control

signal. One possible way around this difficulty, is to consider that (i) the Currently Communicating FAUV (CCFAUV) sends only its final position, or the resultant control signal during the last communication until the current moment, and (ii) all the other FAUVs are assumed static by the CAUV. In this way, the control signal can be composed, and, then, used by any of the estimation techniques presented in Chapter 2. The difficulty of this approach resides in dealing with the process noise. If the algorithm is applied by considering the process noise in one step only, then, the noise will be less significant than the real one, since the FAUV actually takes several steps to send in information. If this issue is fixed by taking into account the noise occurring during the period of time between communications, there are still some problems. Since this inter-communication time might be long, and the behavior of the FAUV within this period is unknown, the effect of noise can hardly be properly incorporated. For example, the FAUV might have moved describing a circle, coming back to a position next to its initial one (since the last communication), and, in this case, the error will have an effect on the vehicle uncertainty which is very different of the one if the FAUV were to move slower in a straight line during this same period of time, as show in Figure 5.2. Thus, it doest not suffice for the FAUV to send in only the current estimated position so that the CAUV could safely guess the associated process noise. The FAUV has to send in the whole information: current estimated pose and also the process noise accumulated since the last communication, which can be represented by the uncertainty of the current estimated position. So, the communication between the CAUV and the FAUVs has to enable sharing the pose and the pose uncertainty with one another.

The fact that the AUVs are moving constantly - i.e., they do not stop to communicate to each other - constitutes an important difficulty. Hence, when the CAUV sends in his position to a FAUV at time k_1, $X_C(k_1)$, the FAUV will receive this information at time k_2. Then, the FAUV will send in its position, $X_F(k_2)$, at time

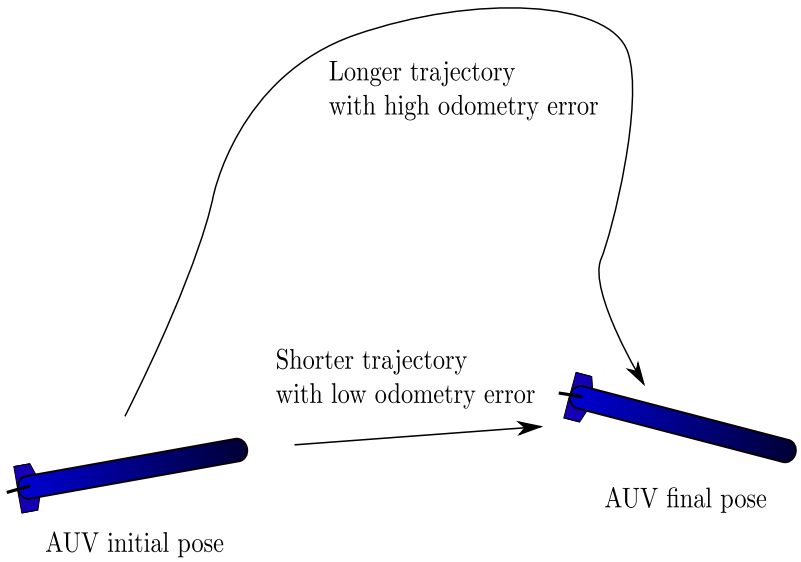


Figure 5.2: Effect of the noise: if we only care about the initial and final position of the robot, the odometric noise affecting the estimated trajectory can be quite different from the one that should have been considered. For the same endpoints, longer trajectories with a lot of curves insert bigger error then the shorter ones.

k_2 (actually at $k_2 + \epsilon$, where ϵ is the time required by the FAUV to start sending in the signal - for simplicity we will consider $\epsilon = 0$), and the FAUV response will get back at the CAUV at time k_3. During this communication period, the positions of both AUVs will have changed. If the messages between AUVs are time tagged (requiring the synchronization of the AUVs clocks), then the problem is solved, and the FAUV can send in its pose at the time instant indicated by the time tag. If the messages are not time tagged, then the CAUV will consider that the position transmitted by the FAUV will be from time instant k_1, that is, $X_F(k_1)$, which is different from the one transmitted, and different from the current one. This error depends on the distance between vehicles, velocity of sound in water and relative velocities between AUVS. If, for example, $2km$ is the maximum distance between AUVs, the maximum speed of the AUV is $1m/s$, and the speed of sound in water $1500m/s$, then in the worst case (the AUVs moving in opposite directions), the estimated time error is $|k_1 - k_2| = 1.5s$, and thus a maximum error of $3m$. This error bound can

be incorporated by adding it to the range sensor error covariance matrix $R(k)$. If you consider two consecutive measurements between the same vehicles in a short period of time, this kind of error is clearly not Gaussian, since the vehicles will still be moving at about the same direction in both measurements, and the errors will be correlated. But, if the measurements between the same vehicles are not taken within a short period of time, which is the case of our setup, we will assume Gaussian error.

Since the transmission of the whole data package - composed by the position and position uncertainties, as well as other data that may vary according to the mission - may take some time, the CAUV must use these data for filtering at the time the first signal from the FAUV was received. We designate it by k_3. This means that the CAUV must keep its past data stored, from k_3 up to the current time. Once the communication is done, the CAUV estimates its position (and uncertainty) at time k_3, and, then, propagate this new estimate to the current time. This propagation requires that the track of its own uncertainty and displacement (control signal) since k_3 are kept, and uses the KF prediction equations to compute his new estimated state and uncertainty at the current time. Notice that the amount of data stored grows with time while the vehicles are involved in a communication, but once the computation is completed, the data can be discarded. Also, as we are going to show in Section 5.3.2, there is a much more compact way - in the sense that does not grow with time - of representing the FAUV's displacement and uncertainty since time k_3.

We assume that it may be possible for one of the FAUVs to surface once in a while in order to update its position and heading trough a GPS (although GPS cannot directly measure heading, it is possible for a sequence of measurements if the AUV is moving, or using some other heading sensor, such as a compass). The surfacing FAUV should be chosen by using a mission dependent criterion, such as, distance to the surface, priority level of the currently assigned task, etc. Here, in order to account for the worst possible scenario, we consider that the CAUV is submerged

and executing a high priority task, and, therefore, can not surface to update its position. We will see later that if the CAUV is actually able to surface, this makes the estimation more efficient.

5.3 Extended Kalman Filter Based Algorithms

The EKF requires the stochastic processes associated with the relevant AUV's data to be Gaussian and, in turn, produces estimates which are also Gaussian. We know that one dimensional Gaussian distributions are represented by only two parameters: mean and variance. In the multi-dimensional case, they can be represented by, respectively, the mean vector and the covariance matrix. Thus, it is possible for the AUVs to share their filter information simply by communicating their estimated (mean) position and the error covariance matrix $P(k)$. This is the reason why the KF is so appealing for underwater applications.

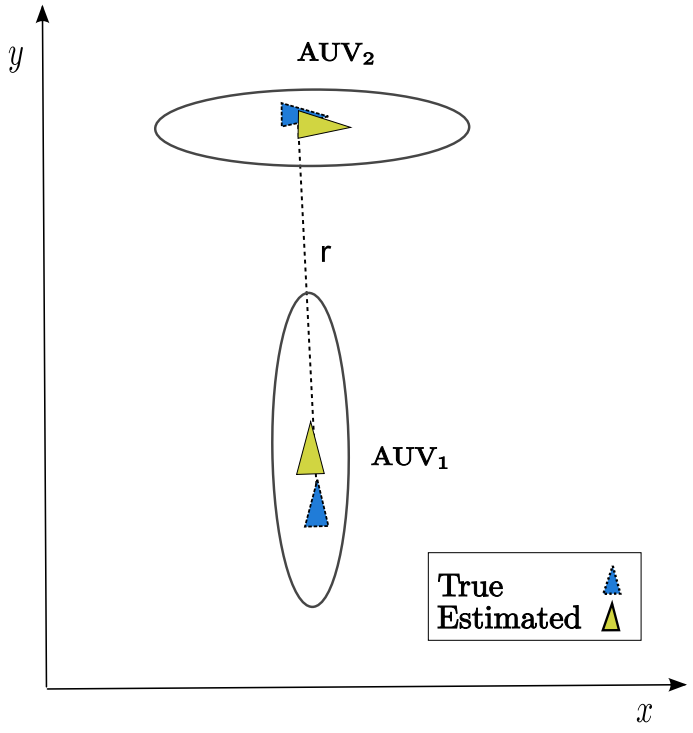
We developed two classes of algorithms using the EKF: one based on a simple localization algorithm, and another one based on the SLAM paradigm.

5.3.1 EKF Localization Based Methods

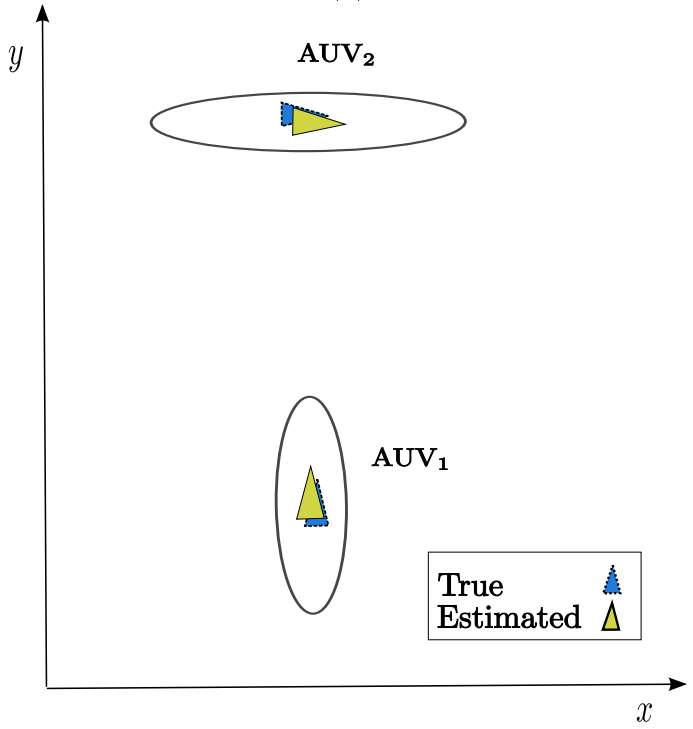
The first two algorithms, called CEKOL1 and CEKOL2 (Cooperative Localization using Extended Kalman Filter based on Localization) are based on an EKF solution for the localization problem and are inspired by the methods for cooperative localization presented in [45, 33, 75]. CEKOL2 is an evolution of CEKOL1, the first giving better results than the last. However, we chose to present both methods in this work so that the reader might follow the train of thought during the development of those methods, since we first developed CEKOL1 and then, based on its ideas, developed CEKOL2. The principle of both of them is to add to the noise of the range sensor the error of the vehicles involved in the observation. While CEKOL1 does so in a geometrical way, CEKOL2 uses statistical tools given by the

KF itself. The CAUV considers the communicating FAUV (CCFAUV) as a known feature in the environment, simply by taking into account its uncertainty. Similarly, the FAUV also considers the CAUV as a known feature. The algorithms' consistency relies strongly on the estimate of the uncertainty of AUVs pose. The idea is to compare the uncertainty information of the FAUV with the pose uncertainty of the CAUV and thus determine how much the CAUV should trust the range information and how much it should change its own pose. If, for example, the FAUV is pretty sure about its position (suppose, for example, that it surfaced recently and updated his pose by GPS), and the CAUV is not so certain, the new estimate of the CAUV pose is likely to change much more than the new estimate of the FAUV. This is clearly seen in Figure 5.3 where the range information combined with the pose and pose uncertainty affects how each one pose estimate will change. Since in Figure 5.3 (a), the range r measures basically the y axis distance between the vehicles, the new estimate pose of the AUV_1 in Figure 5.3 (b) can potentially change more than the new estimate of AUV_2. This happens because AUV_2 already has a low uncertainty in the y axes, and it is receiving range information with high error, since AUV_1 has high uncertainty in the y axes. This means that AUV_2 cannot improve much its position and position uncertainty. Conversely, AUV_1 will be able to use the range information to improve the estimate of its pose and decrease its pose uncertainties in the y axes, as show in Figure 5.3 (b). If we do not take into account the AUVs pose uncertainty, this means we consider them as known features, that is, a known map of the environment, with 100% accuracy, which, of course, is not true.

When two AUVs are communicating, all other AUVs present in the system are disregarded and no cross-correlation between them are generated. Only the positions (and associated uncertainties) of the communicating AUVs are updated. Actually, the pair of communicating AUVs do not even know that other AUVs exist. The assumption that the CCFAUV is a known feature (even considering its uncertainty)



(a)



(b)

Figure 5.3: Incorporation of uncertainties when using the range information. In (a), we can see the vehicles true and estimated pose before the observation, being their uncertainty represented by ellipses. In (b), we observe the new estimated pose and uncertainties of the vehicles after updating by using range information. Notice that AUV_1 is able to improve his pose and pose uncertainty more than AUV_2.

and the absence of cross-correlation between the AUVs lead to overconfidence in the vehicle's pose and can also affect the convergence of the system. We will return to this subject Chapter 7.

5.3.1.1 CEKOL1

The CEKOL1 algorithm updates the estimate of the pose of the CAUV and of the CCFAUV by using data provided by a range sensor. Each one of the vehicles i runs an EKF in which the state vector is constituted by its own pose, as shown in Equation (5.3) for a 2D scenario, and its error covariance matrix reflects the uncertainty of its pose estimate.

$$X_i(k) = [x(k)\, y(k)\, \theta(k)]^T \tag{5.3}$$

$$P_i(k) = \begin{bmatrix} P_{xx}(k) & P_{xy}(k) & P_{x\theta}(k) \\ P_{yx}(k) & P_{yy}(k) & P_{y\theta}(k) \\ P_{\theta x}(k) & P_{\theta y}(k) & P_{\theta\theta}(k) \end{bmatrix} \tag{5.4}$$

Once the CAUV chooses a FAUV to communicate with (randomly of following some protocol, as will be discussed in Section 6.2) the FAUV sends in its current predicted pose estimate $\hat{X}_{pCCF}(k)$ and the predicted error covariance matrix $P_{pCCF}(k)$ to the CAUV. Then, the CAUV computes the range to the Currently Communicating FAUV - from now on designated by CCFAUV - based on the communication TOF, and a measure of the CCFAUV position uncertainty is added to the range sensor error covariance $R(k)$. This value is approximated by one half of the projection of the ellipses error axes onto the line that goes from CAUV position to FAUV position, as show in Figure 5.4. This information is fed into the EKF running in the CAUV, and the filter outputs an updated estimate of the CAUV's position $\hat{X}_C(k)$ and an updated error covariance matrix $P_C(k)$. The same process occurs in the FAUV, which takes the current predicted position estimate $\hat{X}_{pC}(k)$ and predicted uncertainty $P_{pC}(k)$ transmitted by the CAUV and estimates his own new position

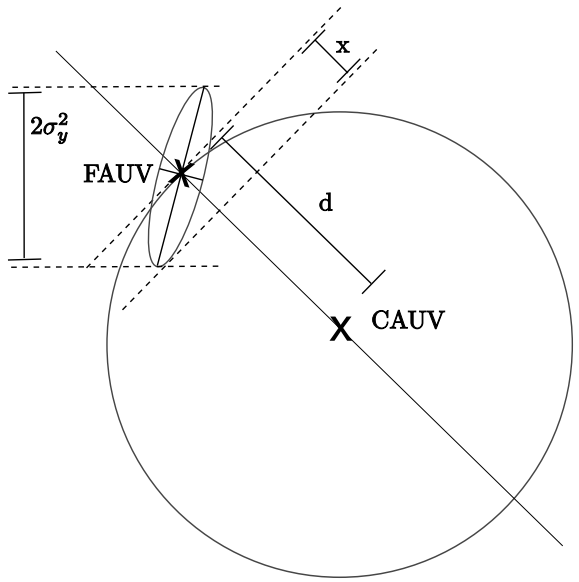


Figure 5.4: Added sensor error: the sensor tells that the CAUV is at a distance d from the FAUV with some associated variance r^2, given by the sensor specification. The considered variance is $r^2 + x$ where x is half of the the projection of the ellipses error axes onto the line connecting the CAUV and the FAUV positions.

and error covariance matrix. There is no information about the correlation of the AUVs involved, that is, we assume that there is no correlation between the estimates of the vehicles. After this process, both the CAUV and the CCFAUV will have their positions and uncertainties updated, and the another FAUV will be selected for the next CCFAUV. The CEKOL1 method is presented in Algorithm 5.3.1.1.

5.3.1.2 CEKOL2

The second algorithm - designated by CEKOL2 - provides the joint update of the pose of the AUVS and their associated error covariance matrices. Here, each vehicle also runs an EKF with its own pose, as in CEKOL1. When an observation is available, the EKF running in each one of the communicating AUVs are modified so that the state vector $X \in \mathbb{R}^6$ is composed by the pose of the CAUV, $X_C(k)$,and that of the CCFAUV, $X_{CCF}(k)$:

$$X_p(k) = [X_C(k)\ X_{CCF}(k)]^T.$$

Algorithm 5.1 CEKOL1 running on CAUV

1. Initialize State Vector and Error Covariance Matrix
2. while true
   (a) if an observation is available
      i. Add to $R(k)$ half of the projection of CCFAUV transmitted position uncertainty $P_{pCCF}(k)$ onto the line connecting the CAUV and the CCFAUV positions.
      ii. Use the observation $y(k)$ (- the distance between CAUV and CCFAUV - and CCFAUV transmitted position $\hat{X}_{pCCF}(k)$ to update current CAUV position $\hat{X}_C(k)$ and uncertainty $P_C(k)$ using EKF update equations
   (b) Use proprioceptive information to predict next CAUV position $\hat{X}_{pC}(k+1)$ and position uncertainty $P_{pC}(k+1)$ using EKF predict equations
   (c) endif
   (d) $k = k + 1$
3. endwhile

Similarly, the predicted error covariance matrix $P_p(k) \in \mathbb{R}^{6\times6}$ in the modified EKF is given by

$$P_p(k) = \begin{bmatrix} P_C(k) & 0 \\ 0 & P_{CCF}(k) \end{bmatrix},$$

where $P_C(k)$ and $P_{CCF}(k)$ are, respectively, the error covariance matrices of the CAUV and CCFAUV positions, at time instant k. We assume, as in CEKOL1, that there is no correlation between the positions of the AUVS, since the off-diagonal terms of P_p are zero. Even if the CCFAUV was observed before, the off-diagonal terms are set to zero. This false assumption will lead to the emergence of over-confidence of the robots pose when the same evidence is repeatedly used, [33]. In [10], the authors propose a solution to overcome this difficulty by keeping a table of measurements and by preventing the re-use of measurements. However, in some applications, this lack of covariance may not be harmful. For example, if we are concerned about the relative positioning of the vehicles only. Such a scenario will be presented in Section 7.5.

Notice that, in CEKOL2, since observations are based on states of the system (all operands belong to the state vector), the AUVs are not considered to be known features with 100% accuracy. In this way, their uncertainties are taken into account

Algorithm 5.2 CEKOL2 running on CAUV

1. Initialize State Vector and Error Covariance Matrix
2. While true
   (a) If an observation is available then
      i. Construct the augmented State Vector $\hat{x}_p(k)$ composed by CAUV position and CCFAUV transmitted position: $\hat{x}_p(k) = [\hat{x}_C(k)\, \hat{x}_{CCF}(k)]^T$
      ii. Construct the augmented Error Covariance matrix $P_p(k)$ composed by CAUV covariance matrix $P(k)_C$ and CCFAUV transmitted error covariance matrix $P_{CCF}(k)$
      iii. Update current CAUV and CCFAUV position and their position uncertainties, $\hat{x}(k)$ and $P(k)$, using EKF update equations and send to CCFAUV its new pose and uncertainty.
   (b) Use proprioceptive information to predict the next CAUV position $\hat{x}_{pC}(k+1)$ and position uncertainty $P_{pC}(k+1)$ by using EKF prediction equations.
   (c) Endif
   (d) $k = k + 1$
3. Endwhile

naturally by the covariance matrix $P_p(k)$. After acquiring the data from the FAUV, the CAUV uses the EKF to update the position and uncertainty of both the CAUV and FAUV at the same time. Like in the previous algorithm, the FAUV can either use the information sent in by the CAUV to proceed with the calculations to update its pose and associated uncertainty, or simply receive the updated pose and uncertainty already computed by the CAUV. In between observations, the AUVs update their state vectors and error covariance matrices in the standard mode, that is, update only its own pose and pose uncertainty in the error covariance matrix, thus discarding all the information about other vehicles. The CEKOL2 pseudo-code is shown in Algorithm 5.3.1.2, while a block diagram is shown in Figure 5.5.

5.3.1.3 Handling Communication Failures in CEKOL Algorithms

Since underwater communications are unreliable, the ability to deal with communication failures is an important requirement for the CEKOL algorithms. We will show that no additional effort is required for the CEKOL algorithms to handle communication failures.

The communication between CAUV and the FAUVs can be divided in basically

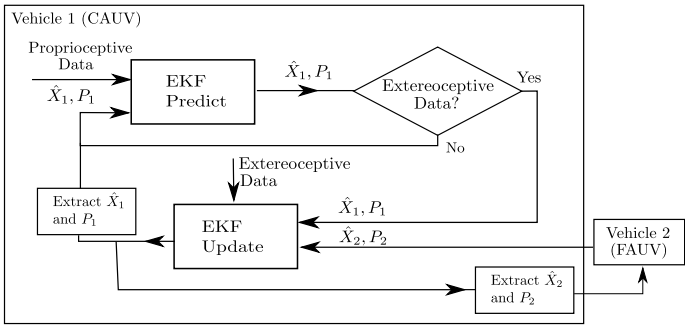


Figure 5.5: Block diagram of the CEKOL2 algorithm running in the CAUV.

3 steps:

1. CAUV invites FAUVi to communicate with it;
2. FAUVi sends information to the CAUV;
3. After computation, CAUV sends back information to the FAUVi.

A communication failure can happen in any of these three steps.

If there is a failure in step 1, then FAUVi will not receive its invitation to communicate. Therefore, the CAUV will not hear anything from it, and, if after a certain fixed time interval, no response is received by the CAUV, then it can just quit waiting for FAUVi and choose another FAUV to communicate with. This process is repeated until a successful response is received.

A failure in the second step means that the FAUV received the invitation and then sent its data to the CAUV, but the CAUV failed in receiving it. In this case, the CAUV will consider this as a failure in the first step: it terminates communication with FAUVi and chooses another FAUV to communicate with. By not receiving any response from the CAUV, the FAUVi will just continue its normal operation.

If there is a failure in the third step, the CAUV received and use the information sent in by FAUVi to update both poses, its own and the FAUV's, but was not able to successfully send it to the FAUV. The FAUV will not benefit from this observation, losing its new pose estimate computed by the CAUV (like it would if it were operating in a broadcast scheme, as presented in Section 5.3.1.4) and will

just continue its normal operation. Notice, however, that the fact that the FAUV did not benefit from the observation did not compromise the update of the CAUV's estimate.

5.3.1.4 Decentralizing CEKOL Algorithms

One major drawback of the centralized system is that if the CAUV fails or gets out of communication range the whole system is compromised. Thus, it is of interest to consider decentralized versions of the algorithms.

The two CEKOL methods based on Localization were presented here as centralized algorithms, with one of the vehicles playing the role of the CAUV which would communicate with other vehicles. But these algorithms are easily made decentralized. Since the CEKOL methods do not keep information about the cross correlation, there is no need for a CAUV, and for each one of the vehicles to not be free to communicate with each other. We may consider this happening in two distinct scenarios:

1. Pairs of communicating vehicles: in this scenario, we consider that each time a vehicle wants to communicate, it chooses another specific vehicle to communicate with. This pair of vehicles will exchange data during the execution of the CEKOL algorithm as described in Sections 5.3.1.1 and 5.3.1.2. One possible way for a vehicle to choose its pair is to broadcast a short signal saying that it is available for communication, and, then, select randomly (or following some optimization protocol, as discussed in Section 6.2) a vehicle from the set of vehicles that answered its broadcast. Once a pair of vehicles communicate, both of them will have updated their pose and pose uncertainty.
2. Broadcast communication. In this scenario, the vehicle that wants to communicate just broadcast the entire data necessary for other vehicles that are capable of listening to perform their part of the CEKOL algorithm. That is,

the communicating vehicle broadcasts its current pose estimate and covariance matrix. All the other vehicles that received this communication (some of the vehicles might be to far to listen to the broadcast or have any other communication problems) will use this information to update their own pose estimate and uncertainty with the CEKOL algorithms. Notice that, with CEKOL2, each vehicle that received the broadcast will also have a new estimate of the broadcasting vehicle, but this information is not sent back, and will just be lost.

5.3.2 EKF SLAM based methods

In this Section, we present two cooperative AUV localization algorithms based on the EKF that takes into account the low communication bandwidth by using a SLAM structure for dynamic maps. Unreliability, delays, and very low bandwidth are the key acoustic communications features that makes the design of a moving SLAM system for a group of AUVs a huge challenge. We propose an approach seeking the minimal information to be shared by the vehicles which is necessary for the execution of a SLAM algorithm, that is, required to sustain the emergence of a consistent cross correlation matrix, which is essential to ensure the SLAM convergence properties.

The two algorithms presented here - CEKOLM1 and CEKOLM2, standing by Cooperative Localization using the EKF based on SLAM - are inspired by the SLAM algorithm and uses a EKF, running on the CAUV, in which the estimated state vector $X_{SEKF} \in \mathbb{R}^{3n}$ (we recall that we are considering a 2D scenario) is the Cartesian product of the pose of all the $n+1$ AUVs in the system, i.e.,

$$X_{SEKF}(k) = [X_C(k)\ X_{F1}(k)\ X_{F2}(k) \dots\ X_{Fn}(k)]^T, \tag{5.5}$$

where X_C and X_{Fi} are, respectively, the state vector of the CAUV and of the i^{th} FAUV. We call this SLAM EKF running in the CAUV by SEKF. The SEKF takes a step every time an observation is available. The covariance matrix $P_{SEKF}(k) \in$

$\mathbb{R}^{3n\times 3n}$ of the SEKF filter, shown in Equation (5.6), contains the covariance matrix of all the vehicles in the system, and also their crosscorrelation terms. The discrete time index k is asynchronous and will increment every time the CAUV successfully communicates with a FAUV.

$$P_{SEKF}(k) = \begin{bmatrix} P_{CC} & P_{CF1} & \cdots \\ P_{CF1} & P_{F11} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{5.6}$$

In between observations, the CAUV also runs a Regular EKF algorithm (designated by REKF) in which the state vector is the pose of the CAUV, $X(j)_C = [x(j)\ y(j)\ \theta(j)]$. The goal of the REKF, also present in all FAUVs, is to predict the AUV pose and update its error covariance matrix $P(j)$ by using proprioceptive information. Notice that the discrete time index used for the REKF is j. This index is a synchronous time, incremented every Δt seconds (given by the update frequency of the proprioceptive sensors in each vehicle). Since there is only one SEKF present in the system (running on the CAUV), its state vector and covariance matrix will be represented by X_{SEKF} and P_{SEKF}. By X_i and P_i we designate the vector state and covariance matrix of the REKF running in vehicle i.

5.3.2.1 CEKOLM1

The CEKOLM1 algorithm provides the augmented state vector composed of the state of all the vehicles in the system and also the augmented error covariance matrix, which contains the cross correlation terms between vehicles. In CEKOLM1, when an observation is available (this means that the CAUV has available the distance and transmitted data from the CCFAUV), the CAUV uses its last computed pose by the SEKF, $\hat{X}_{SEKF_C}(k)$ (CAUV component, $X_C(k)$, in $X_{SEKF}(k)$, as shown in Equation (5.5)), and its current predicted pose by the REKF, $\hat{X}_C(j)$, to calculate the control signal $u_C(k)$ that took the CAUV from $\hat{X}_{SEKF_C}(k)$ to $\hat{X}_C(j)$. Similarly,

the CAUV uses the last CCFAUV computed pose by the SEKF, $\hat{X}_{SEKF_{CCF}}(k)$, and the one being transmitted by the CCFAUV, $\hat{x}_{CCF}(j)$, to compute a control signal, $u_{CCF}(k)$ that took the CCFAUV from $\hat{x}_{SEKF_{F_i}}(k)$ to $\hat{x}_{CCF}(j)$.

It is possible to consider that other FAUVs - other than the CCFAUV - are also communicating their last communicated pose with a random range measurement, but with infinite variance. Hence, their control signal will be zero, their overall uncertainty during the prediction step of the SEKF filter will be that due to the proprioceptive data of the whole motion in between communications. Thus, their range measurement will not be used by the SEKF update equations. In this way, the control signal of the SEKF is given by $u_{SEKF}(k) = [u_C(k)\ 0\ 0\ \ldots\ u_{CCF}(k)\ \ldots\ 0]^T$.

The SEKF process noise covariance matrix $Q_{SEKF} \in \mathbb{R}^{3n \times 3n}$, shown in Equation (5.7) is generated by the error covariance matrix computed by the REKF algorithms running on the CAUV and on the CCFAUV. For all other FAUVs, the process noise is zero.

$$Q_{SEKF}(k) = \begin{bmatrix} P_C(j) & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \ldots & 0 \\ 0 & \ldots & P_{CCF}(j) & \ldots & 0 \\ \vdots & \vdots & \ddots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \tag{5.7}$$

In order to ensure that $P_i(j)$ corresponds, for each AUV, only to the noise occurring from the time of last observation until the current time, this error covariance matrix is set to zero both in the CAUV and in the CCFAUV, at every time an observation is taken.

According to the prediction equation of the EKF for the REKF running in each AUV we have

$$P_p(j+1) = D_X f P(j) D_X f^T + D_u f Q(j) D_u f^T.$$

If there is no observation at time step j, then we simply set $P(j) = P_p(j)$.

Now, by reiterating, we have

$$P(j+2) = P_p(j+2) = D_X f P(j+1) D_X f^T + D_u f Q(j+1) D_u f^T,$$
$$P(j+2) = D_X f (D_X f P(j) D_X f^T + D_u f Q(j) D_u f^T) D_X f^T + D_u f Q(j+1) D_u f^T,$$
$$P(j+2) = D_X f^2 P(j) (D_X f^T)^2 + D_X f D_u f Q(j) D_u f^T D_X f^T + D_u f Q(j+1) D_u f^T.$$

Thus, we conclude that, if there are no observations between times steps j and $j+m$,

$$\begin{aligned} P(j+m) &= (D_X f)^m P(j) (D_X f^T)^m + \\ &\quad (D_X f)^{m-1} D_u f Q(j) D_u f^T (D_X f)^{m-1} + \ldots + D_u f Q(j) D_u f^T \end{aligned} \tag{5.8}$$

where, to simplify the notation, $(D_X f)^m$ means

$$(D_X f)^m = D_X f(j+m) D_X f(j+m-1) \ldots D_X f(j). \tag{5.9}$$

Since every time vehicle i is involved in an observation we set $P_i(j) = 0$, the covariance equation for the REKF running in each vehicle can be written as

$$P(j+m) = (D_X f)^{j+m-1} D_u f Q(j) D_u f^T (D_X f)^{j+m-1} + \ldots + D_u f Q(j) D_u f^T. \tag{5.10}$$

When the CCFAUV communicates with the CAUV, at time $j+m$, the SEKF prediction equation, considering only a single vehicle, for some matrix $M(m)$, is given by

$$P_{pSEKF}(k+1) = M(m) P_{SEKF}(k) M(m)^T + Q_{SEKF}(k). \tag{5.11}$$

Since we defined that the process noise covariance matrix, Q_{SEKF}, is composed by the accumulated uncertainty of the vehicles in between communications, as shown in Equation (5.7), we have

$$P_{pSEKF}(k+1) = M(m) P_{SEKF}(k) M(m)^T + P(j+m). \tag{5.12}$$

Notice that the SEKF predict equations are supposed to give the same result as the REKF equations, if we were not setting the term $P(j) = 0$ when there is an observation, as shown in Equation (5.8). By comparing Equations (5.12), (5.8) and

(5.10), observing that $P_{SEKF}(k)$ plays the role of $P(j)$, we conclude that, in order for Equation (5.12) to represent the total uncertainty for the CCFAUV at the current time (uncertainty at the last observation, $P_{SEKF}(k)$, added with the uncertainty accumulated since this last observation, $P(j)$, until now, $P(j+m)$), we must have

$$M(m) = D_x f(m) D_x f(m-1) \dots D_x f(0) = D_x f(m) M(m-1), \tag{5.13}$$

where m represents the number of time steps executed by the REKF since the last communication and the present time. Remember that the matrix $D_x f$ actually depends on $u(j)$ and $X(j)$, which vary over time. This means that each vehicle must compute the matrix M recursively during each step of the execution of the REKF and send it to the CAUV, together with $P(j)$, at the observation time. Notice that, now, right after an observation, besides setting $P(j) = 0$, we must also set $M(j) = I$ (identity matrix), so that m, in Equation (5.13), will represent the desired number of time steps.

In this way, and by considering for simplicity that we have only 2 vehicles in the system (the CAUV and one FAUV), the SEKF Prediction equation is:

$$P_{pSEKF}(k+1) = M_{12} P_{SEKF}(k) M_{12}^T + Q_{SEKF}(k)$$

where $M_{12} = diag(M_1, M_2)$ and M_i is the M matrix computed by the i^{th} vehicle. In the case of more than two vehicles, M_i is the identity matrix if the i^{th} vehicle is not the CCFAUV. The matrix M_i is a $s \,\mathrm{x}\, s$ matrix, where s is the size of the state vector of each vehicle. However, if, for example, the vehicles are modeled by a 2D unicycle model, this matrix can fully be described by only two parameters.

The reason why we cannot simply use the current covariance matrix transmitted by CCFAUV and substitute it the corresponding terms of P_{SEKF} is because, as the vehicles move, the change in the cross correlation terms (due to initial nonzero

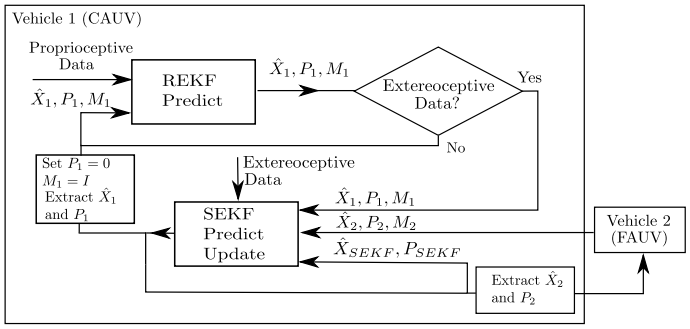


Figure 5.6: Block diagram of the CEKOLM1 algorithm running in the CAUV.

correlation) would then be ignored. As we saw in Section 3.1, since the entire structure of the SLAM problem critically depends on maintaining complete knowledge of the cross correlation between landmark estimates, this miss update of the cross correlation terms could result in inconsistent and divergent solutions to the map construction.

After receiving this information from the CCFAUV, the SEKF can compute the estimated position for all AUVs in the system by using the update equations of the EKF. After this computation, the CAUV sends to the CCFAUV its new estimated pose and pose uncertainty. The modifications of the pose of the FAUVs other than the CCFAUV will be taken into account at the time at which each one of them becomes the CCFAUV. The CEKOLM for the CAUV is shown in Algorithm 5.3.2.1, and a block diagram is shown in Figure 5.6.

The proposed algorithm can be initialized whenever the CAUV (defined before the mission starts) communicates with any other vehicle for the first time. The SEKF state and covariance matrix will be increased to add the data of the new observed vehicle. The only data needed to be known by the vehicles are their current estimated pose and pose uncertainty, given by their REKF running in each one of them.

Algorithm 5.3 CEKOLM1 running on the CAUV

1. Initialize State Vector and Error Covariance Matrix
2. While true
   (a) If an observation is available then
      i. Compute the control signal $u_C(k)$ that took the CAUV from $\hat{X}_{SEKF_C}(k)$ to $\hat{X}_C(j)$
      ii. Compute the control signal $u_{CCF}(k)$ that took the CCFAUV from $\hat{X}_{SEKF_{CCF}}(k)$ to $\hat{X}_{CCF}(j)$ and compose the augmented control signal $u_{SEKF}(k) = [u_{CAUV}(k)\,0\,0 \ldots u_{CCFAUV}(k) \ldots 0]^T$
      iii. Construct the augmented process noise matrix $Q_{SEKF}(k)$ with $P_{CAUV}(j)$ and $P_{CCF}(j)$
      iv. Update the state vector $\hat{X}_{SEKF}(k)$ and uncertainty $P_{SEKF}(k)$ using SEKF prediction and update equations
      v. Extract the computed pose by the SEKF and set it in the REKF of the CAUV, $\hat{X}_{REKF}(j) = \hat{X}_{SEKF_C}(k)$
      vi. Extract and transmit to the CCFAUV its new estimated pose $\hat{X}_{CCF}(k)$ and uncertainty $P_{CCF}(k)$
      vii. Reset the error covariance matrix $P_C(j)$ and set $M_C(j) = I$ (the CCFAUV does the same in its own REKF)
      viii. $k = k + 1$
   (b) Predict the CAUV next pose $\hat{X}_{pREKF}(j+1)$ and uncertainty $P_{pREKF}(j+1)$ by using REKF prediction equations.
   (c) Make $\hat{X}_{REKF}(j+1) = \hat{X}_{pREKF}(j+1)$ and $P_{REKF}(j+1) = P_{pREKF}(j+1)$.
   (d) Compute the matrix $M(j) = D_x f(j) M(j-1)$
   (e) Endif
   (f) $j = j + 1$
3. Endwhile

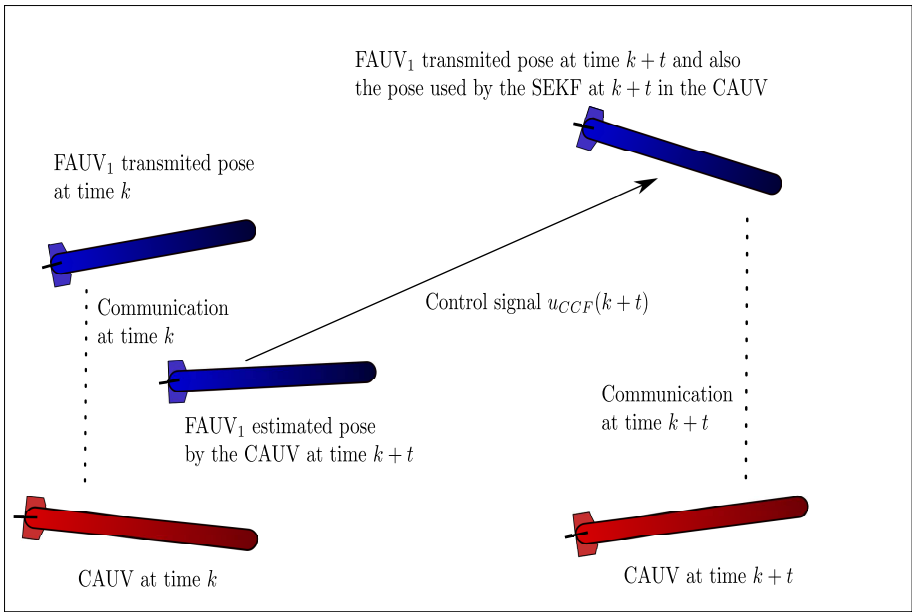
5.3.2.2 CEKOLM2

CEKOLM2 is similar to CEKOLM1, as it also uses an augmented state vector filter. However, by considering the updates in the positions of the vehicles that are not the CCFAUV, it fully uses the cross correlation created in between vehicles. In CEKOLM2, the CAUV contains a table with the last communicated pose $\hat{X}_i(k_i)$, $i = 1 \ldots n$ for each FAUV. When an observation is available, the CAUV uses the penultimate pose received from the CCFAUV, $\hat{X}_{CCF}(k_{CCF})$, which was transmitted at a prior time time k_{CCF}, in the SEKF, (the same as time $j - m$ in the REKF of the CCFAUV), and the current received one, $\hat{X}_{CCF}(j)$, to compute a control signal, $u_{CCF}(k)$ that took the CCFAUV from $\hat{X}_{CCF}(k_{CCF})$ to $\hat{X}_{CCF}(j)$. The control signal of the CAUV itself is computed just as in CEKOLM1. These

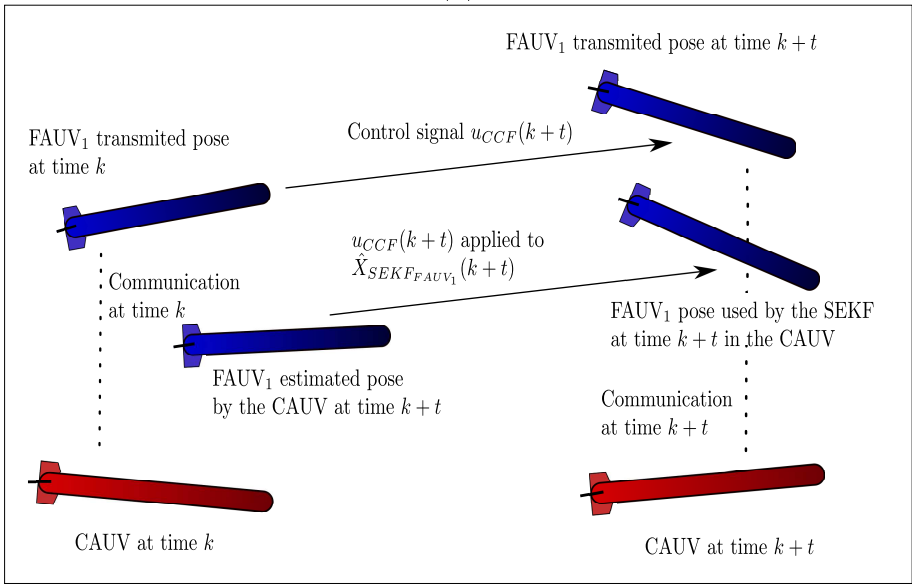
two control signals represent the accumulated motion described by the vehicles since their last communication with any other vehicle (notice that the CAUV might have communicated with other FAUVs in between communications with CCFAUV). This basically takes into account the estimates made by the SEKF on the FAUVS that are not the CCFAUV, just like a SLAM algorithm does, potentially modifying the position of all the features when one of them is observed. This process is shown in Figure 5.7. CEKOLM2 fully uses the correlations created between all AUVs in the matrix $P_{SEKF}(k)$ and when a AUV pose is updated, the pose of all others AUVs are updated as well.

Algorithm 5.4 CEKOLM2 running in the CAUV

1. Initialize State Vector and Error Covariance Matrix
2. While true
   (a) If an observation is available, then
      i. Compute the control signal $u_C(k)$ that took the CAUV from $\hat{X}_{SEKF_C}(k)$ to $\hat{X}_C(j)$
      ii. Compute the control signal $u_{CCF}(k)$ that took the CCFAUV from $\hat{X}_{CCF}(k_{CCF})$ to $\hat{X}_{CCF}(j)$ and compose the augmented control signal $u_{SEKF}(k) = [u_C(k)\,0\,0\ldots u_{CCF}(k)\ldots 0]^T$
      iii. Construct the augmented process noise matrix $Q_{SEKF}(k)$ with $P_C(j)$ and $P_{CCFV}(j)$
      iv. Update the state vector $\hat{X}_{SEKF}(k)$ and uncertainty $P_{SEKF}(k)$ using SEKF prediction and update equations
      v. Extract the computed pose by the SEKF and set it in the REKF of the CAUV, $\hat{X}_{REKF}(j) = \hat{X}_{SEKF_C}(k)$
      vi. Extract and transmit to the CCFAUV its new estimated pose $\hat{X}_{CCF}(k)$ and uncertainty $P_{CCF}(k)$
      vii. Reset the covariance matrix $P_C(j)$ and set $M_C(j) = I$ (the CCFAUV should do the same in its own REKF)
      viii. $k = k + 1$
   (b) Predict the CAUV's next pose $\hat{X}_{pREKF}(j+1)$ and uncertainty $P_{pREKF}(j+1)$ using REKF prediction equations.
   (c) Make $\hat{X}_{REKF}(j+1) = \hat{X}_{pREKF}(j+1)$ and $P_{REKF}(j+1) = P_{pREKF}(j+1)$.
   (d) Compute the matrix $M(j) = D_x f(j) M(j-1)$
   (e) Endif
   (f) $j = j + 1$
3. Endwhile



(a)



(b)

Figure 5.7: Difference between CEKOLM1 and CEKOLM2 when computing the control signal: a) CEKOLM1 control signal b) CEKOLM2 control signal. At time k there is an observation between FAUV$_1$ and CAUV. CAUV will then updates its own pose as the pose of FAUV$_1$. Notice that in between communications at k and $k+t$, $t \geqslant 1$, it is possible for the CAUV to communicate with others FAUVs, and thus change the FAUV$_1$ estimated position. Once CAUV communicates again with FAUV$_1$ at $k+t$, these modifications in FAUV$_1$ pose made in between observations are taken into account by CEKOLM2, but not by CEKOLM1.

5.3.2.3 Considerations about CEKOLM methods

The CEKOLM methods takes into account the uncertainties of the AUVs, as they are not considered to be known static features of the environment, but estimated states of the system itself, as in a SLAM formulation. This unification of the state vector allows the generation of the covariance matrix of all the vehicles in the system, and also their crosscorrelation terms.

In both EKF SLAM based methods, when a FAUV surfaces and updates its position by a GPS or any other means of global localization, it must inform this to CAUV the next time they communicate. The CAUV then must set to zero all the correlation terms that involves the CCFAUV in the SLAM covariance matrix P_{SEKF}. There is also need to set the CCFAUV pose uncertainty in P_{SEKF} according to the one transmitted by the CCFAUV.

A disadvantage of the communication scheme presented for the CEKOLM algorithms is that the communication frequency of any particular FAUV decreases as the number of vehicles on the system increases. However, as noted in [78], the benefits brought for each additional vehicle added to the system is inversely proportional to the number of vehicles, so there is not much advantage in localization by increasing the number of vehicles too much. If the application requires a large number of vehicles, then several subgroups of robots should be considered, with each group using any of the CEKOLM algorithms. Another disadvantage is that if some vehicle i performs an update, by using some kind of exteroceptive sensor, such as a compass, in between two range observations (communications in between vehicles), its matrix P_i will no longer represent accurately the uncertainty accumulated since the last communication. It is possible to also update both P_i and M_i whenever there is an extra observation in between range measurements. However, even though, it will not adequately represent the accumulated uncertainty in between communications,

since the observation would give different results when applied to P_i (the accumulated uncertainty o vehicle i since last range observation) and to the term P_{Fii} of the matrix P_{SEKF} representing the current uncertainty of vehicle i. So, in this case, the formulation of CEKOLM is no longer exactly the same as receiving all the information of the system during all time steps of the algorithm, but it will only give an approximation. However, as we will show in Chapter 7, CEKOLM is still able to greatly reduce the error of the system in these cases.

The CEKOLM algorithm may be used just for relative positioning with respect to the CAUV. This amounts to consider that the CAUV is certain about its position, that is, its uncertainty $P_C = 0$ before any observation. Notice that, in this case, there will be no cross correlation between the FAUVs (they will be zero), and so the CEKOLM reduces to the CEKOL algorithms, since there is no need to keep the cross correlation terms, hence no need for the SEKF running on the CAUV.

5.3.2.4 Handling Communication Failures in CEKOLM Algorithms

Since the underwater communication is unreliable, it is important that the CEKOLM is able to handle communications failures. The communication between CAUV and FAUVs can be organized in three steps:

1. CAUV invites the communication of FAUVi.
2. FAUVi sends information to the CAUV.
3. After updating both position estimates, the CAUV sends back information to the FAUVi.

A communication failure can happen in any one of these three steps. Now, let us analyze how to solve a failure in each on of them.

If there is a failure in step 1, then the FAUVi will not receive its invitation to communicate. Therefore, the CAUV will not hear any response from it, and, after waiting a certain fixed time interval, it simply quits waiting for the FAUVi. Then,

the CAUV chooses another FAUV to communicate with, and repeats this process until a successful response is received.

A failure in the second step means that the FAUV received the invitation, and, then, sent its data to the CAUV, but the CAUV did not received it. The CAUV will consider this as a failure in the first step, and will quit communication with FAUVi. Then , it will choose another FAUV to communicate with. The FAUVi will not know whether the CAUV received its information or not, and, thus, it must consider the two possibilities. Before setting its covariance matrix P to zero and its matrix M to identity, the FAUV will store copies P_{klt} and M_{klt} of both these matrices (here, index klt indicates the last time step the FAUV transmitted data) and keep updating them, as if no communication had taken place at all. The next time the CAUV invites the FAUV i, it must send along the invitation the last time kli he received and used information of vehicle i to update the SEKF. In this way, the FAUV will be able to know what copies of the couple of matrices P and M it must send to the CAUV (the ones that represent the accumulated uncertainty and vehicle dynamics since kli), and all the other copies will be discarded. Notice that for FAUV i to send the time tag kli does not require the synchronization of clocks between vehicles. This time tag can be thought of (or replaced by) a serial number, which is particular to each vehicle. There is no need for each vehicle to keep track of the time tag (or serial number) of the messages of the others. We must also modify the CEKOLM algorithm slightly: instead of sending in its current estimated position $\hat{X}_{Fi}(k)$, the FAUV will send in its displacement since kli, that is $\hat{X}_d = \hat{X}_{Fi}(k) - \hat{X}_{Fi}(kli)$.

If there is a failure in the third step, the CAUV received and used the information sent in by FAUVi to update the SEKF, and memorized the current time as kli (last time information from FAUVi was used). The FAUV will not receive any response from the CAUV and, not knowing whether there was a failure in either the second

or the third communication step, it will take the same precautions as in the case of a failure in the second step. The next time it receives an invitation from the CAUV, it will also receive the time instant kli, and will choose the appropriate matrices P and M and compute the displacement $\hat{X}_d$ to be sent in to the CAUV. Notice that the FAUV will have its pose and uncertainty updated only after having received the information from the CAUV. However, once a response is received, it will have the current best possible pose estimate.

By incorporating the procedures presented above, the CEKOLM algorithm is able to handle communication failures without compromising its internal consistency.

5.3.2.5 Decentralizing CEKOLM Algorithms

The decentralization of CEKOLM algorithms is harder than decentralizing CEKOL algorithms, since in the former, only one vehicle keeps an augmented filter, which we called SEKF, with the cross correlation information of all the vehicles in the system. The key idea to decentralize the CEKOLM algorithm is to keep the cross correlation information of the whole system in each vehicle, without a great increase in the communication burden.

The decentralized version of CEKOLM, which we will call DCEKOLM, will consider that the vehicles are able to broadcast a message. All the vehicles will have a SEKF running on them, as also the usual REKF. At the beginning of the operation, all the vehicles must broadcast their initial position and uncertainty so that each vehicles will have the same initial state of the SEKF.

We can assume that the vehicles are numbered from 1 to n. First the vehicle 1 will play the role of the CAUV and chooses (randomly or according to some given rule) another vehicle v to observe. It sends in an invitation to vehicle v, and, if accepted, then both vehicles will broadcast their pose X_1 and X_v, their accumulated error

P_1 and P_v since their last observation given by the REKF running on each one of them along with their matrices M_1 and M_v and the observation itself (measured distance between them). The amount of information transmitted is approximately the double of the one for the centralized CEKOLM. As in CEKOLM, both vehicles involved in the observation will set to zero their accumulated error P_1 and P_v after the observation.

After receiving the broadcast information, all the vehicles of the system (including vehicle 1 and v) will update their SEKF using this information only (vehicles other than 1 and$v$ will disregard all local information about their estimation since their last observation). In this way, the SEKF in each one of the vehicles will be exactly the same. Once this is complete, it is time for vehicle 2 to play the role of the CAUV and choose another vehicle to observe, and the process goes on cycling through all vehicles.

One problem that might arise is if there is some communication failure and a certain vehicle does not receive successfully the broadcasted information. In this case, this vehicle would not be able to update its SEKF. To solve this difficulty, a reception acknowledgment is requested from each vehicle and, in the case the information had not been received by one of them, resend it (this could be done be another vehicle closer to the one that did not receive). However, this might make the system too slow, and, thus another way to deal with communication failures should be considered.

Here, we outline briefly the idea of how this could be done. Include in every message broadcast by each vehicle a serial number of the observation: a natural number which is incremented after each observation. If a vehicle does not receive a broadcast, its serial number will be smaller than the expected one when it receives the next broadcast. When its turn to play the role of CAUV arrives and it chooses another vehicle to observe, he can ask this vehicle to resend the lost information

(which has to be kept by all the vehicles). With the information that has been kept and the missing one just received, the vehicle is able to reconstruct the current state and have a perfect copy of the SEKF running in the other vehicles that successfully received all the broadcast information.

5.4 Particle Filter Based Algorithm

The Particle Filter (PF) has the advantage to accurately represent non-Gaussian distribution, or even pdf which are not analytical. It also does not make any special assumption on the system model, such as linearity. Since most vehicle models are non-linear, and as commented in Section 4.2 the underwater range measurements noise sources might not be Gaussian, the PF becomes an attractive filtering technique for the Cooperative Localization problem. However, the downside of the PF is the amount of data necessary to represent the pdf. Since the vehicles need to communicate their estimates and uncertainties to one another, this poses a huge challenge to this method.

In our implementation of the Cooperative Localization using PF, called CPF, each vehicle runs independently a PF with the state vector consisting of its own pose $X_i(k) = [x(k)\, y(k)\, \theta(k)]^T$.

As shown in Section 2.1.3 the Particle Filter represents the desired estimated state and its distribution by a set of m weighted samples or particles $\{x_k^i, w_k^i\}_{i=1\cdots m}$, and as the number of particles increase the approximation of the pdf becomes closer to the true one. The problem of using a this approach to perform cooperative underwater localization is the impossibility to share the entire set of particles and weights between vehicles. The transmission of this set using underwater communications is not feasible.

To solve this problem, when the CCFAUV sends in the information to the CAUV, it compress its set of particles and weights by sending only their mean value X_{im}

and covariance P_i. Similar to CEKOL1 (Section 5.3.1.1), the CAUV consider the CCFAUV to be a static feature of the environment, and adds its uncertainty to the sensor error covariance $R(k)$ according to Figure 5.4. With this information, the CAUV can then update the weight of each of its particles according to the PF Equations (2.26). The CAUV can also compress and transmit its own set of particles to the CCFAUV, so it can update the weight of its own set of particles.

This means that we only use some part of the advantages of the PF. The vehicle that sends the information actually converts its pdf to a Gaussian. Only the receiving vehicle fully uses the particle representation of his own pdf. Also, like the algorithms based on Localization presented in Section 5.3.1, CPF does not create or keep any cross correlation between vehicles, leading to the overconfidence problems cited in Section 5.3.1.

Chapter 6

Information Driven Methods and Applications

In the previous Chapter, we proposed some algorithms to perform cooperative localization for AUVs. In this Chapter, we will consider, in Section 6.1, optimization techniques in order to improve estimation in those algorithms by controlling the vehicles spatial distribution, and, in Section 6.2, by controlling the communication protocol. In Section 6.3, we describe an example of a real application scenario where the developed algorithms are employed.

6.1 Optimal Spatial Distribution

In previous chapters, we considered the case of the motion of each vehicle per se, without any concern for its position relatively to the others. However, the relative position of the vehicles is of utmost importance in many applications, and, therefore, it is of interest to control the motion of the vehicles in such a way that the overall functional constraints of the system are satisfied and the behavior of the cooperative localization algorithms is optimized. For example, when the group is moving from one region to another in formation, or when they are searching a given area, the vehicles have to position relatively to one another according to some pattern which might be instantiated in multiple ways.

We would like to exploit this freedom of relative positioning in a way that optimizes the CEKOLM algorithms. We know that an observation between two vehicles in a certain relative position will give us more information than in others. So, the idea is to determine the spacial distribution of the vehicles that maximize the information given by the observations.

In [103], it is shown that a locally optimal strategy for observations is the one for which the determinant of the prediction variance in the Extended Kalman Filter equations is maximized. Robert Sim uses this strategy in [81] to determine the trajectory of a vehicle equipped with bearings-only sensors observing known features. In [9], the authors analyze the best position of a vehicle relatively to two features with some associated errors by taking range measurements to each feature. In this work the range information is used to update the position of the vehicle only. In [87], Stankovic *et al* propose a set of discrete time extremum seeking algorithms in the stochastic context and show how the proposed algorithms can be applied to mobile sensors as a tool for achieving optimal observation positions.

We will provide an analysis of the problem where there are several vehicles involved, and the range information is used to update the position of all vehicles. Based on the uncertainty of each vehicle and its correlation with other vehicles, we determine the best relative vehicle formation that maximizes the information extracted from observations.

In [18] and [91] the authors consider that the planned paths of what they call Survey AUV (a concept similar to our FAUV) are known, and, based on this information, they compute the optimal trajectory of what they call Beacon AUV (AUV equipped with better navigational sensors) so that the estimated position error across all the Survey AUVs are minimized over the mission. In our case, we will consider that the vehicles have no *apriori* knowledge of each others planned path, so we will compute the optimal relative position of the vehicles at a given time t

with the current information available.

It has been shown by Peter Whaite and Frank P. Ferrie in [103] that a locally optimal strategy for observations maximizes $Det(L) = |L|$, the determinant of the prediction variance in the EKF equations, where

$$L(k) = D_X h P_p(k) D_X h^T + R(k). \tag{6.1}$$

The term L, defined by (6.1), is a function of $R(k)$, $D_X h$ - which in turn is a function of the predicted state estimate $\hat{X}_p$-, and of the predicted covariance matrix P_p. So, the optimal spatial configuration will be achieved by determining the position of the vehicles in the system that maximizes $|D_X h P_p(k) D_X h^T + R(k)|$ and, then, commanding the vehicles to this position.

6.1.1 Optimal Observation with Range-only Measurements

6.1.1.1 Single Vehicle

First, let us consider the case of a single vehicle observing a static and known feature on the environment, located at the origin, i.e., $X_f = (0, 0)$. Let us compute the pose $X_v = (x_v, y_v, \theta_v)$ that the vehicle should be to maximize $|L|$.

The observation, which is range only, is given by the distance between the vehicle and the feature:

$$h = \sqrt{(x_v - x_f)^2 + (y_v - y_f)^2}.$$

The Jacobian of h w.r.t x, $D_X h$ is then given by

$$D_X h = \begin{bmatrix} \frac{x_v - x_f}{h} & \frac{y_v - y_f}{h} & 0 \end{bmatrix}.$$

Since the feature is at the origin, then

$$h = \sqrt{(x_v)^2 + (y_v)^2},$$

and

$$D_x h = \begin{bmatrix} \frac{x_v}{h} & \frac{y_v}{h} & 0 \end{bmatrix}.$$

The prediction covariance matrix P_p of the vehicle is given by

$$P_p = \begin{bmatrix} P_{xx} & P_{xy} & P_{x\theta} \\ P_{yx} & P_{yy} & P_{y\theta} \\ P_{\theta x} & P_{\theta y} & P_{\theta\theta} \end{bmatrix}.$$

It is possible that the measurement noise covariance matrix $R(k)$ depends on the pose (x, y, θ) of the vehicle (in this section, we use, from now on, (x, y, θ) instead of (x_v, y_v, θ_v) and P instead of P_p). One of the main sources of erros of underwater accoustic range sensors is the incorrect estimation of the speed of sound in seawater, which depends on several parameters such as temperature, salinity and pressure of seawater, [19]. Hence, typically, the shorter the distance between the vehicle and the feature, the smaller the error is. This would mean that the ideal position of the vehicle is as close as possible to the origin (feature position). Thus, a lower bound on the distance d of the vehicle to the feature is required.

Lemma 1. *The value of $|L|$ does not depend on the vehicle's orientation θ and the optimal value of $|L|$ is achieved when the vehicles are positioned so that the largest axis of its error ellipsoid (ellipsoid defined by the covariance matrix P that shows the region of confidence of the estimated state) is co-linear with the feature position.*

Proof. The optimization problem can be stated as follows:

$$\text{(P) Maximize } |L(x, y)| \text{ subject to } h(x, y) = d,$$

where $h(x, y) = (x^2 + y^2)^{\frac{1}{2}}$ and

$$|L(x, y)| \;=\; \frac{1}{h^2(x, y)} \left(x^2 P_{xx} + xy(P_{xy} + P_{yx}) + y^2 P_{yy}\right) + r(x, y),$$

where $r(x, y)$ is the term originated by the covariance of the range sensor (which we assumed depends on the position of the vehicle). We call (x^*, y^*) a solution to (P) if, there exists $\lambda \neq 0$ such that $\nabla \mathcal{L}(x^*, y^*, \lambda) = 0$, where

$$\mathcal{L}(x, y, \lambda) = |L(x, y)| + \lambda(h(x, y) - d).$$

In what follows, L^*, h^*, and r^* denote the corresponding function evaluated at (x^*, y^*) , and λ is a scalar multiplier. A straightforward computation gives

$$0 = \frac{\partial \mathcal{L}^*}{\partial x} = \frac{\partial r^*}{\partial x} + \frac{x}{h^*}\lambda + \frac{1}{h^{*2}}\left(2x(P_{xx} - |L^*| + r^*) + y(P_{xy} + P_{yx})\right).$$

Similarly for y,

$$0 = \frac{\partial \mathcal{L}^*}{\partial y} = \frac{\partial r^*}{\partial y} + \frac{y}{h^*}\lambda + \frac{1}{h^{*2}}\left(2y(P_{yy} - |L^*| + r^*) + x(P_{xy} + P_{yx})\right).$$

The general solution is obtained is given by $A\,col(x^*, y^*) = -\nabla r^*$ where the matrix A is defined by:

$$A = \begin{bmatrix} \frac{2(P_{xx} - |L^*| + r^*)}{h^{*2}} + \frac{\lambda}{h^*} & \frac{P_{xy} + P_{yx}}{h^{*2}} \\ \frac{P_{xy} + P_{yx}}{h^{*2}} & \frac{2(P_{yy} - |L^*| + r^*)}{h^{*2}} + \frac{\lambda}{h^*} \end{bmatrix}.$$

If r does not depend on (x, y) at the optimal, then the solution is a nonzero vector in a nontrivial null space of A. The non-triviality of the null space of A determines the value of the multiplier λ.

If we also have that $P_{xy} = P_{yx} = 0$, then the value of the multiplier λ may be chosen to obtain either $x^* = 0$ or $y^* = 0$ depending on whether the relation between P_{xx} and P_{yy}.

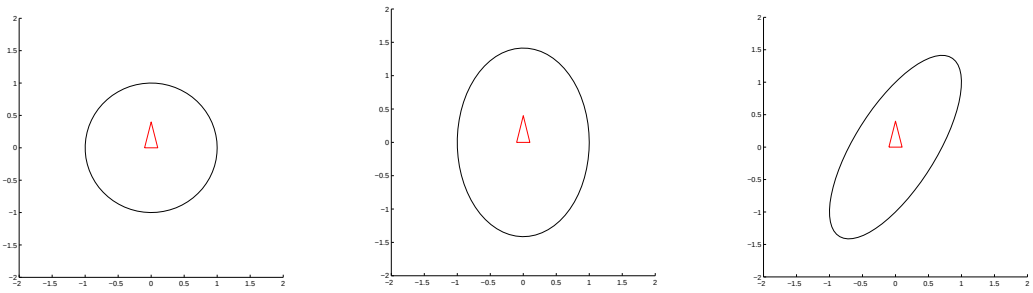
Suppose now that $R(k)$ is a constant function of (x, y). Then, it can be left out of the optimization problem. According to [10], this assumption is realistic, since many experiments have shown that the error in the range measurement is only weakly range-dependent and can be modeled as a Gaussian with zero mean and a fixed variance σ^2. If we write $P_{xx} = cP_{yy}$, $c \in \mathbb{R}$ and $P_{yy} \neq 0$, then, according to Equation (6.1), and recalling that $P_{xy} = P_{yx}$, we will have

$$\begin{aligned} |L| &= \begin{bmatrix} \frac{x}{h} & \frac{y}{h} & 0 \end{bmatrix} \begin{bmatrix} cP_{yy}, & P_{xy} & P_{x\theta} \\ P_{yx} & P_{yy} & P_{y\theta} \\ P_{\theta x} & P_{\theta y} & P_{\theta\theta} \end{bmatrix} \begin{bmatrix} \frac{x}{h} \\ \frac{y}{h} \\ 0 \end{bmatrix}, \\ &= \frac{P_{yy}(cx^2 + y^2) + P_{xy}(2xy)}{x^2 + y^2}. \end{aligned}$$

Notice that the orientation θ of the vehicle does not influence $|L|$. Now, let us consider the case in which the vehicle's x and y variables are not correlated, that is $P_{xy} = P_{yx} = 0$. If we consider that in the present time $c = 1$, which implies that the vehicles has a uniform directional uncertainty, as show in Figure 6.1 (a), we conclude that

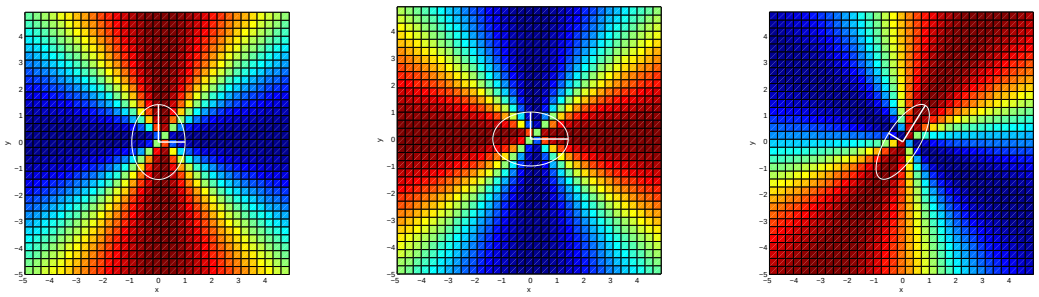
$$|L| = \frac{P_{yy}(x^2 + y^2)}{x^2 + y^2} = P_{yy},$$

which means, as expected, that the determinant of the prediction variance is constant, independent of the vehicle's position. That is, an observation from a certain



(a) (b) (c)

Figure 6.1: Uncertainty in vehicle's position represented as an ellipsoid (the axis of the ellipse are the standard deviation of the vehicle's uncertainty) a) Round uncertainty b) Elliptical uncertainty without any correlation c) Elliptical uncertainty with some correlation.



(a) (b) (c)

Figure 6.2: Computation of the determinant of L (red represents greater values) for 3 cases: a) Vehicle's uncertainty without any correlation and the uncertainty in the y coordinate b) Vehicle's uncertainty without any correlation and the uncertainty in the x coordinate c) Vehicle's uncertainty with some correlation and different values for the axes (uncertainty not round).

position is as good as from any other. Now, if we consider the case where $c \neq 1$ we shall have two different possibilities:

If $c < 1$ than the vehicle has more uncertainty about its y coordinate then about his x coordinate (both coordinates in the world reference frame). In this case, $|L|$ is maximum when $x = 0$ (optimal position when vehicle is on the y axes). So, as expected, the optimal strategy, is to position the vehicle in order to decrease the vehicle's uncertainty in the y axes. The graphic of $|L|$ as a function of (x, y) for this case is shown in Figure 6.2 (a).

The case for which $c > 1$ is dealt with in the same way. The conclusions are as in the previous case but with the roles of x and y switched. The graphic of $|L|$ as a function of (x, y) for this case is shown in Figure 6.2 (b).

Suppose now that the terms $P_{xy} = P_{yx} \neq 0$. We know that the correlation

between the x and y variables represents a rotation in the error ellipsoid of the vehicle, as shown in Figure 6.1 (c). In this case, the optimal strategy is to position the vehicle so that the greater axis of its error ellipsoid is co-linear with the origin (feature position), as show in Figure 6.2(c). □

6.1.1.2 Multiple Vehicles

In the case of a number of vehicles equal to n, $|L| = f(x_{v1}, y_{v1}, x_{v2}, y_{v2}, ..., x_{vn}, y_{vn})$, that is, $|L|$ is a function of (x_{vi}, y_{vi}), $1 \leq i \leq n$, the position of each vehicle. Even if we consider a limited search region for the optimal position for each vehicle, the complexity of the algorithm to determine the maximum of $|L|$ by searching all the values within the considered region will be $O(c^{2n})$, where c is the number of possible positions considered in each axis (in 2D). We can see that, for several vehicles, the complexity of the algorithm is such that it is very difficult its execution in real time. However, we can see from the single vehicle scenario that the actual position of the vehicle is not relevant (considering the covariance of the sensor is independent on vehicle position), but only the bearing relative to the feature, that is, the angle formed between the vehicle, the origin (feature position) and the x axis. Does this relationship also holds true for several vehicles? Next, we will show that this is case.

Lemma 2. *The computation of $|L|$ depends only on the relative bearing of the vehicles (angle formed between the vehicles, the origin and the x axis).*

Proof. First remark that there is no loss of generality if one of the vehicles is considered the origin of the relative position reference frame. For the sake of simplicity, let us consider the case of 3 vehicles involved (the extension to n vehicles is straightforward). Now, the state vector will be $X = [x_{v1}\, y_{v1}\, \theta_{v1}\, x_{v2}\, y_{v2}\, \theta_{v2}\, x_{v3}\, y_{v3}\, \theta_{v3}]^T$, the observation matrix will be

$$h = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{v1} - x_{v2})^2 + (y_{v1} - y_{v2})^2} \\ \sqrt{(x_{v1} - x_{v3})^2 + (y_{v1} - y_{v3})^2} \end{bmatrix},$$

and

$$D_X h = \begin{bmatrix} \frac{x_{v1}-x_{v2}}{h_1} & \frac{y_{v1}-y_{v2}}{h_1} & 0 & -\frac{x_{v1}-x_{v2}}{h_1} & -\frac{y_{v1}-y_{v2}}{h_1} & 0 & 0 & 0 & 0 \\ \frac{x_{v1}-x_{v3}}{h_2} & \frac{y_{v1}-y_{v3}}{h_2} & 0 & 0 & 0 & 0 & -\frac{x_{v1}-x_{v3}}{h_2} & -\frac{y_{v1}-y_{v3}}{h_2} & 0 \end{bmatrix}.$$

Without any loss of generality, we consider one of them as being fixed and compute the position of the other vehicles relative to the fixed one. By considering that the

first vehicle is at the origin $(x_{v1}, y_{v1}) = (0, 0)$, then

$$h = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_{v2}^2 + y_{v2}^2} \\ \sqrt{x_{v3}^2 + y_{v3}^2} \end{bmatrix},$$

$$D_X h = \begin{bmatrix} \frac{-x_{v2}}{h_1} & \frac{-y_{v2}}{h_1} & 0 & \frac{x_{v2}}{h_1} & \frac{y_{v2}}{h_1} & 0 & 0 & 0 & 0 \\ \frac{-x_{v3}}{h_2} & \frac{-y_{v3}}{h_2} & 0 & 0 & 0 & 0 & \frac{x_{v3}}{h_2} & \frac{y_{v3}}{h_2} & 0 \end{bmatrix}.$$

Now, for each vehicle i, suppose that $y_{vi} = a_i x_{vi}$, $a_i \in \mathbb{R}$. We want to prove now that $|L|$ is a function of a_i (and not of both y_{vi} and x_{vi}). Now we have

$$D_X h = \begin{bmatrix} \frac{x_{v2}}{h_1} & 0 \\ 0 & \frac{x_{v3}}{h_2} \end{bmatrix} \begin{bmatrix} -1 & -a_2 & 0 & 1 & a_2 & 0 & 0 & 0 & 0 \\ -1 & -a_3 & 0 & 0 & 0 & 0 & 1 & a_3 & 0 \end{bmatrix} = M_1 M_2.$$

Similarly, $D_X h^T = M_2^T M_1^T$, so

$$L = M_1 M_2 P M_2^T M_1^T + R,$$

The matrix $M_2 P M_2^T$ is a 2x2 matrix which is a function of a_2, a_3 and P. Lets call

$$M_2 P M_2^T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \tag{6.2}$$

Then

$$\begin{aligned} L &= M_1 \begin{bmatrix} A & B \\ C & D \end{bmatrix} M_1^T + R, \\ &= \begin{bmatrix} \left(\frac{x_{v2}}{h_1}\right)^2 A & \frac{x_{v2}}{h_1}\frac{x_{v3}}{h_2} B \\ \frac{x_{v2}}{h_1}\frac{x_{v3}}{h_2} C & \left(\frac{x_{v3}}{h_2}\right)^2 D \end{bmatrix} + \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}, \\ |L| &= \left(\left(\frac{x_{v2}}{h_1}\right)^2 A + r_1\right)\left(\left(\frac{x_{v3}}{h_2}\right)^2 D + r_2\right) \\ &\quad - \left(\frac{x_{v2}x_{v3}}{h_1 h_2}\right)^2 BC \qquad (6.3) \\ &= \frac{x_{v2}^2 x_{v3}^2}{(x_{v2}^2 + a_2^2 x_{v2}^2)(x_{v3}^2 + a_3^2 x_{v3}^2)}(AD - BC) \\ &\quad + \frac{x_{v2}^2 A r_2}{(x_{v2}^2 + a_2^2 x_{v2}^2)} + \frac{x_{v3}^2 D r_1}{(x_{v3}^2 + a_3^2 x_{v3}^2)} + r_1 r_2, \qquad (6.4) \\ &\quad \frac{(AD - BC)}{(1 + a_2^2)(1 + a_3^2)} + \frac{A r_2}{(1 + a_2^2)} + \frac{D r_1}{(1 + a_3^2)} + r_1 r_2. \end{aligned}$$

So we can conclude that $|L|$ is a function of a_i, R and P.

This means that $|L|$ depends only on the angle of the vehicles relative to each other. □

With this simplification, the algorithm complexity is now $O(c^n)$, where c is the number of possible angles considered and n is the number of vehicles. Even though, the algorithm is still not practical to search the entire space for a large number of vehicles. So we must use some numerical method to find the solution.

In our implementation, the algorithm to determine the optimal spatial distribution of the vehicles is computed by the CAUV. The CAUV considers itself to be at the origin, and computes the optimal position (actually the optimal angle θ_{max}) relative to itself that the CCFAUV should be. Once θ_{max} is found, it is sent to the CCFAUV.

For the CCFAUV to be able to compute the real position it should drive to, it must know the estimated position of the CAUV. The CAUV must also transmit its current heading and velocity, so that the CCFAUV will have and idea about the future positions it might occupy while it is not communicating with the CAUV. Hence, in between observations, each FAUV assumes that the CAUV is moving in a straight line with constant velocity. If the CAUV is not describing a trajectory close to a straight line (the "straightness" of the trajectory should be related to the frequency with which the CAUV observes the FAUVs), the method will not position the FAUVs in the optimal position, and thus give little advantage, as will be shown in Chapter 7.

One problem that arises with commanding the FAUVs to the optimal positions relative to the CAUV is that, in this way, they will tend to perform longer trajectories than if they were just keeping the same relative position to the CAUV all the time. This additional motion will not only increase the uncertainty of the system, but also will increase power consumption which is a critical issue for AUVs. Therefore, in order to save power, we need to compute minimum power consumption trajectories. A similar effect was observed by Robert Sim in [81], where he presents a study of a bearings-only localization system. Because of this, it is important that the control

laws applied to drive the vehicles to the respective target positions are such that the traveled distances are minimized. Next, we will present a modified control law to do so.

6.1.2 Control Law

In this section, we will present a method to synthesize a control law that minimizes the distance traveled by the vehicle to reach the specified target position. This optimization has advantages of requiring less power consumption and producing trajectories with minimal process error injected into the localization system, and, thus, yielding minimal uncertainty on the final vehicle's pose estimate.

In [1], Aicardi *et al* proposed some very simple and effective closed loop control law for unicycle vehicles by using Lyapunov control theory. Consider that the position of the vehicle $X_v = (x_v, y_v, \theta_v)$ relatively to the target position $X_{ref} = (x_r, y_r)$ represented in polar coordinates (as depicted in Figure 6.3) given by

$$\begin{aligned} R &= \sqrt{(x_v - x_r)^2 + (y_v - y_r)^2}, \\ \alpha &= atan((y_r - y_v)/(x_r - x_v)) - \theta_v. \end{aligned}$$

The control signals v (linear velocity along θ_v) and w (angular velocity) are given by

$$\begin{aligned} v &= K_1 R\cos(\alpha), \\ w &= K_2\alpha + K_1\frac{\sin(\alpha)\cos(\alpha)}{\alpha}(\alpha + K_3\theta_v), \end{aligned} \tag{6.5}$$

where K_1, K_2 and K_3 are positive gains. Notice that, due to *atan* function, this control law becomes singular when the vehicle is very close to the reference. Thus, the controller must be turned off in those situations.

This control law assumes that the vehicle can have non-positive linear velocity, [1]. But, since this is usually not possible for AUVs, if the computed velocity is

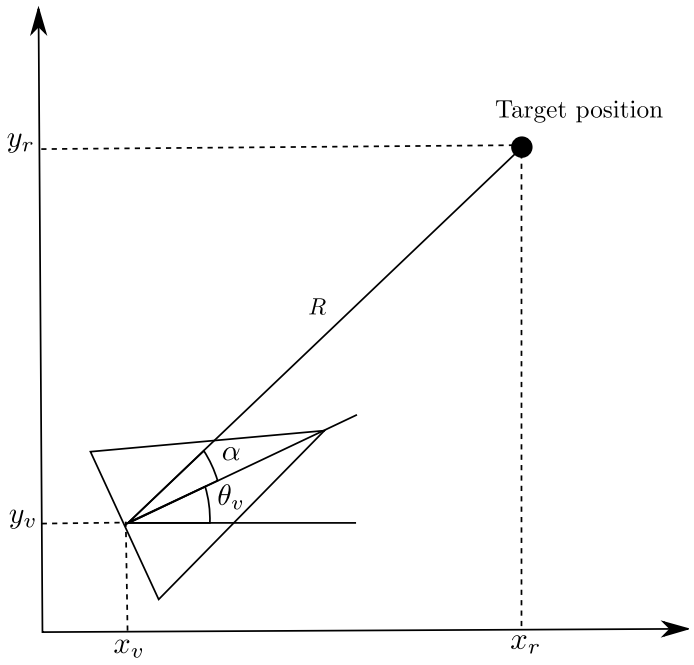


Figure 6.3: Vehicles pose (x_v, y_v, θ_v) and target position (x_r, y_r) represented in polar coordinates (R, α).

negative, then we just set $v = e$, a small positive value. This value e is determined by the dynamics of the vehicle and should be such that the AUV remains robustly controllable.

In order to minimize the distance traveled by the vehicle, we propose here a new feature, to be added to the control strategy, that we called a virtual reference. The idea is to avoid the extra motion of the vehicle when the reference is coming towards its position, as will be defined below. In this case, the vehicle will move as little as possible and wait for the reference to pass by. Mathematically, the virtual reference is computed whenever the projection

$$P^y_{X_v} = -\sin(\theta_v)(x_v - x_r) + \cos(\theta_v)(y_v - y_r)$$

of the position of the vehicle X_v on the y axis of the reference frame, y^r, has a positive value. In this case, the AUV will target a virtual reference located along y^r with some value w greater then $P^y_{X_v}$. When the reference passes by ($P^y_{X_v} < 0$), the vehicle switches and targets once again the real reference. Figure 6.4 shows a case

where the virtual reference is created.

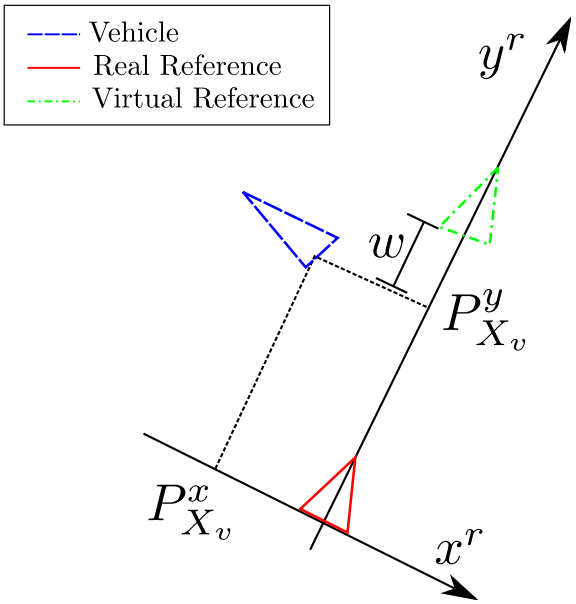


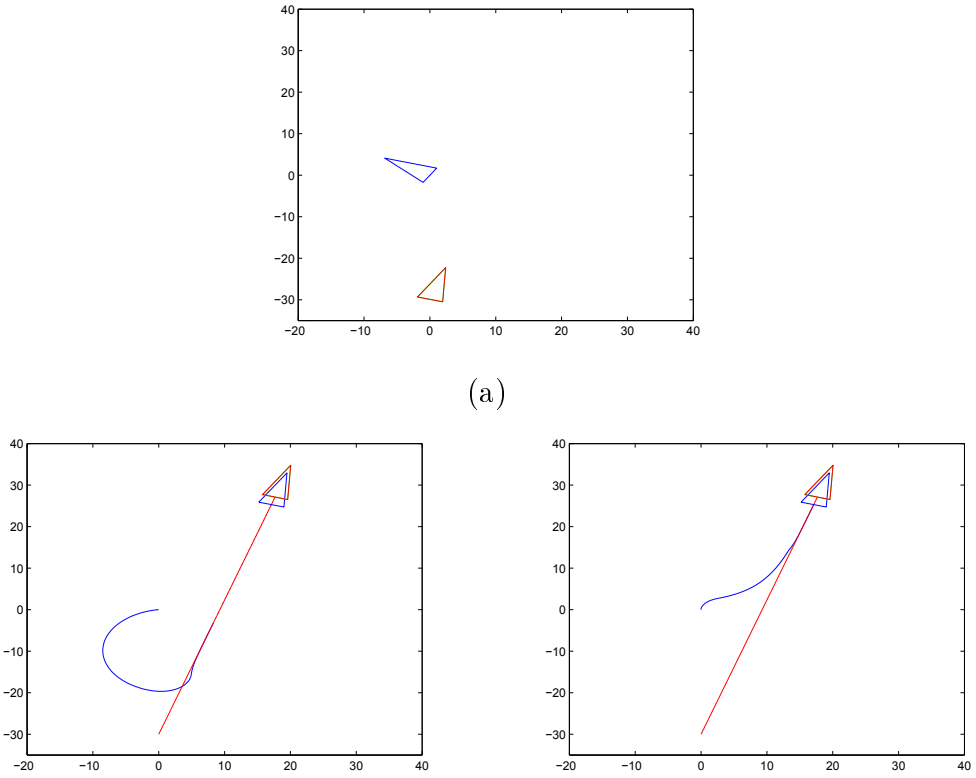
Figure 6.4: When the vehicle (blue) is above the real reference (red) along the y^r axes, that is, the projection of the position of the vehicle X_v on the y axis of the reference frame has a positive value, a virtual reference (green) is created and targeted by the vehicle.

It is also necessary to decrease the linear velocity gain K_1 when the vehicle is targeting the virtual reference. K_1 is decreased proportionally to the inverse of the distance between the vehicle and the virtual reference. In this way, the vehicle will move slower as it approaches the virtual reference, and wait for the real reference to pass by. However, the velocity should be above a minimum value e in order to ensure controllability. In order to avoid the singularity introduced when the vehicle is too close to the reference, the vehicles switches between virtual and real reference only after the real reference has passed a certain distance d from the vehicle ($P^y_{X_v} < -d$). The full control law is shown in Algorithm 6.1. In Figure 6.5, we can see a comparison between the execution of the control algorithm using the virtual reference scheme and that when not using it. Notice that with the virtual reference, the distance traveled by the vehicle is much smaller, thus injecting less noise in the vehicles pose and, at the same time, saving power.

Algorithm 6.1 Control Algorithm with virtual reference

1. Initialize State Vector, Error Covariance Matrix and references for each FAUV
2. While true
   (a) For $i = 1..n$, where n is the number of FAUVs in the system
      i. Compute the reference position $X^i_{ref}(t)$ of the i^{th} FAUV based on the reference of the last step $X^i_{ref}(t-1)$, and the FAUV current information about the heading and velocity of the CAUV
      ii. Compute the projection $P^y_{X_v}$ of the i^{th} FAUV estimated position on they^raxis of the frame fixed on the reference
      iii. If $P^y_{X_v} < -d$
         A. Compute the virtual reference $X^i_{vref}(t)$ and set it as the target for the i^{th} FAUV
         B. Decrease the gain k_1 proportionally to the inverse of the distance between the i^{th} FAUV and $X^i_{vref}(t)$
      iv. Endif
      v. Compute the control signal u^i by using the control laws (Equation 6.5) based on the i^{th} FAUV current position and the reference
      vi. Move the i^{th} FAUVs based on the control signal u^i
   (b) Endfor
   (c) Move the CAUV based on his own internal state and current mission.
   (d) If there is an observation
      i. Update the CCFAUV information about the heading and velocity of the CAUV
      ii. Based on the covariance matrix P_{SEKF} (the one with all the vehicles computed by the CAUV), compute the angle θ_{max} of the CCFAUV relative to the CAUV that maximizes $|L|$
      iii. Compute the target position $X_{ref}(t+1)$ of the CCFAUV based on θ_{max} and some radius$r$ (the desired distance between the FAUVS and the CAUV)
   (e) Endif
   (f) $t = t + 1$
3. Endwhile

During an execution of the algorithm, as the CAUV communicates with the FAUVs, the optimal angle θ_{max} for each FAUV changes with time. If the optimal angle of a FAUV varies too rapidly, then this implies fast changes in the FAUVs position, and more noise is introduced in its pose estimate. Moreover, these fast changes may prevent the vehicle to reach the optimal position. One strategy to prevent this situation, consists in, before reaching its current target, the FAUV communicating once more with the CAUV in order to receive a new target. Therefore, we used a filter to slow down the optimal angle changes. This filter takes the weighted mean of the past n values of θ_{max} ($\theta_{max}(k-1)$, $\theta_{max}(k-2)$, ..., $\theta_{max}(k-n)$) and the current



(a)

(b) (c)

Figure 6.5: Comparison of the control laws: a) initial position of the vehicle (blue) and reference (red), which is moving forward in a straight line b) Pure control law applied c) Control law with the virtual reference

target one $\theta_{max}(k)$.

6.2 Optimal Communication Protocol

The same idea of maximizing the determinant of the prediction variance in the EKF equations can be used by the CEKOLM algorithms so that the CAUV is able to chose which vehicle to communicate with in order to optimize the observations. In this case, we must simply compute $Det(L)$ for each possible communication pair (CAUV,FAUV$_i$) and choose vehicle i that maximizes it. This is depicted in Figure 6.6.

One difficulty that arises with this formulation is that the CAUV does not have

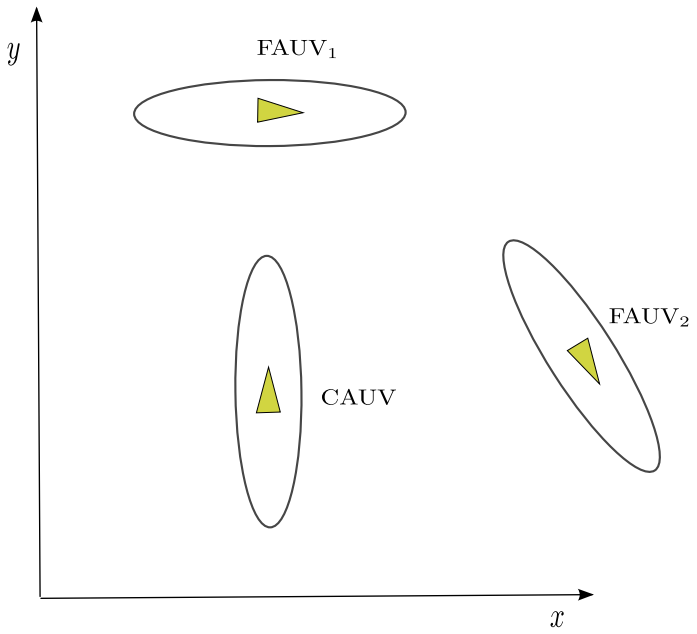


Figure 6.6: Estimated position and uncertainties of three vehicles. The CAUV must choose which FAUV to communicate with in order to optimize the observation. This is done by simulating both communications pairs,(CAUV,FAUV_1) and(CAUV,FAUV_2), and choosing the one that maximizes the determinant of the prediction variance $Det(L) = |L|$.

the current position of all the vehicles involved in the system. It also does not know the current uncertainty of all the vehicles. It does have the position and uncertainty of all vehicles at the last time instant in which each one was observed. So, when computing the best FAUV to communicate with, the CAUV will use the best information available, which consists on the last computed pose and computed uncertainty of each vehicle.

The fact that the CAUV is choosing which vehicle it will observe can cause a problem. It might happen that some of the vehicles will not be observed for a long period, or not observed at all. Imagine a scenario in which the current uncertainties of all the FAUVs stored on the CAUV are all similar (about the same uncertainty volume for each one). Once the CAUV chooses the optimal FAUV to observe, and receives its data, it will first update its uncertainty (it will have increased since the last time it was observed), and then decrease its uncertainty by the update

equations using the observation. If the rate at which the uncertainty decreases is less than the rate it increases, the uncertainty of this particular CCFAUV will keep increasing, and it will probably be chosen by the CAUV to be the next vehicle to be observed. Although the uncertainty of all the other vehicles are increasing, the CAUV is unaware of this fact, since it is not in communication with them.

One way to solve this difficulty is to assign a maximum period T^i_{max} which FAUV_i can be unobserved. Once this period is reached, FAUV_i must be chosen by the CAUV. Another way would be to use the DCEKOLM algorithm to perform the cooperative localization. In this case, since all the vehicles take turns playing the role of the CAUV, everyone of them will surely participate in at least one observation each cycle (a cycle being complete when all the vehicles get to play the role of the CAUV).

6.3 Cooperative Underwater Plume Tracing

A cooperative localization scheme in which the vehicles share a minimum amount of data and are still able to keep a consistent correlation matrix was proposed last Chapter to solve the localization problem for AUVs. This solution gives only accurate relative localization between the vehicles. If global localization is needed, one of the vehicles must surface and get GPS data. However, there are many classes of applications for which this relative localization suffices, as, for example, searching for a minimum for a scalar field. This can be regarded as an abstraction of a wide range of practical applications, among which plume tracing is of utmost importance given the great societal impact of many of its instances, [96], such as, finding underwater - either chemical or oil pollution - sources, investigation of natural phenomena, exploitation of natural resources, etc.

Although the systematic sweeping of a given volume of water in the ocean is a very simple approach, typically it requires huge amount of very expensive resources

over extended period of time and, may even be useless if constraints on time are significant as it is the case of intermittent phenomena. These drawbacks motive the approach addressed in this thesis in which a team composed of multiple AUVS cooperate in order to search for the source of the plume by employing a strategy inspired in the simplex optimization technique. This approach requires the concentration of the plume's substance obtained by appropriate sampling of the water column as well as the localization data of the points in which sampling took place. The idea of using multiple vehicles to search for a minimum in a scalar field is not new, as shown in [111]. The authors show in [52] that a multi-robot approach for plume tracing in a wind-varying and obstructed environment has some advantages. In [56], the authors propose a method for tracing an plume in a indoor environment using multiple robots and an improved ant colony algorithm. In this work we introduce a distributed plume tracing algorithm in which a swarm of robots is able to self-organize into a sensing grid by using local information. The ideas used for our work were first presented by Lobo in [71] and further developed in [72] and [85], where the authors propose a physical realization of a simplex downhill method with multi-vehicle, and present a layered control architecture to control the team of vehicles involved. Here, we extend the ideas by studying how the simplex method behaves when its vertices have an estimated position with associated uncertainties.

In order to keep the key ideas of our approach simple, we will not consider the stochastic aspects inherent to the natural turbulence in the water column, and, instead, we focus in meeting the localization requirements, what, given the characterization of the environment, is the most fundamental challenge.

Suppose we want to search for a minimum in a scalar field $f : \mathbb{R}^n \rightarrow \mathbb{R}$, that is, find the point $x_{min} \in \mathbb{R}^n : f(x_{min}) < f(x), \forall_x \in \mathbb{R}^n$.

The simplex downhill is a classical nonlinear optimization method developed by Nelder and Mead, [64], which has the advantage of not using the derivative of the

function being minimized. Given a n-dimensional function $f(x)$,$f : \mathbb{R}^n \rightarrow \mathbb{R}$ to be minimized, the method moves an initial simplex, i.e., a geometric figure consisting of $n+1$ independent vertices connected by line segments in n dimensions (for example, if $n = 2$ the simplex is a triangle as shown in Figure 6.7) downhill through reflections, which replaces the worst vertex (i.e., the point in which $f(x)$ has its highest value) with a point reflected through the centroid of the remaining N points. Supposing that the vertices are ordered so that $f(x_1) < f(x_2) < ... < f(x_{n+1})$, the reflection step consists in computing first the centroid $x_c = (x_1 + x_2 + ... + x_n)/n$, then the reflected point $x_r = x_c + \alpha(x_c - x_{n+1})$, followed by the replacing of x_{n+1} by it, and, finally, reordering the points. This process is shown in Figure 6.7. The value of α can be adjusted, depending on the value of $f(x_r)$ when compared with the other vertices, so that the reflected point can move along the reflection direction more (called reflection and expansion) or less (called reflection and contraction).

The process is repeated until the value of the reflected point $f(x_r) > f(x_i),\ i = 1, ..., n$. Once this happens, it is possible to either stop the iterative procedure or perform what is called a contraction or simplex reduction, by reducing the size of the simplex, what enables the method to increase the precision of minimum point estimate.

It is possible to employ a team of vehicles to perform a physical realization of the simplex, as presented in [85]. For example, take a team of underwater vehicles searching for the source of a chemical leak, such as a leaking underwater oil pipe. In this case, the vehicles are equipped with a sensor to measure the concentration of oil present on the water. As they get closer to the source, the concentration of oil increases. If the vehicles take measurements at the positions defined by the vertices of a simplex and communicate with each other, it is possible to use the simplex algorithm to guide the vehicles towards their goal. One advantage of this method is that it does not require the vehicles to have an accurate global positioning system,

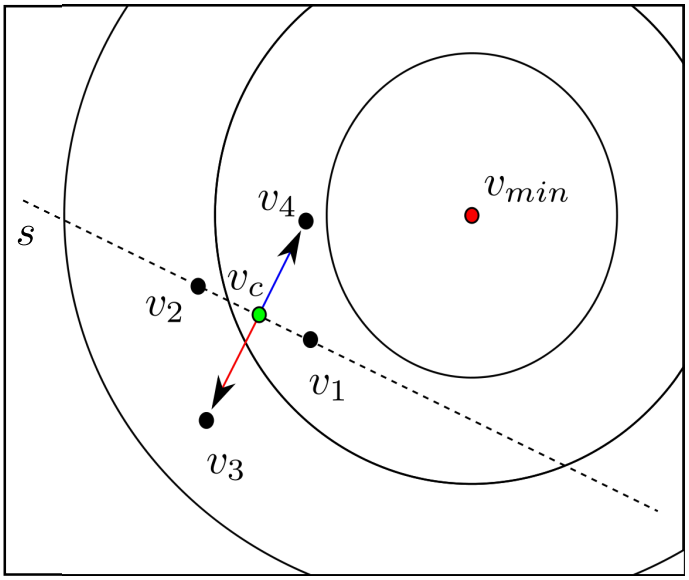


Figure 6.7: Simplex reflection: the vertex with highest function value v_3 is reflected through the centroid v_c of the remaining vertices and replaced by vertex v_4. Here,v_{min} represents the minimum of the function.

but only an accurate relative position.

However, if there is no accurate relative position, the simplex algorithm may fail. If each vehicle uses only their own estimate of its position, there is no way to know which vehicles are actually closer to the minimum. Now, we investigate how accurate this relative positioning must be so that the simplex method will work properly.

As mentioned in Section 5.3, and further explained in Chapter 8, the algorithms presented in this thesis are able to perform relative positioning accurately. We could, for example, use the CEKOLM algorithm, presented in Section 5.3.2, to perform cooperative vehicle localization without any global positioning system to achieve relative positioning with a given uncertainty. With this relative positioning, if the region of uncertainty of the vehicles is small when compared with the size of the simplex the simplex optimization works normally, as will be commented later.

Our simplex is similar to a simplex with fixed size, as in [85], in the sense that

there is no expansion, contraction or reduction of the simplex. There is only reflection. However, since it might be too strict to require the vehicles to reach their target positions (the computed vertices of the simplex) with zero error in finite time, we consider that the vehicle is at the target position when it is close enough to it (inside some ball with radius r centered in the target position). This means that the size of the simplex can actually change. To prevent it from becoming too small or too big, the simplex is reset to a new equilateral one in the current region whenever one of the sides of the simplex is $minside < simpleside_i < maximumside$.

In a fixed sized simplex, the algorithm will stop if the reflected point is equal to the previous worst point. However, since in our formulation the size of the simplex is actually not constant, we test whether the new point is sufficiently close to the previous one or not. Due to noise in the vehicles sensor readings or to environmental disturbances in the function to be minimized (due, say, to turbulence), this stopping criteria can happen in a point that is not the actual minimum. To eliminate this false detection, we require that the stopping criteria happens m times inside a given region. The final minima will be the average of the m minima found.

Up to now, there is no general proof of the convergence of the simplex method. In [46] it is shown that it is heuristic, and proved only for one-dimension and some cases of two dimensions (some family of strictly convex functions). However, the method is known to work very well in practice, [46], and the idea is to move towards the minimum at each algorithm step, producing a rapid initial decrease in function values.

As has been done throughout this thesis, we will consider here the case of finding the minimum in $\mathbb{R}^2$, performing relative localization in $\mathbb{R}^2$, and the results can be extended to $\mathbb{R}^3$. One pertinent question to be raised is how robust with respect to uncertainty in its vertices is the simplex method. In Figure 6.8, we show the scenario we are considering: given the set of estimated vertices positions $\hat{v}_1$, $\hat{v}_2$ and

$\hat{v}_3$, with some associated uncertainties (ellipses representing the region of confidence of the estimated value of the vertices), such that the value of the function on their real positions v_1, v_2 and v_3 are such that $f(v_1) < f(v_2) < f(v_3)$. We want to known how this influence the reflected estimated vertex $\hat{v}_4$ and its uncertainty (red dashed ellipsis), and how this affects the optimization process to find the minima $v_{min} : f(v_{min}) < f(v), \forall_v \in \mathbb{R}^2$.

In this physical realization of the simplex, there will be first an estimated computed vertex $c\hat{v}_4$ based on the previous estimated vertices $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$ (the real computed position of the reflected vertex cv_4 would be given by the true vertices positions v_1, v_2 and v_3). Once vehicle j reaches, at time t, the area close to the estimated computed vertex $c\hat{v}_4$, this vertex will be replaced by the estimated position of the vehicle $\hat{x_j}(t)$ and will now be called $\hat{v}_4$, and its true position v_4 will be the true vehicle position $x_j(t)$.

There are two kinds of uncertainties associated with the simplex algorithm with uncertain vertex.

The first type of uncertainty is produced by the entity that outputs the vertex - in our case the vehicle. Since the vehicle is uncertain about its position, once it reports the position of a vertex, this vertex will contain the same uncertainty of the vehicle.

The second type of uncertainty is the one given by the reflection step of the simplex algorithm. Since the estimated vertices $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$ have some associated uncertainty, the computed reflected vertex cv_4, will also exhibit some uncertainty which is a function of the uncertainty of the the other vertices. Remember that once some vehicle j reaches, at time t, the estimated position $c\hat{v}_4$, this vertex will now be called v_4 and it will now have a uncertainty of the first type, which is the same as the uncertainty of the vehicle at this point, that is, $P_j(t)$.

We want to investigate how these uncertainties affect the simplex method. First

lets show how to compute the second type of uncertainty - the uncertainty in the estimated computed reflected vertex $c\hat{v}_4$. Let us write the 2D simplex algorithm in the form

$$X_{k+1} = AX_k.$$

This can be done by considering $X = [x_1\, y_1\, x_2\, y_2\, x_3\, y_3]^T$, where $v_i = (x_i, y_i)$ represents the i^{th} vertex and they are ordered such that $f(v_1) < f(v_2) < f(v_3)$. We need to find a suitable matrix A that represents the evolution of the vertices in the simplex method. Once this is done, if we consider the simplex with uncertain vertices and represent those uncertainties by a covariance matrix P, then the evolution of the uncertainty will given by, [94],

$$P_{k+1} = AP_kA^T. \tag{6.6}$$

We can decompose the simplex algorithm in two basic steps:

1. Reflection: in 2D this can be written as $v_4 = (v_1 + v_2)/2 + \alpha((v_1 + v_2)/2 - v_3)$ followed by $v_3 = v_4$. If $\alpha = 1$ (which is common practice, [46]), then we can write $v_3 = v_1 + v_2 - v_3$. Since the vertices v_1 and v_2 do not change in the reflection, we can compute matrix A as:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \end{bmatrix} \tag{6.7}$$

2. Ordering of the vertices such that $f(v_1) < f(v_2) < f(v_3)$. This step does not alter the values of X or P, just changes their ordering ($AA^T = I$).

We conclude that by formula 6.6 we can compute the covariance of the new vertex found considering the matrix A in (6.7). This process is depicted in Figure 6.8. The estimated position of the vertex $c\hat{v}_4$ is computed from the vertices $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$, and will have some associated uncertainty represented by a red dashed ellipsis, which is a function of the original uncertainties of $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$. So by simply using the transition matrix A presented and Equation (6.6) we can compute the uncertainty of $c\hat{v}_4$.

Now, we know how to compute the uncertainty of the new vertex estimate $c\hat{v}_4$, but how does this uncertainty affects the simplex method? We are supposed to move in the direction of the minima, following somewhat the gradient direction of the function. This means that the highest value vertex v_3 and the computed reflected vertex cv_4 must be in opposite sides of the line s connecting the other two vertices, as shown in Figure 6.7 (that's why it is called reflection). If the estimated position $c\hat{v}_4$ is above $\hat{s}$ - line depicted in Figure 6.8 connecting the real vertices v_1 and v_2, that is, passing trough some point in the region of confidence of both $\hat{v}_1$ and $\hat{v}_2$ - but the true position of cv_4 is below, the simplex algorithm will think it is moving towards the minima, but actually it is moving away from it, and thus we are compromising its convergence. Even worst than that, it might happen that the value of $f(cv_4) < max(f(v_1), f(v_2))$ and the value of $f(v_4) > max(f(v_1), f(v_2))$. In this case, the simplex algorithm, in the next step, would give the new reflected vertex back at $\hat{v}_3$. Since this is the stopping criteria for detecting a minimum, the algorithm would stop and return the wrong estimated minimum point $\hat{v}_{min}$ (once it happens, this effect will keep repeating itself, so the algorithm will stop even if we wait for this to happen for m times). This means that the uncertainty ellipsis of $c\hat{v}_4$ cannot intersect the line $\hat{s}$, giving any chance that the real vertex cv_4 will be on the wrong side of $\hat{s}$.

Since we do not know the true position of the vehicles, v_1 and v_2, in order to

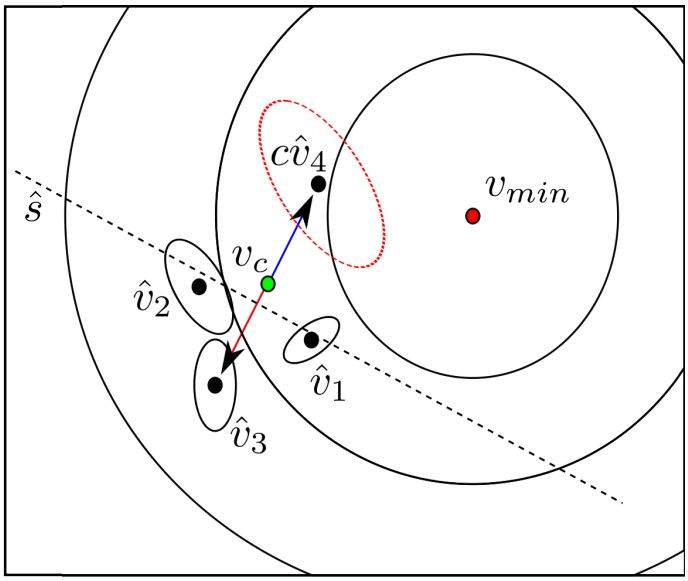


Figure 6.8: Simplex with uncertainties: for the simplex algorithm to work properly it must move in the direction of the minimum v_{min}. The simplex vertex v_3, being the one with the highest value, will be reflected, generating a new computed vertex cv_4. Since the vertices have errors, the estimated position of $c\hat{v}_4$ will also have an error (red ellipsis). If this error is too large, it is possible that the true position of cv_4 is bellow the straight line $\hat{s}$ (connecting v_1 and v_2), while the estimated one is above, causing problems to the simplex algorithm.

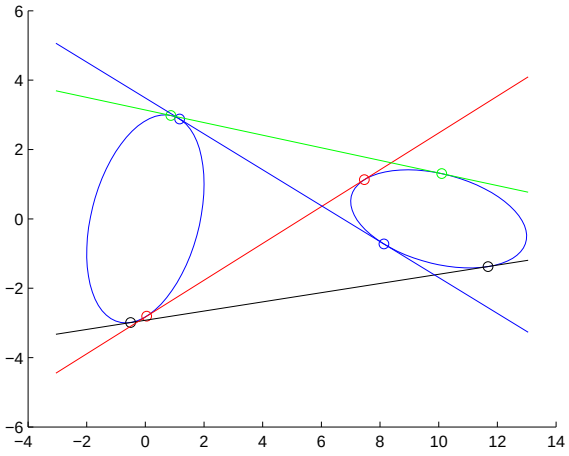


Figure 6.9: Four lines tangent to two non-overlapping ellipses.

compute the line $\hat{s}$, we will consider the worst case scenario. We will compute the four lines $\hat{s}_k$, $k = 1, 2, 3, 4$, tangent to both uncertainty ellipses, as shown in Figure 6.9.

In order to compute the four tangent lines, we must first extract the semi-major and semi-minor axesa_i andb_i and the rotation angle θ_i, of both error ellipses of vertices v_i, $i = 1, 2$, from their respective error covariance matrix

$$P_i = \begin{bmatrix} P_{xx_i} & P_{xy_i} \\ P_{xy_i} & P_{yy_i} \end{bmatrix}, \; i = 1, 2. \tag{6.8}$$

In order to extract θ_i we can compute,

$$\theta_i = -\frac{1}{2} atan(\frac{2P_{xy_i}}{P_{xx_i} - P_{yy_i}}). \tag{6.9}$$

The semi-major and semi-minor axes can be computed by:

$$If \; P_{xx_i} > P_{yy_i} \qquad a_i = \sqrt{max(eigenvalues(P_i))}, \tag{6.10}$$

$$b_i = \sqrt{min(eigenvalues(P_i))}. \tag{6.11}$$

$$If \; P_{xx_i} < P_{yy_i} \qquad b_i = \sqrt{max(eigenvalues(P_i))}, \tag{6.12}$$

$$a_i = \sqrt{max(eigenvalues(P_i))}. \tag{6.13}$$

Consider the canonical equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{6.14}$$

for an ellipse centered at the origin with no rotation. After rotating such an ellipse by θ_i and displacing it to a new center (x_{c_i}, y_{c_i}), we find the general ellipse equation given by

$$\begin{aligned} f_i(x_i, y_i) &= \frac{((x_i - x_{c_i})cos\theta_i - (y_i - y_{c_i})sin\theta_i)^2}{a_i^2} + \\ &\quad \frac{((x_i - x_{c_i})sin\theta_i + (y_i - y_{c_i})cos\theta_i)^2}{b_i^2} - 1. \end{aligned} \tag{6.15}$$

By differentiating $f_i(x_i, y_i)$ relative to x_i we get

$$\frac{dy_i}{dx_i} = \frac{b^2 cos\theta g(x_i, y_i) - a^2 sin\theta h(x_i, y_i)}{a^2 cos\theta h(x_i, y_i) - b^2 sin\theta g(x_i, y_i)}, \tag{6.16}$$

where

$$g(x_i, y_i) = (x_i - x_{c_i})cos\theta_i - (y_i - y_{c_i})sin\theta_i, \tag{6.17}$$

$$h(x_i, y_i) = (x_i - x_{c_i})sin\theta_i + (y_i - y_{c_i})cos\theta_i. \tag{6.18}$$

Now, in order to compute up to four tuples, each one composed of two pairs of points $(x_1, y_1, x_2, y_2)_k$, $k = 1, ..., 4$, that define the four tangents $\hat{s}_k$, we must solve the non-linear system given by Equation (6.19). The first two equations of the system imply that each pair of points belong to their respective ellipse. The third equation says that the slope at those points in their respective ellipse are the same, and the last equation of the system says that the slope is equal to the slope of the line connecting the two pairs of points.

$$\begin{cases} f_1(x_1, y_1) = 0 \\ f_2(x_2, y_2) = 0 \\ \frac{dy_1}{dx_1} = \frac{dy_2}{dx_2} \\ \frac{dy_1}{dx_1} = \frac{y_2 - y_1}{x_2 - x_1} \end{cases} \tag{6.19}$$

The solutions $X_k = (x_1, y_1, x_2, y_2)_j$ to the nonlinear system determine the tangent lines $\hat{s}_k$. If the ellipses do not intersect, there will be four possible solutions. It is possible to use iterative numerical methods to solve this nonlinear system. By giving different values of the initial approximation x^0 (controlling in what quadrant of each ellipses the initial approximation are placed) it is possible to compute all possible solutions.

Once the tangents are found, we can check if the uncertainty ellipse of $c\hat{v}_4$ intersect any one of them. If this is the case, that means the vehicles have reached a maximum uncertainty value, and, thus, it must be reset (relatively to the CAUV, that is). This can be done by telling the FAUV to stay around its current position (loiter with slow velocity) and commanding the CAUV to a sequence of optimal positions for the estimation of the position of the FAUV, as presented in [93]. Other way to guarantee a low position uncertainty is just to perform relative localization only, by considering the position of the CAUV as certain (zero uncertainty). This way, every time an FAUV communicates with it, its uncertainty drops back to small levels.

After finding the minimum location, since the vehicles are only relatively accurately localized, this location will have an error (translation + rotation) relative to a global reference frame. To remove this error, one of the vehicles could stay submerged, next to the minimum location, while two other vehicles surfaces to get GPS. The surface vehicles can then update the position of the one that is submerged (forming a long baseline on the surface). If there is only two vehicles on the system, the one that surfaced can take a range measurement and then move to another position before taking another one. Once the submerged vehicle updates its position, if it is close enough to the minimum, it can simply report back its position and the estimated minimum. Otherwise, it can compute the transformation matrix T that took it from its position before the updates from the vehicle on the surface to his

current updated position, and apply this transformation to the estimated minimum location.

In the past two Chapters we presented several methods to perform cooperative localization for underwater vehicles and to improve their performance. In the next Chapter we will validate those methods by several simulation scenarios and experimental data.

Chapter 7

Simulations, Experimental Results and Discussions

In this Chapter, we present and discuss the results obtained with our cooperative localization approach described in the previous sections. In Section 7.1, we will present some simulation and experimental results and comparisons of the EKF based methods proposed in Section 5.3. Simulation results of the PF based method are shown in Section 7.3. Subsections 7.4.1 and 7.5 present some results of, respectively, the proposed Information Driven Methods and of the Plume Tracing application proposed in Chapter 6.

7.1 EKF Based Methods

To describe the pose of a vehicle relative to a global reference frame in a 3D environment, we need 6 variables: 3 for position, $[x\, y\, z]$, and 3 for the angles, $[\phi\, \theta\, \psi]$, designated, respectively, by Roll, Pitch and Yaw. Due to the low bandwidth communication in the underwater environment, we would like to describe the vehicles pose with an as small set of variables as possible, in order to minimize the amount of data that has to be shared by the AUVs, especially, in the light of the fact that the uncertainty data grows quadratically with the size of the state vector. Depth sensors are very cheap and precise, so the CAUV can use its own measured depth

and the depth received from the FAUV to project the FAUV´s position into a 2D plane, and, thereby, reduce the cooperative localization from a 3D to a 2D problem, [11], concerning the position of the vehicle. Similarly, inclinometers can be used to measure the pitch angle and project the velocity of the vehicle onto the (x, y) plane. Moreover, since the AUV's center of gravity is usually slightly below the center of buoyancy, providing a restoring moment in pitch and roll [24], it is possible that, with the help of the fins, to keep the roll angle always close to zero. Thus, in our simulations, we consider a 2D vehicle model, and, thus, the shared information by the AUVs at time instant k is composed of the vector $[x(k)\, y(k)\, \psi(k)]$, its 3×3 uncertainty matrix $P(k)$, and the depth information $z(k)$.

In all the simulations, we used a 2D model of the AUVs (unicycle model) where the control input $u(k) = [v_x\, v_y\, v_\Theta]^T$, where v_x and v_y are the velocity in the x and y axes and v_Θ is the turn rate of the vehicle at each time step. We will assume that the vehicle is equipped with a set of sensors (for example an Inertial Measurement Unit) which will be the dead reckoning information available. Thus, the Gaussian process noise has 3 components $q(k) = [q_{vx}(k)\, q_{vy}(k)\, q_{tr}(k)]^T$, where q_{vx} and q_{vy} are the noise in the velocity in x and y axes, respectively, and q_{tr} is the noise, in degrees, in the turn rate. The mean is zero and the value of Q in the simulations is given by (7.1). The external sensor, a range detector, has variance $R = [0.5]$. In all the simulations the time step is 0.1 seconds (vehicle models updated with a frequency of 10 Hz).

$$Q = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.35 \end{bmatrix} \tag{7.1}$$

Figure 7.1 shows a simulation of CEKOLM2 with 3 FAUVs, the observation period of 5 seconds (that is, the CAUV communicates with an FAUV every 5 seconds) and no FAUV has GPS update during the simulations. We can see the true

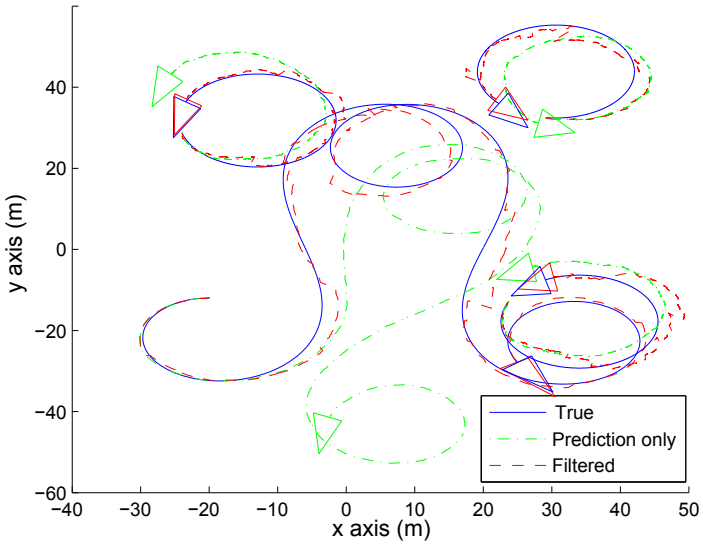


Figure 7.1: 2D simulation of CEKOLM2 during 320 seconds with 3 FAUVs plotted in the(x, y)-plane. The FAUVs describe circles at a speed of $2m/s$, while the CAUV has a longer trajectory at a speed of $1m/s$. The observation period is 5 seconds, and there are no FAUV position updates with GPS.

trajectory described by the AUVs in blue. While the FAUVs describe circles, the CAUV describes a longer trajectory that resembles an octopus. The filtered trajectory estimated by the algorithm is shown in red, and the prediction estimate obtained by using only proprioceptive information (the prediction equations are in 2.17) is in green. Notice that, at the beginning - the CAUV, for example, starts at $[-20 \ -12 \ \pi/2]^T$ -, the three trajectories match, but, as time increases, the prediction only trajectory drifts away from the true trajectory for all AUVs, as expected, especially for the CAUV, which is moving faster in this simulation. On the other hand, the filtered estimate stays close to the true trajectories and drifts away slower.

The results shown from now on in this Section are from simulations in which all the AUVs have the same trajectory (identical control signals, i.e., the trajectories are similar to the trajectory described by the CAUV in Figure 7.1), but the initial position and orientation are random, as shown in Figure 7.2. All the experiments are repeated 400 times in order to get a sufficiently accurate estimate of mean error

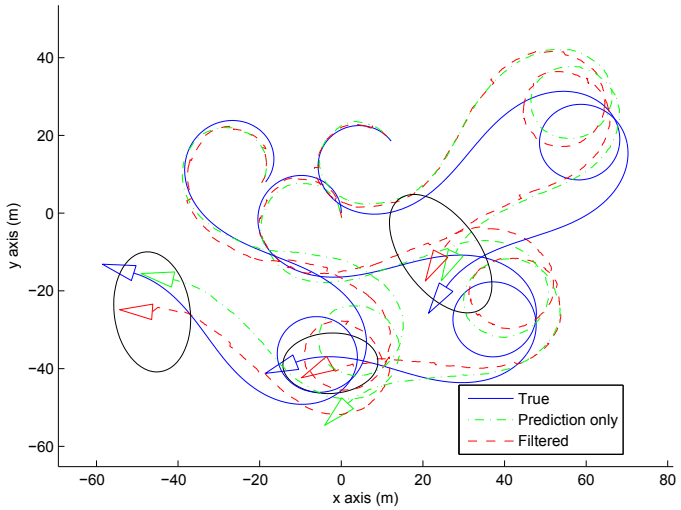


Figure 7.2: 2D simulation of CEKOLM2 during 250 seconds with 3 vehicles, showing the complete trajectory described by all the vehicles.

value. The error in each execution, err, is the mean of the Euclidean error between the true trajectory, in (x, y)-plane, described by the CAUV (CAUV_{true}) and the filtered one (CAUV_{filt}) for each time step of the algorithm, as show in Equation (7.2).

$$err = \frac{1}{nsteps} \sum_{k=0}^{nsteps} \parallel CAUV_{true}(k) - CAUV_{filt}(k) \parallel \tag{7.2}$$

Figure 7.3 shows the mean CAUV error of the simulations of the four proposed EKF algorithms for several observation periods (how often the CAUV communicates with the FAUVs) for the case in which the number of FAUVs is constant and equals to 3 (total of 4 vehicles: 1 CAUV + 3 FAUVs), and there is no GPS update for the FAUVS. As expected, as the observation period increases, so does the mean error for all algorithms. In Figure 7.4(a), we can see the mean error for the FAUVs. You can notice the the FAUV error is higher than the CAUV's. This happens because, if there are n FAUVs in the system, the mean observation period for a FAUV is n times longer than that for the CAUV. Since, in this simulation, there are 3 FAUV,

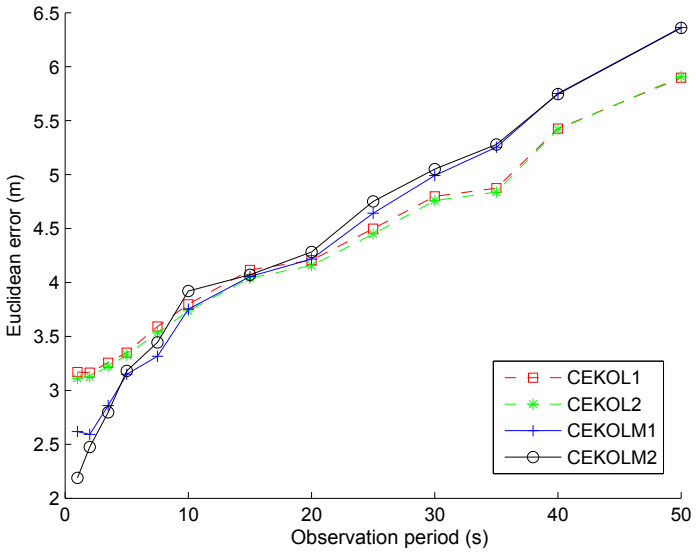
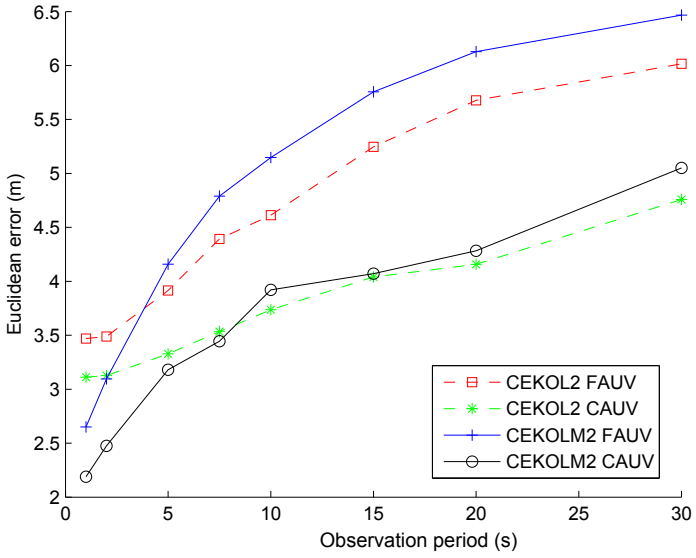


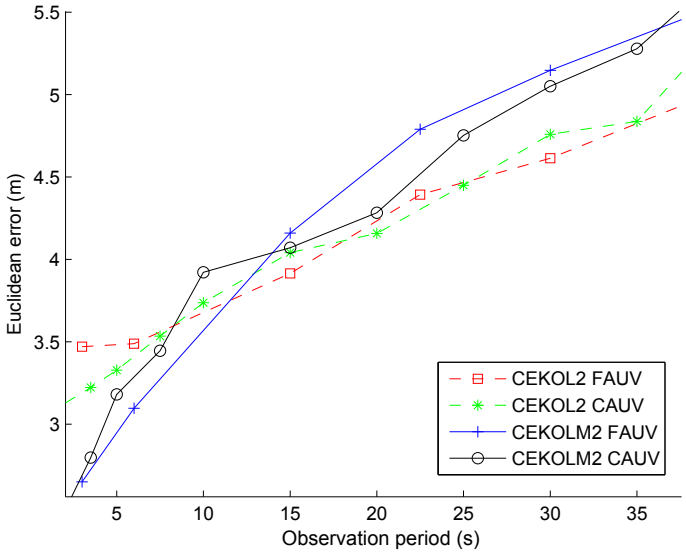
Figure 7.3: Mean Euclidean CAUV error varying the observation period. Total of 4 AUVs (1 CAUV + 3 FAUVs) without FAUV position GPS update.

the mean observation period for a FAUV is 3 times longer than that for the CAUV. So, in a simulation with observation period t_o, the mean error in the pose of the FAUV is approximately equal to the mean error of the CAUV in a simulation with observation period $3t_o$. This was confirmed in 7.4(b), where the scale of the time axis (observation period) of the FAUV error graphic was multiplied by 3 and plotted against the CAUV error. For example, we can see in Figure 7.4(a) that the mean error of a FAUV using CEKOL2 with observation period of $5s$ is almost $4m$, which is approximately the same value of the mean error of the CAUV using CEKOLM2 with observation period of $3 \times 5 = 15s$.

As shown in Table 7.1, if the observation period is $5s$, the mean error for CEKOL1 is $3.4m$ and for CEKOLM2 is $3.1m$. For these parameter values, the error of the algorithm proposed by Alexander Bahr *et al* [11], which is the method found in the literature that fits our problem and have similar bandwidth requirements, called Cooperative Navigation Algorithm (CONA), that adjusts the position of the CAUV only, yields a mean error of 11.6 m, even higher than the prediction only mean error



(a)



(b)

Figure 7.4: (a) Mean Euclidean CAUV and FAUV error variation with the observation period. (b) Same as (a) but the FAUV's graphic scaled by a factor of 3 on the observation period axis.

Algorithm	Mean Euclidean Error (m)	Variance
Prediction	8.2	20.5
CONA	11.6	21
CEKOL1	3.4	2.5
CEKOL2	3.3	2.4
CEKOLM1	3.1	2
CEKOLM2	3.1	1.9

Table 7.1: Mean Euclidean error and variance of the algorithms for CAUV during 320 seconds, with an observation period of 5 seconds and no GPS update. The total distance traveled by the AUVs is 320 meters.

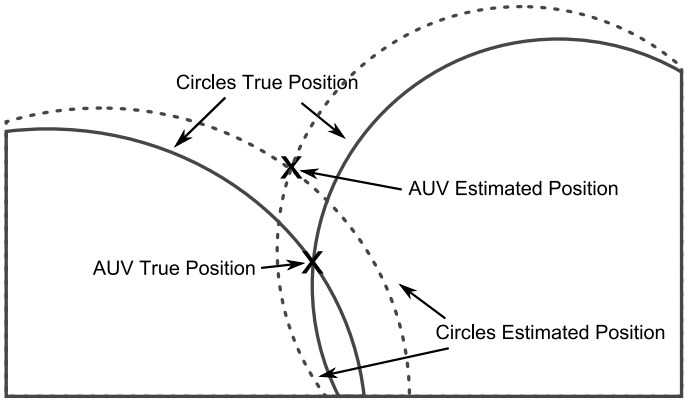


Figure 7.5: Impact of errors in the execution of the CONA algorithm: if the circles' radius are sufficiently large, even a small error in the estimated position of the circle can cause a big error in the AUV's position estimate.

(8.1 m). This happens because CONA uses the intersection of circles to estimate the CAUV's position. If the system noise (process and sensor) is high, the position of the calculated circles exhibit some error, and, even if this error is small, it can cause a high error on the CAUV's position estimate, which depends on the position of the involved FAUVs, as discussed in [9], and shown in Figure 7.5. If CONA is modified to estimate not only the CAUV's position but also the FAUV´s positions, the mean error becomes even larger (22.3 m).

One can see in Figure 7.3 that the algorithms of the same group - i.e., the CEKOL1 and CEKOL2 based on localization, and the CEKOLM1 and CEKOLM2

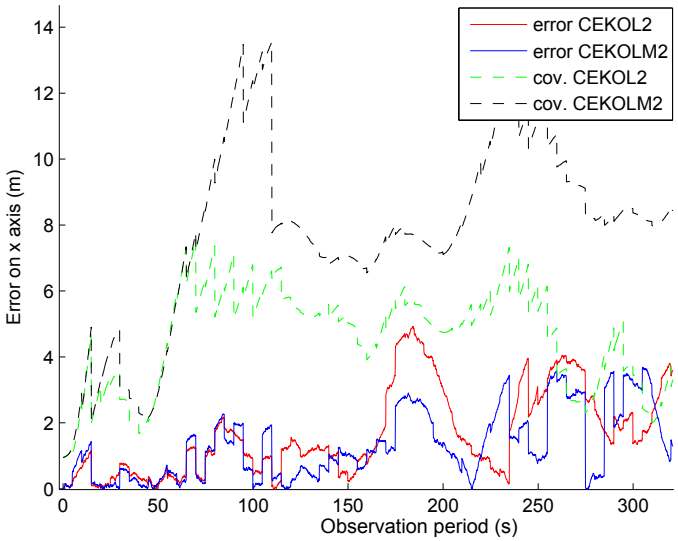
based on SLAM - present similar behaviors and yield similar results as the observation period varies. When the observation period is less than 10 seconds, CEKOLM1 and CEKOLM2 are better, but when the observation period becomes greater than $25s$, CEKOL1 and CEKOL2 yield better results. This happens because, when observation period is small, the repetitive use of the same information, as discussed in Section 5.3.1, leads to overconfidence and, thus, to worse results in CEKOL algorithms. You can also see that even when the observation period is as high as $50s$, the results are still better (around 27% for CEKOL algorithms and around 21% for CEKOLM ones) than the mean error of the prediction method. On the other hand, as the observation period increases, the algorithms based on localization do not loose performance as fast as the ones based on SLAM. This is due to the no correlation assumption of this class of algorithms. Since in between observations, the correlation between vehicles tend to get smaller (when compared with their uncertainties) - that is, the normalized correlations decrease -, it is not too harmful to neglect those correlations. To better understand this phenomenon, we show, in Figure 7.6, the absolute error on the x axis for the CAUV in an execution of CEKOL2 and CEKOLM2, and the region of confidence ($3\sigma_{xx}$) computed by using the covariance matrix $P(t)$ for these algorithms. The CEKOL1 and CEKOLM1 exhibit similar results to those of CEKOL2 and CEKOLM2, respectively. There is no GPS update and the observation period is $5s$ in (a) and $25s$ in (b). For small observation periods, say, $5s$, CEKOL2 underestimates the covariance of the CAUV, although with higher observation periods, such as $25s$, the covariance becomes more accurate. This happens because, in CEKOL2, we falsely assumed that there was no correlation between robots pose as they observe each other during the simulation. This assumption makes a repetitive use of the same evidence, causing the robots to decrease their pose uncertainty more than is warranted by data. As discussed in [33],

one way to reduce this effect is to ignore repetitive observations in short time intervals. This is precisely what happens when the observation period becomes large. In this way, there is no sequence of observations in a short time interval, and thus the repetition of observations decreases, leading to a more accurate computation of the robot's pose uncertainty, as shown in Figure 7.6.

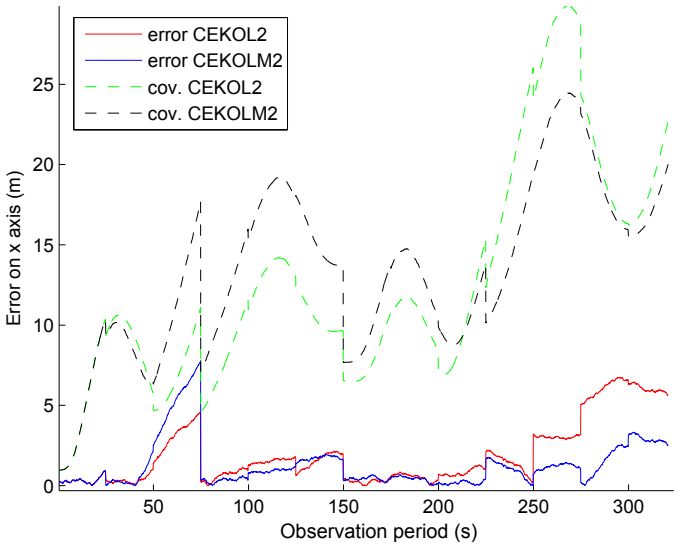
Figure 7.7 shows the variation of the mean Euclidean error of the algorithms with the period with which a randomly selected FAUV updates its position estimate by GPS and for a fixed observation period of $5s$. When the update period is less than $50s$, all the algorithms have similar results. However, as the update period increases, CEKOLM1 and CEKOLM2 show better results when compared with the others. This is due to the fact that, when the update period is small, the estimation problem is closer to a simple localization problem, and, since the position of all FAUVs are well known, the mean error is small. In this case, it is clear that the SLAM based approach of CEKOLM1 and CEKOLM2 brings no advantage.

Figure 7.8 shows the variation of the mean error of the algorithms with the number of FAUVs in the system. If the number of FAUVs is too small (less than 3) the algorithms exhibit higher errors, as we would expect from a conventional localization or SLAM method in an environment with fewer features. As the number of FAUVs increases, the error decreases, until there are about 4 FAUVS. After this point, as the number of FAUVs increases, the results by CEKOLM1 and CEKOLM2 get slightly worse. This happens because, with more FAUVs, the communication frequency with each FAUV decreases and the FAUV mean error gets bigger, as shown in Figure 7.4.

On the other hand, the CEKOL1 and CEKOL2 algorithms start with larger errors then the CEKOLM1 and CEKOLM2, and, since they are less sensitive with respect to the observation frequency, the error gets smaller as the number of FAUVs increases until it stabilized at about 8 FAUVS. This can be seen in Figure 7.3.



(a)



(b)

Figure 7.6: Error and region of confidence ($3\sigma_{xx}$) on$x$ axis for CAUV in an execution of CEKOL2 and CEKOLM2. There is no GPS update and the observation period is (a) 5 seconds, (b) 25 seconds.

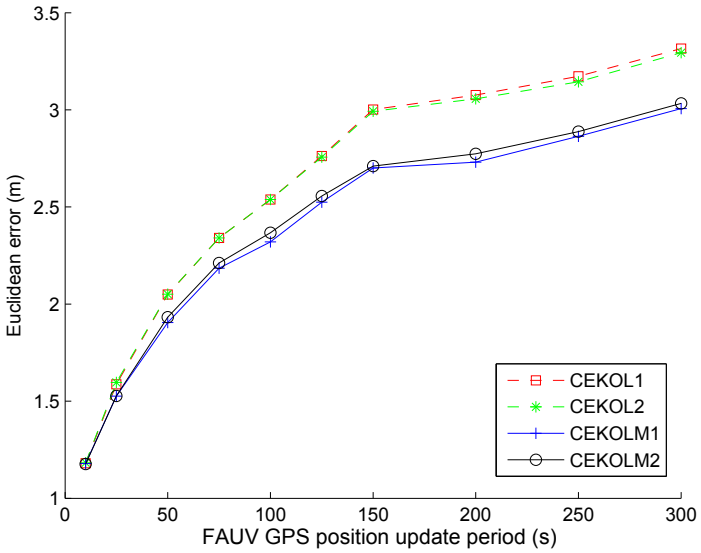


Figure 7.7: Mean Euclidean CAUV error variation with the period with which a randomly selected FAUV updates its position estimate by GPS. The observation period is $5s$, and there are 3 FAUVs.

Although the CAUV mean error stays constant as the number of FAUVs gets larger, the mean FAUV error will increase. This happens because, as the number of FAUVs increases, so does the observation period for the FAUVs, as already discussed in the context of Figure 7.4.

Table 7.2 shows the mean Euclidean error (as a percentage of distance traveled) and the variance of the CAUV, for 400 simulations, describing a straight line trajectory, for the proposed algorithms and without any filtering. In this simulation, the observation period is $5s$, there are 3 FAUVs and none of the FAUV's pose uses GPS update. Notice that both the mean error and the variance are greatly decreased by the algorithms.

By using a 5%level of significance statistical test, we can accept the hypothesis that, for this example, CEKOL1 and CEKOL2 exhibit the same mean error, and that CEKOLM1 performs as good as CEKOLM2. Moreover, we can conclude that CEKOLM1 and CEKOLM2 yield better results than CEKOL1 and CEKOL2.

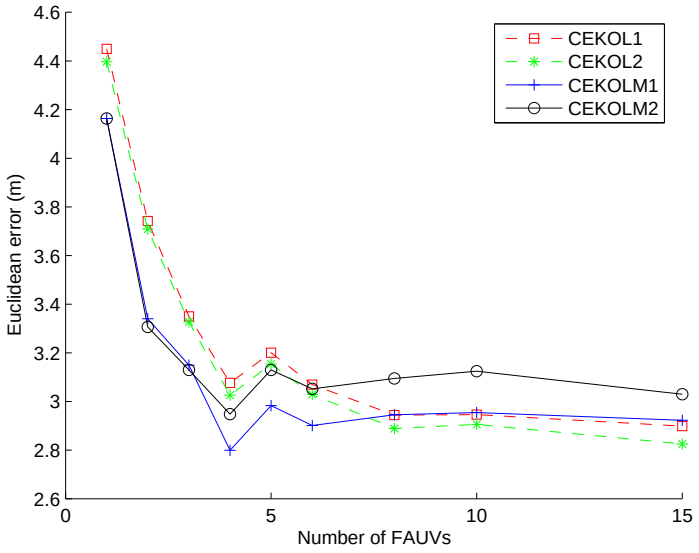


Figure 7.8: Variation of the mean Euclidean CAUV error with the number of FAUVs.

Algorithm	Mean Euclidean Error	Variance
Prediction	11%	20
CEKOL1	3%	3.6
CEKOL2	2.7%	3.2
CEKOLM1	1.8%	1.8
CEKOLM2	1.6%	1.2

Table 7.2: Mean Euclidean error as a percentage of the distance traveled and variance for a CAUV performing a straight line trajectory.

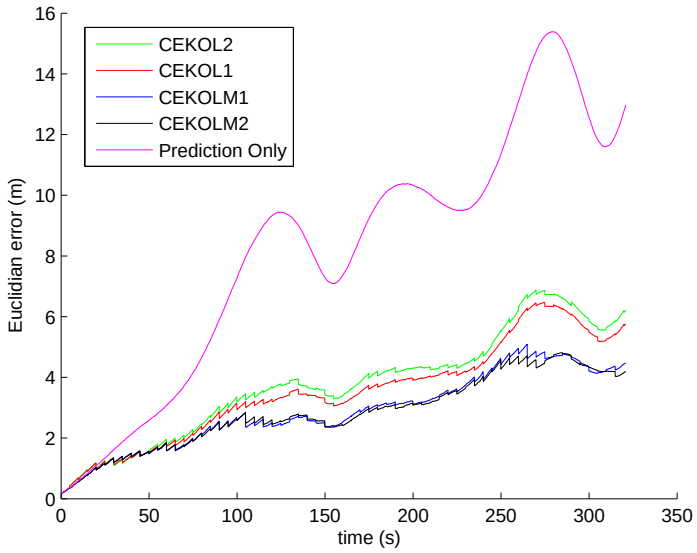


Figure 7.9: CAUV mean Euclidean error as time increases with no FAUV GPS update

In Figure 7.9, we show the mean Euclidean error of the CAUV, describing a trajectory like the one shown in Figure 7.1, as a function of time. Notice that with no GPS update, the error is not bounded, and grows as the CAUV moves. But even if the error is not bounded, one can see that the proposed algorithms greatly reduce the mean error compared with the prediction only approach.

Now, we consider the case in which each one of the AUVs already has some sort of exteroceptive sensor, (or, equivalently, a set of properly fused sensors). In this way, each AUV could use an EKF to improve its own position estimate. If the exteroceptive sensor in each AUV provides enough information so the system is observable and has a small error variance, then the use of another sensor/filtering technique does not bring much advantage, and the proposed methods are not very useful. However, if the exteroceptive sensor does not give enough information, then, in spite of the potentially great reduction in the estimate error, the system will not be observable, and the proposed methods can be applied to further reduce the error.

In the next simulations, we consider that all the involved 4 AUVs are equipped

with a compass. In this way, they are able to measure their heading. The measurement error variance of the compass, in degrees, is $\sigma^2 = 10$. The error variance of the range sensor is the same as in the past simulations, 0.5. In this way, the error covariance matrix of the system is given by $R = \begin{bmatrix} 0.5 & 0 \\ 0 & 10 \end{bmatrix}$. The process noise Q was increased and is now given by Equation (7.3).

$$Q = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{7.3}$$

In Figure 7.10, we see the mean Euclidean error in CAUV position, and its variance, for 20 simulation runs. We show that the results using a KF with the compass information only and, for CEKOLM2, using the compass and range observations between AUVs. In these simulation runs there is no GPS update and the observation period is $5s$. We can see that CEKOLM2 reduces the covariance and the error. In this case, the total error was reduced by 32%. The algorithms proposed here reduced the "compass only" mean Euclidean error by 30% to 40%.

In order to validate the ability of the CEKOLM algorithms to handle communication failures, we considered failures in each one of the three steps of the communication process mentioned in Section 5.3.2.4 in a series of CEKOLM2 simulation runs. Since the amount of data transmitted in each step is different, we assigned different probabilities of communication failure for each one of them: 10% chance of failure was assigned to step 1, a 50% failure for step 2, and 25% for step 3. The simulations showed that the algorithm could perform properly and maintain its internal consistency, in spite of the communication failures.

By running a sequence of simulations of the algorithm in two scenarios: (1) perfect communications, and (2) simulated communication failures, we could observe that the results were similar (mean Euclidean error was $4.58m$ and $4.50m$, respectively). Notice that, in order to be able to compare the mean error of the

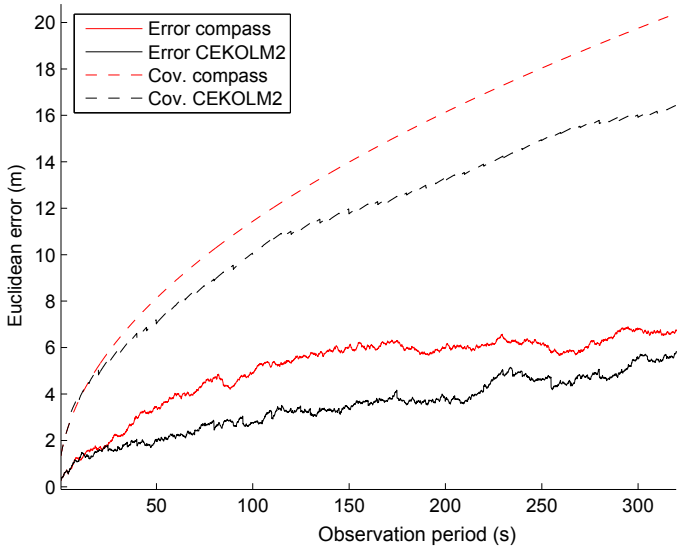


Figure 7.10: The CAUV mean Euclidean error as time increases with no FAUV GPS update and the region of confidence ($3\sigma_x$) computed by using the covariance matrix$P(t)$ with CEKOLM2

two scenarios, we have to adjust the observation frequency of scenario 1 so that it represents the mean success rate of communications in scenario 2. For example, with the probabilities of communication failure in the 3 steps presented above, only 33.75% of the communications are successfully finished. This means that if the CAUV tries an observation every second, the actual mean period of successfully complete observations is $2.96s$.

7.2 Experimental Results

Experimental data from the AUV SEACON, which was developed in FEUP - Faculdade de Engenharia da Universidade do Porto by the Laboratory for Underwater Systems and Technologies - and depicted in Figure 7.11), was used to validate the CEKOLM2 estimation. The AUV is equipped with the following sensors: a motor rpm counter with a sampling frequency of 1Hz, an Inertial Motion Unit (IMU) at 50 Hz, a compass at 50 Hz, a pressure sensor at 15 Hz, and a GPS at 1 Hz.

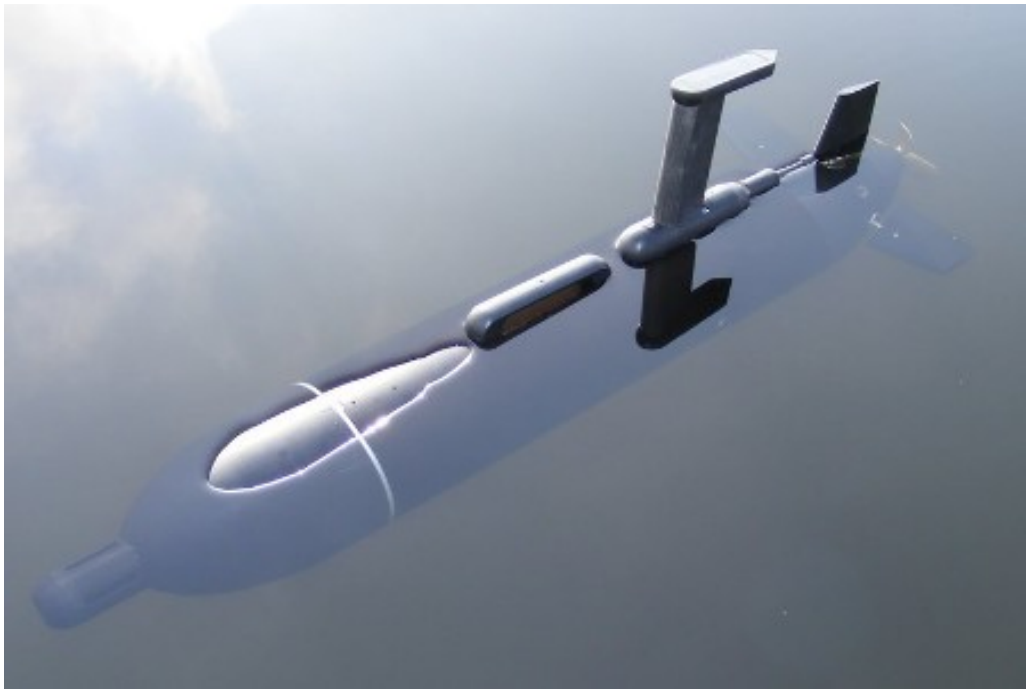


Figure 7.11: The SEACON AUV developed in FEUP - Faculdade de Engenharia da Universidade do Porto by the Laboratory for Underwater Systems and Technologies.

The vehicle executed 3 missions of about 8 minutes each at the water surface and the sensor data collected for each one of the missions. The only reason for the missions to be run at the surface is to enable the use GPS as a ground truth for the experiments. One of the trajectories described by the vehicle is depicted in Figure 7.13. Then, this data was loaded on the simulator, and from the 3 different missions executed at different times we simulated 3 vehicles operating simultaneously. We also simulated the communications and range measurements between the vehicles. The range measurements were simulated with a Gaussian error with variance $R = [1]$, and the observation period was $10s$. In the simulations we considered only the 2D position of the vehicle (latitude, longitude and heading).

First, we simulated the situation in which the vehicles are not equipped with a compass by ignoring the information of this sensor. In this case, the filterless (i.e., prediction only) trajectory, with data only from the IMU can diverge from the true one very quickly. In this scenario, the range measurements can greatly reduce the

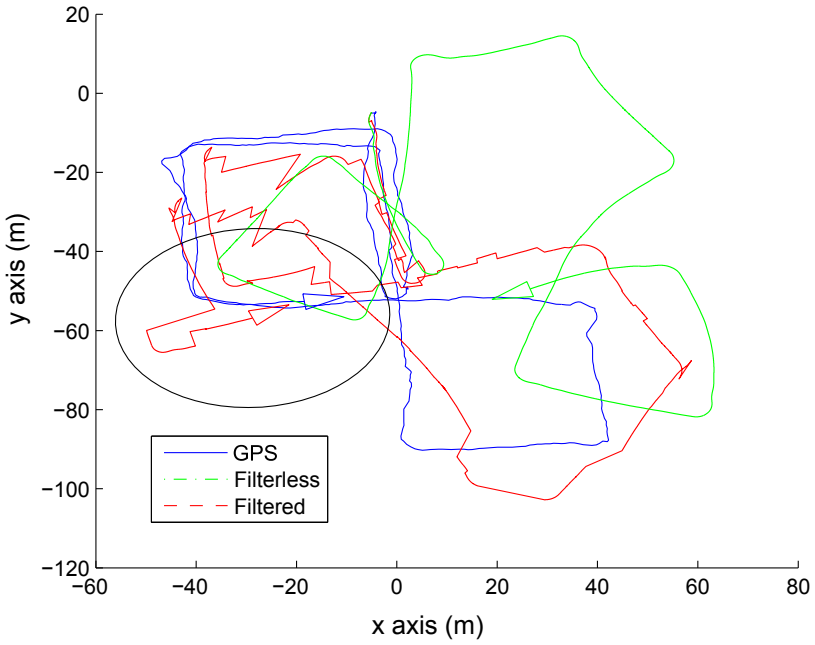


Figure 7.12: Results of the CEKOLM2 algorithm with real data. Each vehicle traveled about $360m$ during $450s$. The simulation considered 3 vehicles working simultaneously. The simulated observation period was $10s$. The ellipse shows the uncertainty region of the AUV position estimate at the end of the experiment. We can see that, without the compass, the filterless trajectory (in green) diverges very fast.

error of the estimated state, as can be seen in Figure 7.12. With an observation period of 10 seconds, the filterless mean euclidean error of the CAUV was $49.22m$, while the filtered trajectory had a mean error of $15.91m$, an error reduction of 67%.

If we use all the sensor data available - i.e., compass and IMU -, the mean Euclidean error obtained by the filterless trajectory (of the CAUV), when compared with the GPS, was $4.89m$, while the error of the filtered trajectory using CEKOLM2 was $2.66m$ (experiment with observation period of $10s$). The computed trajectories described by the CAUV are shown in Figure 7.13. Although the performance improvement is not as high as in the case in which there is no compass, the algorithm was still able to reduce the error by 45%.

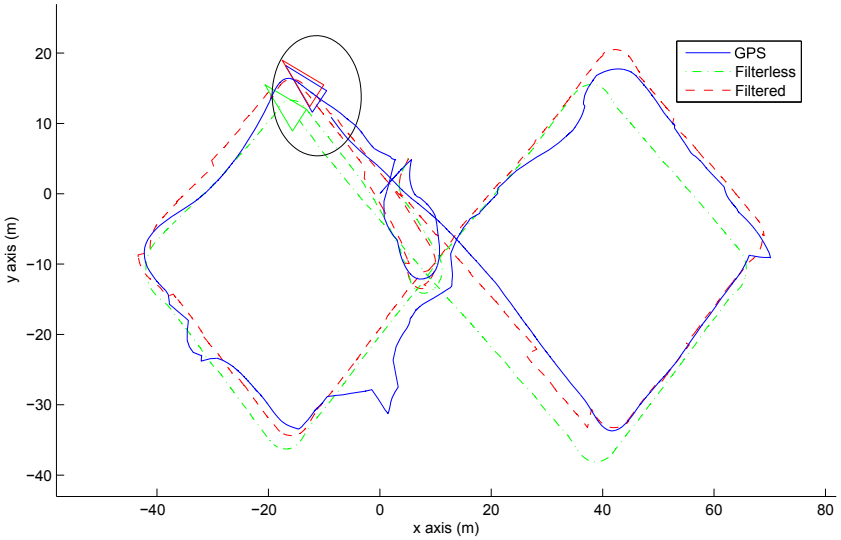


Figure 7.13: Results of CEKOLM2 with real data. Each vehicle traveled about $360m$ during $450s$. The simulation considered 3 vehicles working simultaneously. The simulated observation period was $10s$. The ellipse shows the uncertainty region of the estimated vehicle position at the end of the experiment.

7.3 PF Based Methods

In this Section, we will present some results of the PF Cooperative Localization method, CPF, presented in Section 5.4. The algorithm was tested for several configurations (number of particles and ratio between number of particles and degeneracy limit as shown in Equation (2.28)).

In each simulation run, we used for each vehicle the same 2D model and noises used in the EKF simulations, i. e., the range detector has a variance $R = [0.5]$ and the process noise is given by

$$Q = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.35 \end{bmatrix}. \tag{7.4}$$

The trajectory described by the vehicles are also similar to the trajectories described on the simulations of the EKF method. In Figure 7.14, we can see the trajectory described by the CAUV in a simulation run.

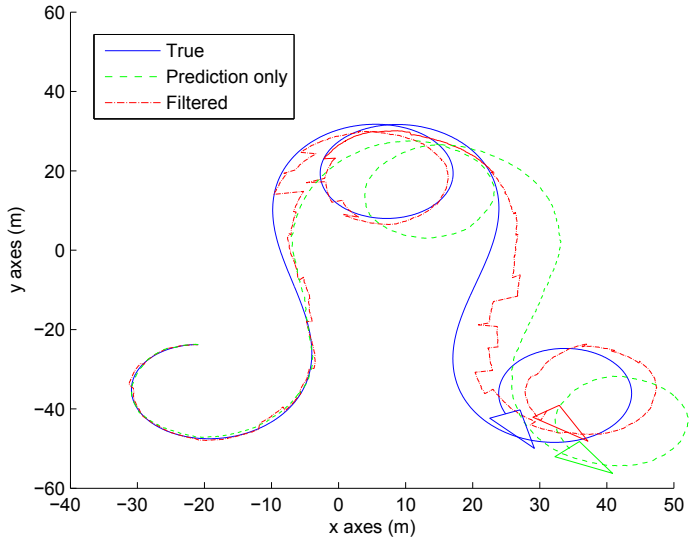


Figure 7.14: 2D simulation of CPF using 50 particles for each vehicle, during 320s, showing the trajectory described by the CAUV. There are 3 FAUVs and they describe the same trajectory of the CAUV with random initial poses. The observation period is 5s, and there is no FAUV position update with GPS data.

In Figure 7.15 we show the mean Euclidean error of the algorithm for 400 simulation runs for several different number of particles. We can see that as the number of particle increases, the mean error of the algorithm decreases, with lower decrease rates as the number of particles get higher. However, as the number of particles increases, so does the complexity of CPF. For about 100 particles, there is a small decrease in the error, and the algorithm is already non practical for real-time applications as the 400 simulation of 320s each took several days on an Intel Core2 Duo 2GHz processor.

The best configuration achieved by CPF with an observation period of 5s and with 3 vehicles was with 100 particles and 2.5 as the ratio between number of particles and degeneracy limit. The mean error of CPF with this configuration after 400 executions was 4.0543m and the variance of the error was 3.1019. We can see, clearly, from the comparison of the results in Table 7.1, that the EKF based methods performed better.

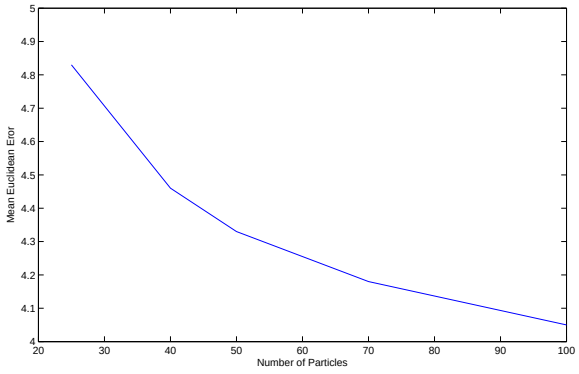


Figure 7.15: The mean Euclidean error of CPF method decreases as the number of particles increases.

One of the factors for the poor performance of the CPF is the lossy compression of the pdf made in the vehicle sending in this information: as in the Gaussian distribution, only two parameters, mean and variance, are considered. This is done because it is impossible for the vehicles to share all the particles and weights that represent their pdfs. This Gaussian approximation of the pdf, by the vehicle sending information discards the key advantage - and the raison d'être - of the PF methods: representing non Gaussian distributions by a set of particles. The particle representation is used only to describe the pdf of the pose of the vehicle receiving the information. Also, notice that, as in the CEKOL algorithms, the CPF method does not create a cross-correlation matrix, which can lead to the underestimation of the uncertainty.

7.4 Information Driven Methods

In this section, we present some simulation results regarding the information driven methods proposed in Sections 6.1 and 6.2.

7.4.1 Results of Optimal Spatial Distribution

Since the best position of the FAUV relative to the CAUV depends only on the angle formed between the vehicle, the origin (feature position) and the x axis, as shown in Equation (6.4), its computation requires the desired distance R_d, that each FAUV is from the CAUV be chosen. We notice also that, in a real application, the CAUV should check the target position assigned to each FAUV for the detection of a collision situation. However, since it has only an estimate of the vehicle's pose, each vehicle must also have some collision avoidance system.

In Figure 7.16 we show a simulation from the CEKOLM2 algorithm where the FAUVs are commanded to the best possible position to maximize the information from the observations. In this simulation, the CAUV is commanded to take a curvy trajectory (blue trajectory on Figure 7.16 (a)). The observation period is $2s$ and the desired distance between CAUV and FAUVs is $20m$. The total distance traveled by the CAUV is $320m$.

Table 7.3 shows the mean error and variance for 400 executions with this curved trajectory described by the CAUV for 3 cases: 1) the algorithm computes the optimal angle θ_{max} for each FAUV and commands them to the optimal position; 2) the algorithm chooses a random angle, at the beginning of the simulation, for each FAUV, and commands them to keep this angle through out the simulation; 3) the algorithm computes the worst possible angle $\theta_{min} = (\theta_{max} + 90°)$ for each FAUV and, then, commands them to these worst case positions. Notice that, when compared to the results from CEKOLM2 algorithms, for which all the AUVs described trajectories similar to the CAUV, shown in Table 7.1, the results in table 7.3 show higher mean errors and higher uncertainties. This happens because, now, the FAUVs have to follow the CAUV with intermittent observations, while keeping a certain formation. This causes them to travel greater distances and with much more irregular trajectories, which results in a higher accumulated process error. Even though, the

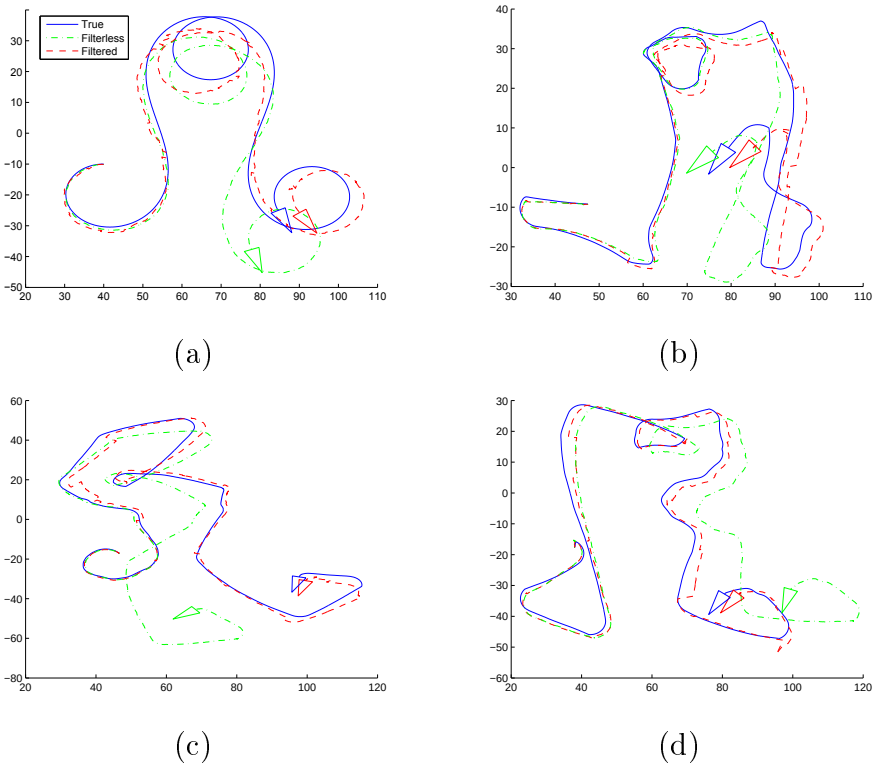


Figure 7.16: Simulations results for CAUV curved trajectory: a) CAUV b), c) and d) FAUVs. observation period of $2s$ and radius of $20m$.

Algorithm	Mean error	Variance
CEKOLM2 with optimal angle	4.0	5.8
CEKOLM2 with random angle	4.0	3.9
CEKOLM2 with worst possible angle	3.9	3.9
Prediction	8.2	20.5

Table 7.3: Comparison between CEKOLM2, in a curvy trajectory, commanding the vehicles to the optimal position, to a random position (based on a random angle), to the opposite position from the optimal($\theta_{max} + 90°$) and for prediction only. The observation period is $5s$. For this kind of trajectory, all the algorithms reduce by approximately 50% the error of the prediction only, but they all behave similarly.

error is reduced by approximately 50% when compared to the prediction only case. Notice however, that the 3 cases behave almost in the same way. This happens because the FAUVs are prevented from maintaining an optimal position relative to the CAUV from the sampling rate which is very low for such an highly variable (curved) trajectory.

In order to fully use the advantages brought by the optimization algorithm, the trajectory described by the CAUV must consist of line segments. Figures 7.17 and 7.18 show two simulations with this kind of trajectory. In Figure 7.17 the observation period is of $2s$, and you can observe that the trajectories described by the FAUVs are closer to the trajectory of the CAUV. In Figure 7.18 the observation period is $5s$, so the trajectories of the FAUVs are not so close of the CAUV, but are still very similar. In both simulations, the total distance traveled by the CAUV is $320m$.

In Table 7.4 we show the mean error and variance for 400 executions, with observation period of $5s$, where the CAUV describes a squared trajectory similar to the one shown in Figures 7.17 and 7.18. As in Table 7.3, we show the results for the same 3 cases: optimal, random and worst case. We can see now that the optimal algorithm performs a lot better and is capable reducing the Euclidean error by 20% while greatly reducing the variance.

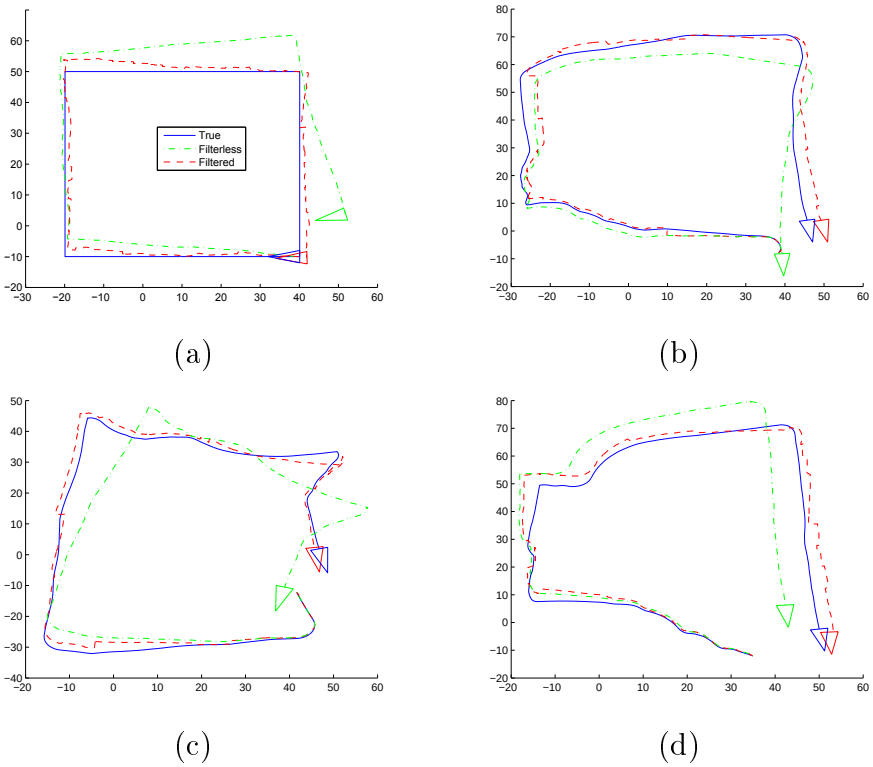


Figure 7.17: Results for square trajectory a) CAUV b), c) and d) FAUVs. observation period of $2s$ and radius of $20m$.

Algorithm	Mean error	Variance
CEKOLM2 with optimal angle	3.8	3.4
CEKOLM2 with random angle	4.8	9.4
CEKOLM2 with worst possible angle	4.8	9.3
Prediction	8.2	17.2

Table 7.4: Comparison between CEKOLM2, in a square trajectory, commanding the vehicles to the optimal position, to a random position (based on a random angle), to the opposite position from the optimal($\theta_{max} + 90°$) and for prediction only. The observation period is $5s$. For this kind of trajectory, the algorithm with optimal angle performs better than the others.

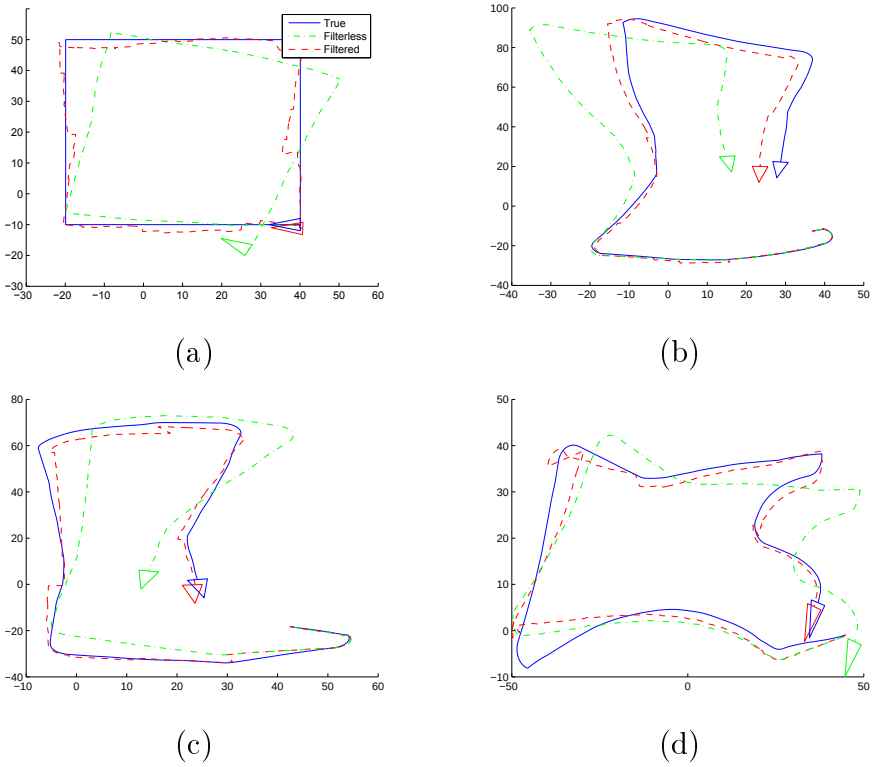


Figure 7.18: Results for square trajectory a) CAUV b), c) and d) FAUVs. observation period of $5s$ and radius of $20m$.

7.4.2 Results of Optimal Communications Protocol using DCEKOLM

The simulation results shown in this Section, in order to test the optimal communications protocol method proposed in Section 6.2, were performed in a group of vehicles using the DCEKOLM algorithm, the decentralized version of CEKOLM, proposed in Section 5.3.2.5. As discussed in 6.2, in this formulation, the vehicles take turns playing the role of the CAUV, and thus the observation of each vehicle is guaranteed in every cycle.

First we conducted 400 simulation runs using 4 vehicles performing trajectories similar to the ones pictured in Figure 7.2. At each simulation run they were started at random positions and orientations and traveled during $320s$. The observation period was $10s$. This is twice the value in most of the simulation runs of the CEKOLM algorithm presented in previous sections, since DCEKOLM requires the transmission of about two times the data of CEKOL. In Table 7.5 we can see the mean Euclidean error using three different algorithms: CEKOLM2, DCEKOLM with random communications and DCEKOLM with optimal communication protocol.

We can see that the DCEKOLM algorithm, when each vehicle chooses randomly another vehicle to communicate with, performs worst then the CEKOLM2 (when both of them have the same observation period of $10s$).

Although the use of optimal communication protocol slightly reduces the error of DCEKOLM, it is still worst than the CEKOLM2 algorithm. There is not much advantage choosing the best vehicle to communicate with because the current vehicle playing the role of the CAUV does not know the current position of the FAUVs, only their position the last time it observed them. Since the trajectories overlap each other, there might be significant changes in their positions since last communication.

In order to confirm that the bad results of the optimization of the communication protocol were due to the significant changes in vehicle position in between observations, we tested the algorithm in a scenario where each vehicle is operating

Algorithm	Mean error	Variance
CEKOLM2	4.7	3.7
DCEKOLM with random communications	5.1	3.7
DCEKOLM with optimal communication protocol	5.0	4.2
Prediction	10.1	19.0

Table 7.5: Comparison between CEKOLM2, DCEKOLM with random communications and DCEKOLM with optimal communication protocol. The vehicles are initialized anywhere inside the working area.

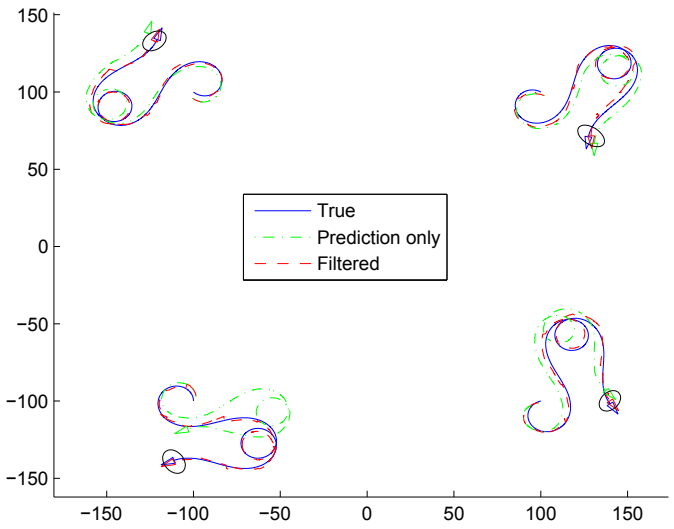


Figure 7.19: DCEKOLM with four vehicles operating in non overlapping areas

in a fixed area. These areas do not overlap and are arranged in a way such that the vehicles keep a more or less constant bearing to each other. In Figure 7.19 we can see the result of a simulation run in this scenario.

In Table 7.6 we can see the mean Euclidean error of 400 simulation runs in this scenario. As always, vehicles would start in random positions and orientations, but inside their respective areas. In this case we can see the DCEKOLM algorithm with the optimal communication protocol performs slightly better (a 7% improvement) than DCEKOLM without any communication strategy, and as good as CEKOLM2.

Algorithm	Mean error	Variance
CEKOLM2	4.1	2.3
DCEKOLM with random communications	4.4	2.1
DCEKOLM with optimal communication protocol	4.1	1.9
Prediction	10.1	19.0

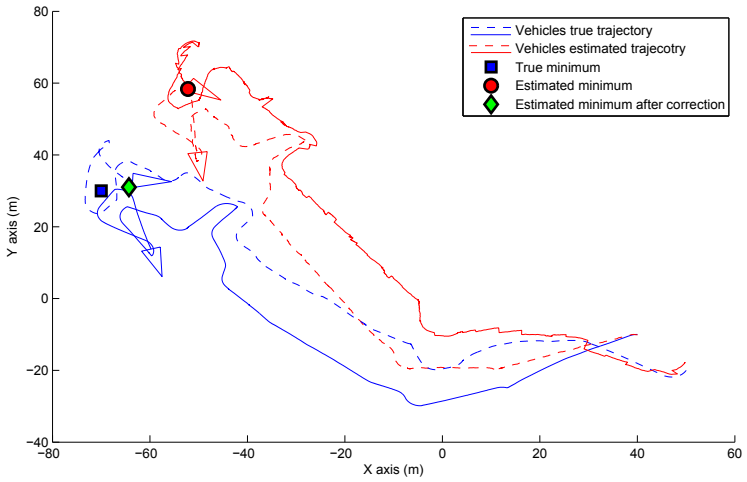
Table 7.6: Comparison between CEKOLM2, DCEKOLM with random communications and DCEKOLM with optimal communication protocol. The vehicles are initialized anywhere inside their respective working area, which are non overlapping.

7.5 Plume Tracing

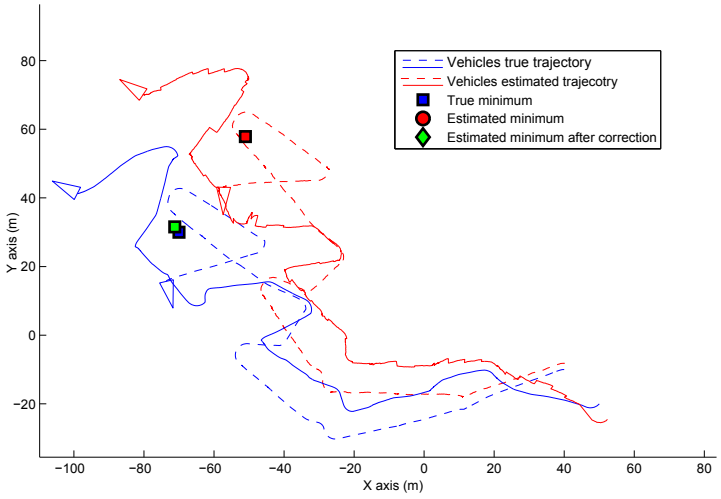
In this section we present the results of the cooperative plume tracing algorithm proposed in Section 6.3.

In Figure 7.20, we can see the result of two simulation runs of the proposed algorithm to find a minimum in a scalar field with two vehicles working cooperatively. They communicate with each other every $5s$ to exchange information to perform both the CEKOLM algorithm and the simplex downhill with uncertain vertices algorithm.

We can see that, although the vehicles are not able to keep an accurate global localization (estimated trajectories in dashed red diverge from the true trajectories in solid blue), they are relatively well localized, as expected. Once the estimated minimum point $\hat{V}_{min}$ (red circle) has been found and the vehicles update their estimated positions by using some global localization technique (such as GPS after surfacing or communicating with a surface vehicle), a homogeneous transformation T mapping the final estimated position of the first vehicle $\hat{X}_1$ into the final true position X_1, that is $X_1 = T\hat{X}_1$, with intermittent observations. In this way, it is possible to apply T to $\hat{V}_{min}$, to obtain the point of minimum of the scalar field $\hat{V}_{mincor} = T\hat{V}_{min}$ (green diamond) in a global reference frame.



(a)



(b)

Figure 7.20: Results of two simulation runs of the Plume Tracing algorithm using simplex downhill with uncertain vertices. The simulations employed two vehicles performing cooperative localization using CEKOLM2 algorithm. Once the vehicles find the estimated minimum, relative to their reference frame, this value can be corrected in order to obtain the point of minimum relatively to a global reference frame.

Chapter 8

Convergence of SLAM with Moving Features and Partial Observability

In this chapter, we provide simulation evidence for a simple instance of SLAM with moving features in the context of partial observability provided by the range sensors. After introducing some key concepts underlying the observability analysis of the estimation scheme, we provide arguments in Section 8.1 that plausibilify the observed results in the convergence analysis for the nonlinear case. We also study the convergence of the SLAM with dynamic map considering increasing complexity scenarios in Section 8.2. We also show some of the problems that might arise from the partial observability of the range only sensor in SLAM in Section 8.3 and how we dealt with it.

8.1 Observability Analysis

In this section, we consider the observability analysis regarding the localization of multiple vehicles which are able to take relative observations between them. As discussed in [53], observability implies bounded error covariance in the localization estimate of the robots. In [77], the authors conducted an observability analysis of a linearized system composed of several robots equipped with encoders and sensors able to measure their relative pose. They showed that the system is observable

only when one of the robots has some global positioning system. In [4] the authors consider the relative motion between two vehicles and, by using Lie derivatives, perform an observability analysis and derive a metric to quantify how much the observability of the system is dependent on the type of relative motion of the vehicles. In [53] the authors consider several types of relative observation between a team of vehicles (such as range and bearings in different reference frames) and use Lie derivatives to construct the observability matrix for all the cases.

In order to study the observability of the proposed system in this thesis, let us first revise some key concepts pertaining the theory of observability for linear and for nonlinear systems .

Given a linear system described by Equation (8.1), it is said to be observable if, given any time interval $[0, T]$ and the input and output history in this time interval, i.e., $\{(u(t), y(t)) : t \in [0, T]\}$, it is possible to determine the value of state at the initial time, i.e., $x(0)$.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{8.1}$$

In order to determine the observability for linear systems one may investigate the observability matrix O, given by, [6],

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}. \tag{8.2}$$

A linear system is said to be observable if the rank of O is equal to n (dimension of the state vector), also called full rank, [6].

Let us remind some well established results for the state estimation of linear systems, [69]. Without any loss of generality, we may fix matrix $D = 0$. Let us denote $\hat{x}$ to be the estimate of the state variable x, and $e = \hat{x} - x$ the error between the true state and its estimate. The state estimate $\hat{x}$ satisfies the following ordinary differential equation

$$\dot{\hat{x}} = A\hat{x} + Bu - K_e(y - C\hat{x}).$$

Here, the gain K_e has to be chosen so that the error e is driven asymptotically to zero. It is immediate to conclude that the error satisfies the equation

$$\dot{e} = (A + K_e C).$$

It is easy to show, [69], that if the system (or, equivalently, the pair (C, A)) is observable, then it is always possible to find a matrix K_e so that the poles of the characteristic polynomial of the matrix $A + K_e C$ can be placed arbitrarily, and, in particular, in the interior of the left hand side of the complex plane. This amounts to the existence of a matrix K_e with the desired property which can be easily computed with several schemes, among which the Ackermann formula.

Since the system that describes the problem addressed in this thesis, namely, the motion model of an underwater vehicle, is nonlinear, that is, of the type

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t))) \\ y(t) &= h(x(t), u(t)), \end{aligned} \tag{8.3}$$

and, therefore, the previous analysis can not be applied. In particular, the observability criterion has to be defined in a different way since the mere usual linearization of the nonlinear system may lead to erroneous results, [47]. We could, for example, formulate the error of the estimate in much the same way as the linear case, so that

$\dot{e} = f(\hat{x}) - f(x) + L(h(\hat{x}) - h(x))$, and linearize it about its fixed point $e = 0$. However, this linearization is a function of the true state x which is unknown (and also not a fixed quantity), [74]. So we need to use some nonlinear techniques to determine the observability of nonlinear systems. In [74] the author shows that there are mainly three methods for this purpose: Lyapunov methods, extended linearization and Lie-algebraic. Each one of these has its advantages and disadvantages. However, since each technique is suited only for problems with special structure, comparisons are difficult to make. Here we will use Lie-algebraic to perform an observability analysis of the system proposed.

It is shown that, when the data satisfies sufficiently smoothness assumptions, then the nonlinear system (8.3) is observable if and only if a certain observability matrix is full rank. This observability matrix is computed with the help of Lie derivatives.

The observability matrix O of the system (8.3) computed with the help of Lie derivatives is as follows

$$O = \begin{bmatrix} O_0 \\ O_1 \\ \vdots \\ O_{n-1} \end{bmatrix} = \frac{\partial l(x)}{\partial x},$$

where

$$l(x) = \begin{bmatrix} L_f^0(h) \\ L_f^1(h) \\ \vdots \\ L_f^{n-1}(h) \end{bmatrix} = \begin{bmatrix} h \\ \dot{h} \\ \vdots \\ h^{n-1} \end{bmatrix}.$$

So, by dropping the argument h in $l(x)$, i.e., $L_f^i = L_f^i(h)$, since

$$O_0 = \frac{\partial L_f^0}{\partial x} = \frac{\partial h}{\partial x},$$

hence

$$L_f^1 = \dot{h} = \frac{\partial h}{\partial x}\frac{dx}{dt} = O_0\dot{x},$$

and more generally

$$L_f^d = \frac{\partial}{\partial x}[h^{d-1}]\frac{dx}{dt} = \frac{\partial}{\partial x}[L_f^{d-1}]\dot{x} = O_{d-1}\dot{x},\ d = 1, ..., n. \tag{8.4}$$

Now let us apply the observability matrix using Lie derivatives to the system proposed in this thesis, that is, a multi-vehicle system. Let us first consider that the system has just the CAUV and 1 FAUV, that is, there are only 2 AUVs. In this case, the state vector is given by (notice that, from here on, we define the state vector of each vehicle by X_i compared to the x axis component of each vehicle i)

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = [x_1\ y_1\ \theta_1\ x_2\ y_2\ \theta_2]^T,$$

where X_1 and X_2 are the vector state for the CAUV and FAUV respectively. Considering once again the unicycle model described in Equation (3.10) and repeated here

$$\begin{bmatrix} \dot{x_v} \\ \dot{y_v} \\ \dot{\theta_v} \end{bmatrix} = \begin{bmatrix} v_x\cos\theta_v - v_y\sin\theta_v \\ v_x\sin\theta_v + v_y\cos\theta_v \\ v_\theta \end{bmatrix},$$

for each vehicle v, the differential equations for the group of two vehicles are then given by

$$\dot{X} = \begin{bmatrix} \bar{X}_1 & 0 \\ 0 & \bar{X}_2 \end{bmatrix} u,$$

where

$$\bar{X}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix},\ i = 1, 2 \tag{8.5}$$

and $u = [u_1\, u_2]^T$, $u_i = [v_{x_i}\, v_{y_i}\, v_{\Theta_i}]$, is the control signal.

If the system has no global positioning system, the observation function, which gives the range between the AUVs, is given by:

$$h(X) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Notice that the proprioceptive information (encoders or IMU) do not appear in the observation function, but in the control signal u_i. If, in order to make the computations a lot simpler, we consider the observation is actually given by half of the square of the distance, as suggested in [4], that is,

$$h(X) = \Delta_{12} = \frac{1}{2}((x_1 - x_2)^2 + (y_1 - y_2)^2),$$

the first line of the observability matrix O_0 will be given by

$$O_0 = \frac{\partial L_f^0}{\partial X} = \frac{\partial h}{\partial X} = [x_1 - x_2, y_1 - y_2, 0, x_2 - x_1, y_2 - y_1, 0].$$

The vector L_f^1 can be computed by

$$\begin{aligned} L_f^1 &= (x_1 - x_2)(u_1c_1 - u_2s_1) - (x_1 - x_2)(u_4c_2 - u_5s_2) \\ &\quad +(y_1 - y_2)(u_2c_1 + u_1s_1) - (y_1 - y_2)(u_5c_2 + u_4s_2), \end{aligned}$$

where $c_i = cos(\theta_i)$ and $s_i = sin(\theta_i)$.

By using Equation (8.4), the second line O_1 can be computed by $\frac{\partial L_f^1}{\partial X}$ as follows

$$O_1^T = \begin{bmatrix} u_1c_1 - u_4c_2 - u_2s_1 + u_5s_2 \\ u_2c_1 - u_5c_2 + u_1s_1 - u_4s_2 \\ (y_1 - y_2)(u_1c_1 - u_2s_1) - (x_1 - x_2)(u_2c_1 + u_1s_1) \\ u_4c_2 - u_1c_1 + u_2s_1 - u_5s_2 \\ u_5c_2 - u_2c_1 - u_1s_1 + u_4s_2 \\ (x_1 - x_2)(u_5c_2 + u_4s_2) - (y_1 - y_2)(u_4c_2 - u_5s_2) \end{bmatrix},$$

and so on for the rest of the observability matrix. By using a symbolic computation software, the computed row reduced echelon form of the observability matrix, excluding the all zero rows, will be:

$$O = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & y_1 - y_2 \\ 0 & 1 & 0 & 0 & -1 & x_2 - x_1 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}.$$

As expected, this observability matrix has rank $3 < 6$, and thus the system is not observable. As we said above, the observation function considered here is the range between the AUVs. However, proprioceptive information (dead reckoning) is used during the prediction steps of the algorithm (in the REKF in each vehicle). These additional information is the one that provides the estimation of the state of the system at each time step, i.e., makes each vehicle's state observable, although with unbounded error as time grows. But even if the covariance error in position grows without bound, the algorithms proposed are able to estimate the states with smaller error, a lot lower when compared with a prediction only result, as show in Figures 8.1 and 8.2.

If we add a GPS position measurement for one of the vehicles (the FAUV for example), so that the GPS updates only the position (x_2, y_2), the observation function will be given by

$$h(X) = \begin{bmatrix} \Delta_{12} \\ x_2 \\ y_2 \end{bmatrix}.$$

In this case, after computing the new observation matrix, we can see that it has a rank $5 < 6$, hence, the system is yet not observable. But if the GPS updates the pose (position and heading) of the CAUV, the observation function will be:

$$h(X) = [\Delta_{12} \;\; x_2 \;\; y_2 \;\; \theta_2]^T.$$

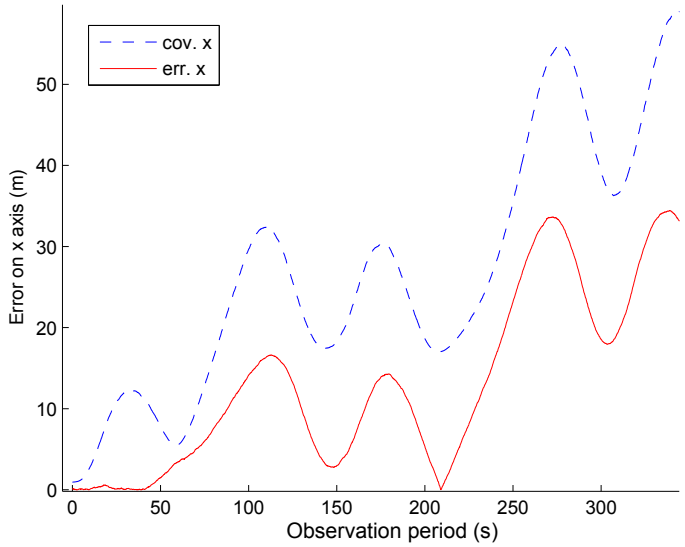


Figure 8.1: Absolute error on x axis for the CAUV in a prediction only execution and the region of confidence ($3\sigma_x$) calculated based on the covariance matrix $P(t)$.

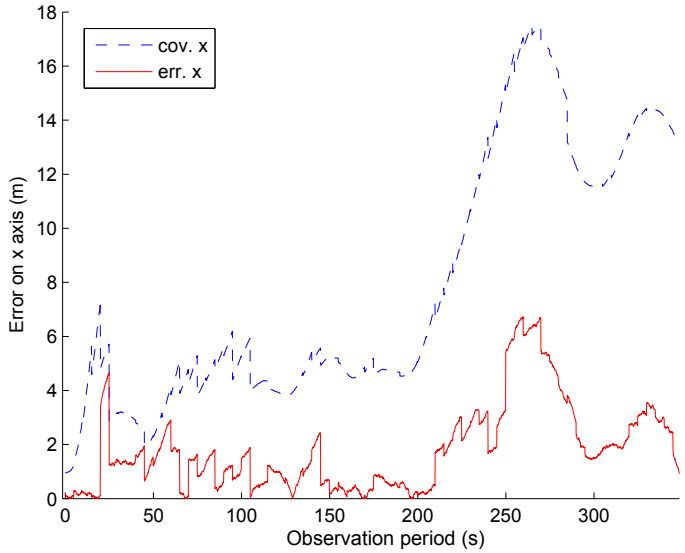


Figure 8.2: Absolute error on x axis for CAUV in an execution of CEKOLM2, with no GPS update, observation period of 5 seconds and the region of confidence ($3\sigma_{xx}$) calculated based on the covariance matrix $P(t)$

Now the observability matrix is full rank, and the system is observable. In this case, if the FAUVs have GPS update every time step of the algorithm, the system can be considered a simple localization problem with known landmarks. Let us extend this result for n AUVs. If you consider that at some instant the CAUV has observed all the FAUVs, and if only one of them (the k^{th} AUV) has absolute positioning capabilities, as show in Equation (8.6), the system will be observable.

$$h(X) = [\Delta_{12}\ \Delta_{13}\ \cdots\ \Delta_{1n}\ x_k\ y_k\ \theta_k]^T \tag{8.6}$$

This is shown in Figure 8.3. We can observe that during more then 10 minutes of simulation (the AUV moves 700 meters) the error is not increasing (varies from about 1 to 3 meters). Since the observation period is 5 seconds, in between observations the error can grow, but the overall error is still bounded. In this simulation, the FAUV1 updates his pose by GPS constantly, that is, it must be on the surface all the time.

So, by considering that the CAUV communicates with all other FAUVs at each time step, if one of the vehicles involved in the CEKOLM algorithm has global positioning capabilities, the system becomes observable. However, for underwater vehicles it is hard to have a global positioning system, since the vehicle cannot use GPS when submerged. So, if none of the vehicles are able to acquire low uncertainty estimate of its pose, the vehicles are able to keep only an accurate relative positioning. This can be seen by considering the position of each vehicle in the system relative to the reference frame fixed in one of them (vehicle v). But this is the same as the vehicle v having a zero uncertainty about his pose, after all, he is always in the origin of his own reference frame. This means that, with this consideration, the system is observable, and thus the relative positioning will have bounded error.

The two simplified schemes of the algorithms running in one of the vehicles are shown in Figure 8.4: a) vehicle observing feature (another vehicle) with unbounded

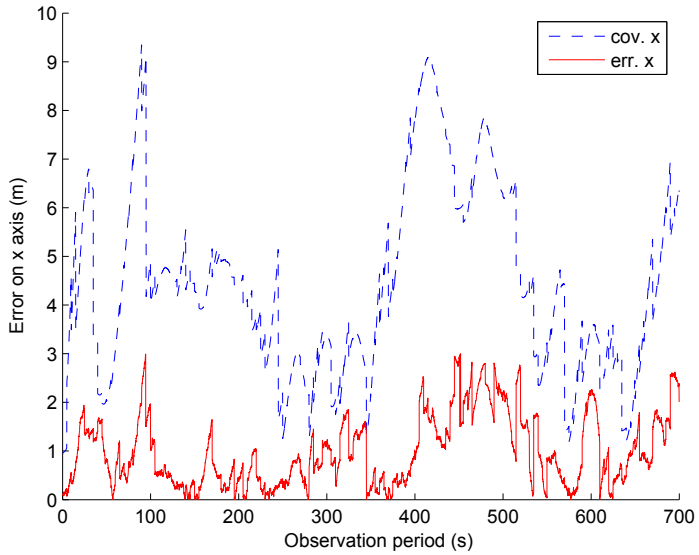


Figure 8.3: Absolute error on x axis for CAUV in an execution of CEKOLM2 with 3 FAUVs, observation period of 5 seconds, and the region of confidence ($3\sigma_{xx}$) calculated based on the covariance matrix $P(t)$. FAUV1 pose is constantly updated by GPS.

pose error and b) vehicle observing feature with known (bounded error) pose. In the first case, the range measurement is able to reduce the global localization uncertainty of the vehicle and also keep a bounded error relative pose between the vehicle and the feature. In the second case the system becomes observable and thus the error of the estimated state of the vehicle will be bounded. As discussed in the last paragraph, the transition between the two cases can be done by a change in the reference frame: in a) we have a global reference frame and in b) we have a feature centered reference frame.

We showed that if one of the vehicles is able to update its pose by a global positioning system all the time (at each time step) the system becomes observable. However, as noted previously, the FAUVs will not update their pose by GPS constantly, but just when its possible to surface (this depends on the mission being executed). So what happens to the error in this circumstances?

In Figure 8.5 we show the error on the x axis for the CAUV in an simulation of

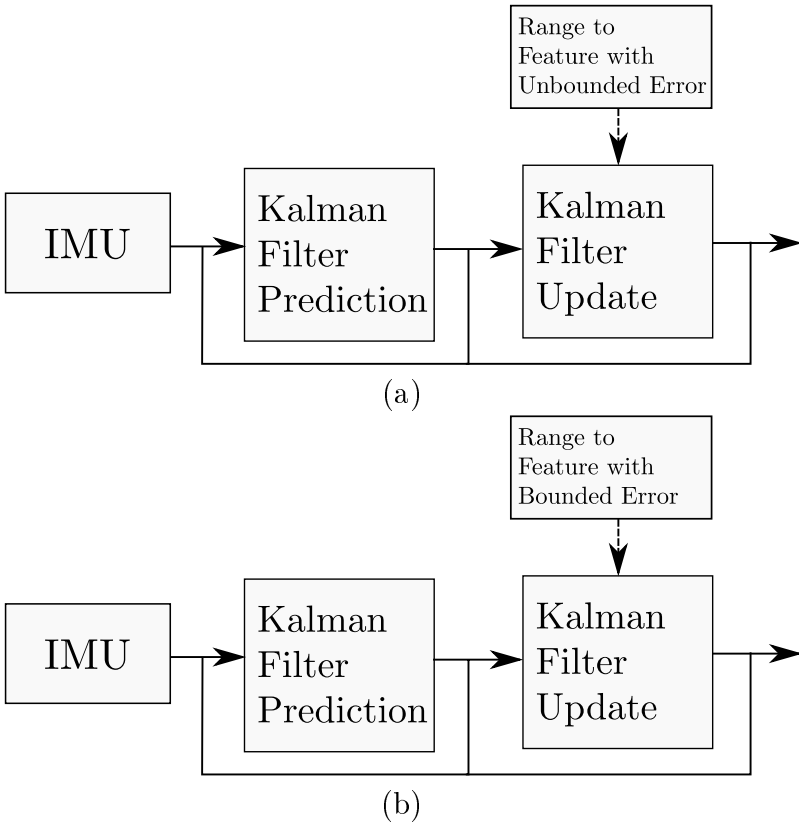


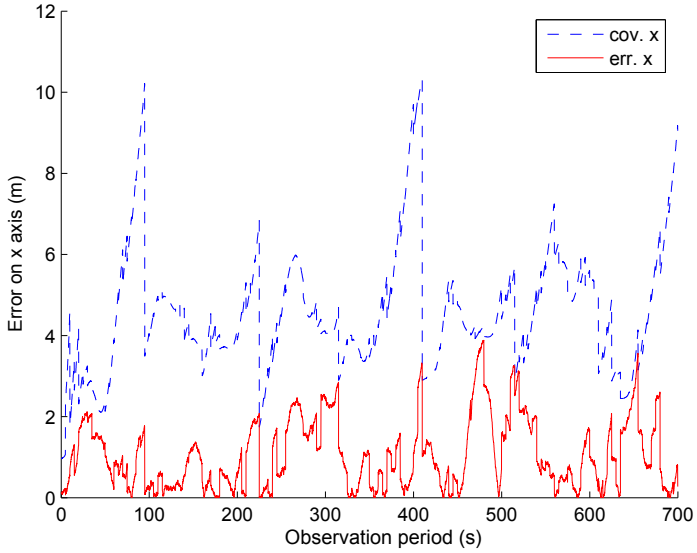
Figure 8.4: Two possible schemes of the algorithms. In a) the vehicle observes a feature with unbounded localization error. In this case, although the estimated state error still grows without bounds, the fusion of the asynchronous range sensor information (represented by a dashed line) is able to reduce the global error and also to bound the relative error of the vehicle/feature pair. In b) if the vehicle observes a feature that has a bounded error pose, the range measurement will bound the pose error of the vehicles as well.

CEKOLM2 with 3 FAUVs. The observation period is 5 seconds. In this experiment, one FAUV is randomly selected to update its pose by GPS every 100 seconds. This means that each FAUV can stay about 5 minutes underwater, before having to surface in order to update its pose, and the CAUV, of course, can be submerged during the whole experiment. You can notice that once again the error is not increasing during the simulation. It is higher than in Figure 8.3, as expected, but it is still bounded. Whenever a FAUV that has just updated his position is observed, the error in the CAUV pose greatly decreases, as we can see in Figure 8.3 (a), right after time $t_k = k * 100$, $k = 1, 2, ...$, when the CAUV observed an FAUV that updated it pose recently by GPS. After this observation, the CAUV will have a better pose estimate, and can "propagate" this better estimate through the others FAUVs, helping them to better estimate their pose.

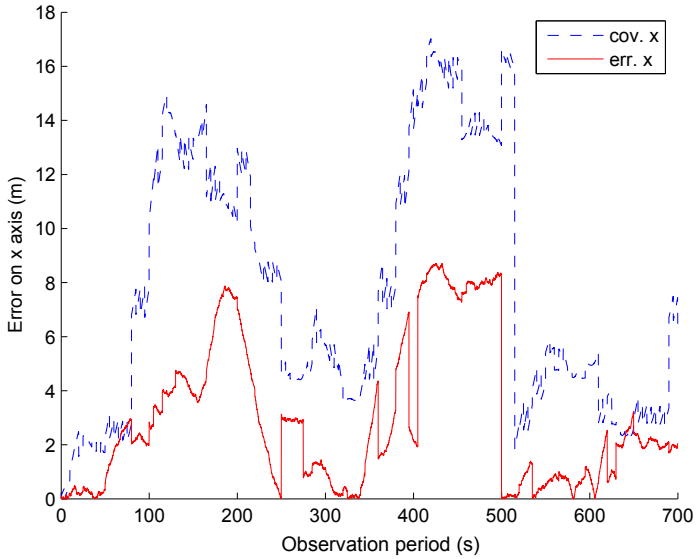
8.2 Convergence Analysis Based on SLAM Properties

It the previous Section it was shown that, by considering that the CAUV communicates with all other FAUVs at each time step, the CEKOLM algorithm becomes observable if one of the vehicles involved in the system has global positioning capabilities. However, for underwater vehicles, it is hard to have a global positioning system. So, if none of the vehicles is able to acquire low uncertainty estimate of its pose, the vehicles are able to keep only an accurate relative positioning. This convergence analysis was done by constructing and examining the observability matrix of the system.

In this section, we will approach the convergence analysis of the SLAM problem with moving features in a different way. We shall consider cases of increasing complexity to verify the limits of the convergence properties of the SLAM algorithms when applied to a dynamic map. In this analysis we take into account the slow



(a)



(b)

Figure 8.5: Absolute error on x axis for the CAUV (a) and for FAUV1 (b) in an execution of CEKOLM2 with 3 FAUVs, observation period of 5 seconds, and the region of confidence ($3\sigma_{xx}$) calculated based on the covariance matrix $P(t)$. One FAUV randomly updates its pose by GPS every 100 seconds.

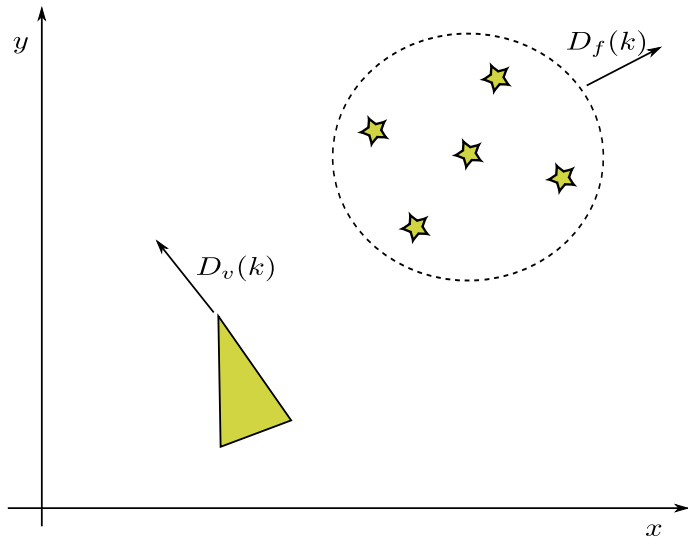


Figure 8.6: Features moving in a certain fixed formation with an a priori known motion by the vehicle

communication, and do not consider that the CAUV communicates with all other vehicles at each time step. We will begin with the case where the features move in a certain fixed unknown formation and their motion is known a priori by the vehicle. Then, we will consider the case where this motion is unknown. Next, we will investigate the case where the formation is not fixed - the features may change their formation configuration during the execution of the algorithm - but with known motions. Finally, we will consider the case where the features are free to move in any way, but their motion is known by the vehicle.

8.2.1 Features Moving in a Certain Fixed Unknown Formation with an *a Priori* Known Motion by the Vehicle

In this case, the features must keep a fixed formation moving around the working space. We assume that this motion is known by the vehicle. This scenario is represented in Figure 8.6, where $D_v(k)$ and $D_f(k)$ represents a displacement taken by the vehicle and the feature formation, respectively, at time instant k.

As usual in the SLAM problem, the state vector $X(k)$ is composed of the vehicle pose, $X_v(k)$, and map features parameters $X_{fi}(k)$ (represented in the global or world

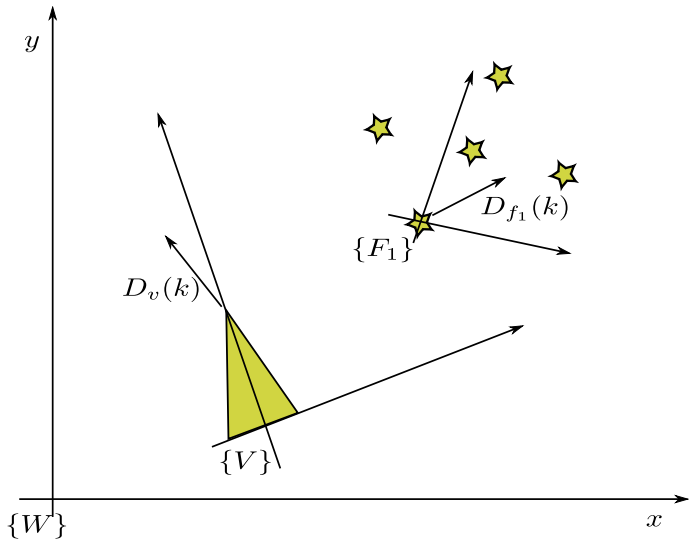


Figure 8.7: Displacement vector of the vehicle $D_v(k)$ and displacement vector of the first feature $D_{f1}(k)$ and the reference frame $\{V\}$ which is attached to the vehicle and $\{F_1\}$, attached to the first feature .

coordinate frame $\{W\}$), that is

$$X(k) = X^W(k) = [X_v(k)\ X_{f1}(k)\ X_{f2}(k)\ \ldots\ X_{fv}(k)]^T.$$

However, since the features are dynamic, the usual SLAM system model will be modified. Now, the features model will incorporate their motion. This way, the system model will be

$$\begin{aligned} X_v(k+1) &= f_v(X_v(k), u_v^V(k), q_v(k), k) \\ X_{fi}(k+1) &= f_f(X_{fi}(k), u_{f_i}^{F_i}(k), q_{f_i}(k), k), \end{aligned} \tag{8.7}$$

where $u_v^V(k)$ and $u_{f_i}^{F_i}(k)$ are the control signals of the vehicle and of the features, at time instant k, represented in the vehicles reference frame $\{V\}$ and each feature reference frame $\{F_i\}$, respectively. $q_v(k)$ and $q_{f_i}(k)$ are the process noises for the vehicle and for each feature, respectively. This scheme is represented in Figure 8.7.

Recall that since the features must keep a formation, we can represent the feature formation position and orientation as the pose of $\{F\}$ relative to $\{W\}$, where $\{F\}$ is a reference frame fixed on the feature formation, as shown in Figure 8.8. The control

signals $u_f^F(k)$ will then represent the motion of the feature formation reference frame. This control signal is contaminated by a noise $q_F(k)$. Knowing, at time k, the control signal of the feature reference frame, $u_f^F(k)$, the topology of the formation, $\tau(k)$, and the position of the other features, $X_f(k)$, allows for each feature i to compute its own control signal $u_{f_i}^{F_i}(k) = g_i(u_f^F(k), \tau(k), x_f(k))$, where g_i is a function so that the formation will be kept. In this way the features model will be given by:

$$X_{fi}(k+1) = f_f(X_{fi}(k), g_i(u_f^F(k), \tau(k), x_f(k)), k), \tag{8.8}$$

Since we consider that all the features keep their formation (in practice the formation is not perfect and there will be some fluctuations, but the effect of this fluctuations will be discussed later), it is possible to consider that all the features are static relative to $\{F\}$, and "transfer" their motion to the vehicle. That is, we shall represent the state vector X^W (vehicle pose and feature positions relative to the world) relative to the feature formation frame of reference $\{F\}$, denoted by X^F. In this way, we will add to the displacement vector of the vehicle $D_v(k)$, the negative of the features displacement vector $D_f(k)$, so that the resulting displacement of the vehicle, at time instant k, relative to the reference frame $\{F\}$ is given by $D_v^F(k) = D_v(k) - D_f(k)$. This is represented in Figure 8.8.

If we consider a 2D model (unicycle model) for the vehicle and feature formation motion function, the state vector of the vehicle (and also for the feature formation, or feature reference frame) relative to the world reference frame will be composed by the position (x, y) on the plane and the orientation θ (vehicles local reference frame relative to the global reference frame), as show in Equation (3.5).

$$X_v(k) = [x_v \; y_v \; \theta_v]^T \tag{8.9}$$

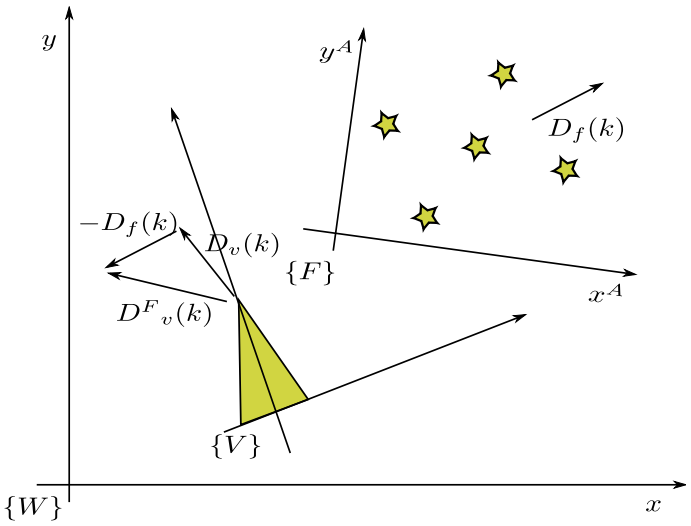


Figure 8.8: Compensating the motion of the feature formation reference frame $\{F\}$ by adding its negative vector $-D_f(k)$ to the displacement vector of the vehicle $D_v(k)$, resulting in a displacement vector $D_v^F(k)$ representing the displacement of the vehicle relative to $\{F\}$.

The control input would be:

$$u^V(k) + q_v(k) = [v_x^V(k)\ v_y^V(k)\ v_\Theta^V(k)]^T,$$

where v_x^V and v_y^V are the velocity in the x and y axes of the vehicles local reference frame $\{V\}$, v_Θ^V is the turn rate of the vehicle at each time step, and q_v is the process noise of the vehicle . This process is represented in Figure 3.2. The discrete motion model of the vehicle is given by Equation (3.10) (notice that since the noise is already incorporated in $u^V(k)$, the model is now a function only of $X_v(k)$ and $u^V(k)$).

$$\begin{aligned} X_v(k+1) &= f_v(X_v(k), u^V(k)) \\ \begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} &= \begin{bmatrix} x_v(k) + v_x^V(k)\cos\theta_v(k) - v_y^V(k)\sin\theta_v(k) \\ y_v(k) + v_x^V(k)\sin\theta_v(k) + v_y^V(k)\cos\theta_v(k) \\ \theta_v(k) + v_\theta^V(k) \end{bmatrix} \end{aligned} \tag{8.10}$$

If we want to express the state vector X relative to $\{F\}$, we can see clearly that

$$X_{fi}^F(k+1) = X_{fi}^F(k),$$

that is, expressed in $\{F\}$ the features are static. In order to represent the pose of the vehicle relative to $\{F\}$ we must consider two consecutive motions at time instant k:

1. The motion in the vehicle caused by his own control signal. This first motion will take vehicle from $X_v^F(k)$ to what we called $X_v^F(k+1)'$.
2. The motion on the vehicle caused by the motion of $\{F\}$ itself. This will take the vehicle from $X_v^F(k+1)'$ to $X_v^F(k+1)$.

By considering first only the motion caused on the vehicle due to his own control vector, since the control signal $u^V(k)$ is represented in the vehicles frame, the system model Equations (8.7) represented in the reference frame $\{F\}$ will now be :

$$X_v^F(k+1)' = f_v(X_v^F(k), u_v^V(k), q_v(k)). \tag{8.11}$$

Now, we have to consider the motion caused on the vehicle due to the motion of the reference frame $\{F\}$. Let us represent the control vector, in the world frame, that causes the motion of $\{F\}$ by:

$$u_F(k) + q_F = [v_x \; v_y \; v_\Theta]^T,$$

where q_F is the feature formation process noise.

The motion of the vehicle represented in $\{F\}$ is caused by the rotational and translational component of $u_F(k)$. By considering first the translational component $[v_x \; v_y]^T$ we shall have

$$X_v^F(k+1) = X_v^F(k+1)' - [v_x^F \; v_y^F]^T. \tag{8.12}$$

By considering now the motion caused by the translational part of $u^F(k)$ in each component of $X_v^F(k) = [x_v^F \; y_v^F \; \theta_v^F]^T$:

$$\begin{aligned}
x_v^F(k+1) &= cos(-v_\Theta)x_v^F(k+1)' - sin(-v_\Theta)y_v^F(k+1) \\
y_v^F(k+1) &= sin(-v_\Theta)x_v^F(k+1)' + cos(-v_\Theta)y_v^F(k+1) \\
\theta_v^F(k+1) &= \theta_v^F(k+1)' - v_\Theta
\end{aligned}$$

If we put all the motions of the vehicle caused by the motion of the reference frame $\{F\}$ we can see that

$$X_v^F(k+1) = f_v(-u_F, X_v^F(k+1)'). \tag{8.13}$$

This gives us the following system model, representing the state vector relative to the reference frame $\{F\}$:

$$\begin{aligned} X_{fi}^F(k+1) &= X_{fi}^F(k), \\ X_v^F(k+1)' &= f_v(X_v^F(k), u_v^V(k), q_v(k)), \\ X_v^F(k+1) &= f_v(-u_F, X_v^F(k+1)', q_F(k)). \end{aligned} \tag{8.14}$$

Now, we just have to use the regular Kalman Filter update equations to compute the predicted covariance matrix P_p^F. This can be simply done by considering the two separate motions taken by the vehicle: first, we consider the change in the covariance matrix of the vehicle, given by the motion caused by its own control signal, which will give us what we called $P_p^F(k+1)'$, and, then, consider the changes given by the motion caused by the motion of the frame $\{F\}$, giving us $P_p^F(k+1)$.

The Kalman update equation for the covariance matrix is given by:

$$P_p(k+1) = D_X f P(k) D_X f^T + D_u f Q(k) D_u f^T,$$

where $D_X f$, $D_u f$ and $D_X h$ denote, respectively, the Jacobians of f w.r.t. x and w.r.t. u. Remember that when there is no observation $P(k) = P_p(k)$. So, to compute $P_p^F(k+1)'$ we will have

$$P_p^F(k+1)' = D'_{X_v^F} f_v P(k) D'_{X_v^F} f_v^T + D'_{u_v^V} f_v Q_v(k) D'_{u_v^V} f_v^T.$$

The Jacobians of the vehicle motion function f_v shown in Equation (3.10) are then given by:

$$D'_X f_v(x,u) = \begin{bmatrix} 1 & 0 & -v_x(t)\sin\theta_v(t) - v_y\cos\theta_v(t) \\ 0 & 1 & v_x(t)\cos\theta_v(t) - v_y\sin\theta_v(t) \\ 0 & 0 & 1 \end{bmatrix} \tag{8.15}$$

$$D'_u f_v(x,u) = \begin{bmatrix} \cos\theta_v(t) & -\sin\theta_v(t) & 0 \\ \sin\theta_v(t) & \cos\theta_v(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.16}$$

Similarly, $P_p^F(k+1)$ will be given by

$$P_p^F(k+1) = D_{X_v^F} f_v P(k+1)' D_{X_v^F} f_v^T + D_{u_F} f_v Q_F(k) D_{u_F} f_v^T,$$

and in this case, we can show that

$$\begin{aligned} D_X f_v(x,u) &= D'_u f_v(u,x) \\ D_u f_v(x,u) &= D'_X f_v(u,x) \end{aligned}$$

Now, we still need to analyze the fluctuations on the formation of the features. We imposed that the features must keep a formation, but of course this formation is not perfect since each feature is moving (subject to noise) through the environment. At a given time instant some of the features might be a certain distance $d(k)$ from its ideal position (depending on the control algorithm used for keeping the formation). We will assume that $d \sim \mathcal{N}(0, \sigma_d^2)$, that is, the distance the a feature keeps from its ideal formation position is given by a normal distribution with zero mean and variance σ^2. In this way we can incorporate this fluctuations in the sensor noise covariance matrix R to get a new measurement noise covariance matrix $R_n = s(R,d)$, where the function s is given by the type of sensor being used. For example, if the considered sensor is range-only, then the function can be approximated by $s(R,d) = R + \sigma_d^2$.

At this stage, we completely converted this first case of SLAM with moving features in formation with known motion to the original SLAM problem relative to the reference frame $\{F\}$ attached to the features. This means that we can ensure all the convergence properties proved to the SLAM relative to $\{F\}$ - the uncertainty of the features decrease and the correlation between them increase as observations are made. In the limit, as the number of observations increases, the landmark estimates become fully correlated and given the position of any feature, the others can be determined with absolute certainty. Also, in the limit, the error in the absolute location (relative to $\{F\}$) of every feature reaches a lower bound determined only by the error that existed when the first observation was made.

Notice that since the reference frame $\{F\}$ moves relatively to the world frame and this motion is contaminated by a noise $q_F(k)$, the algorithm cannot guarantee a lower bound for the error in the absolute location of the features relative to the world frame. However, if the noise $q_F(k) = 0$, then this property will hold true.

We showed that it is possible for a vehicle to create an accurate relative map of a group of features moving in formation, by just making a change in the representation of the problem. If we do not represent the system relative to $\{F\}$ but to $\{W\}$ instead, we still can create the relative accurate map, and although the error in the pose estimation of the vehicle and features will not be bounded, it can still be greatly reduced by the SLAM algorithm with the moving features (this was shown, in section 7.1, by simulations with the CEKOLM algorithm operating in a even more generic scenario).

8.2.2 Features Moving in a Certain Fixed Unknown Formation with an Unknown motion

In this case, the features also keep a fixed formation, but this motion is not known by the vehicle. So there is the need for the vehicle to estimate the motion done by

the features. Hence, we will incorporate the velocity vector V_F of the features in the state vector: V_{Fx} and V_{Fy}, representing the linear velocity of the feature formation in the x and y axes, and $V_{F\theta}$, representing the angular velocity (in a 2D scenario):

$$X(t) = [X(k)_v \; X(k)_{f1} \; X(k)_{f2} \; \ldots \; X(k)_{fn} \; V_{Fx} \; V_{Fy} \; V_{F\theta}]^T.$$

Notice that since the velocity of the features is unknown, the vehicle cannot compensate for it at the beginning of the execution of the algorithm. First the vehicle must estimate the velocity of the features. It can obtain this initial estimate of the formation velocity when it observes a minimum set of two features at least two times. This observations will give two pairs of points that enables the computation of a 2D spatial transformation.

Suppose that the vehicle observes both features $f_1$and f_2 at time instants k_1 and k_2: $X_{f_1}(k_1)$, $X_{f_2}(k_1)$, $X_{f_1}(k_2)$ and $X_{f_2}(k_2)$. By fixing the feature reference frame $\{F\}$ in the position of f_1 (and axis aligned with the estimated alignment of the world reference $\{W\}$), we can represent $X_{fi}(k)$ relative to $\{F\}$: $X^F_{fi}(k)$. Then we can compute the transformation matrix $T^{k_2}_{k_1}$ so that $X^F_{fi}(k_2) = T^{k_2}_{k_1} X^F_{fi}(k_1)$. The velocity vector V_F can thus be obtained by extracting the translational and rotational components of $T^{k_2}_{k_1}$ and dividing them by the elapsed time between k_1 and k_2. Notice that the velocity vector computed is represented in the reference frame $\{F\}$, just like $u^F_f(k)$ presented in section 8.2.1.

It might not be possible to observe two features at the same time k_1. If this is the case, then we can consider that the set of features were first observed during the interval $k_{21} = k_2 - k_1$ and observed the second time during the interval $k_{43} = k_4 - k_3$. To get a good estimate of V_F it is important that the dynamics of the feature formation is slow when compared to the time intervals k_{21} and k_{43}. It is also important that the time interval between observations $k_{32} = k_3 - k_2 \gg k_{21}, k_{43}$. In this way, the set of observed features will not have moved much during k_{12} and

Features	Feature pos	Time	Feature Uncertainty
1,3	$X_{f1}(k_1)$,$X_{f4}(k_1)$	k_1	$P_{f1}(k_1)$,$P_{f2}(k_1)$
1,2,4	$X_{f1}(k_2), X_{f2}(k_2), X_{f4}(k_2)$	k_2	$P_{f2}(k_2), \ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 8.1: Past positions and covariance matrix of the set of observed features relative to the world reference frame.

neither during k_{34}. If the number of features observed at time instants k_1 and k_2 are greater then two, it is possible to use an optimization method to better estimate V_F .

Once there is this first estimate of V_F, the vehicle can finally introduce the position estimate of the first observed feature into the state vector and start to compensate the motion of the features, just like in the last case presented. Once again, this is done by computing the vehicles position relative to $\{F\}$, X_v^F, and proceed the same way with the features, that is, starting to use the vector state relative to $\{F\}$, X^F. Since the velocity components of the feature formation V_{Fx}, V_{Fy} and $V_{F\theta}$ are already represented in $\{F\}$, they shall remain the same in X^F. But it is also important for the vehicle to keep a record of his motion without compensating for the feature formation motion and the observed features position, that is, relative to the world reference frame. Whenever a set of features is observed, the vehicle must keep a table with the last observed set positions relative to the world reference, the time instant (or interval) where the observation took place and the features uncertainty at this time instant, as show in Table 8.1.

In this way, whenever a set of features (or even just a subset) is re-observed, the vehicle can use the current observation and the one in the table to take another measurement of the features velocity buy computing the transformation matrix, extracting the translational e rotational components, and dividing them by the time elapsed between the observations of the set of features. With this new measurement of the feature formation velocity, we can use the EKF equations to update its

estimated value.

The error covariance of the estimated velocity R_v (error covariance matrix of the velocity sensor, which is actually a virtual sensor based on two consecutive sets of range sensor measurements to a set of feature) will be given by some function g of the set S_k composed of position uncertainty (covariance matrix) of each feature on the observed set at the current time instant k and the set S_{k_l} composed of the uncertainties of the observed features at the last time instant k_l the set was observed:

$$R_v = g(S_k, S_{k_l}). \tag{8.17}$$

Actually, R_v depends on the error covariance matrix R_n of the range sensor itself (already considering the fluctuations of the formation, as discussed above) and on the vehicle covariance matrix at the current time instant and the last time instant the feature was observed, but all of this values are already incorporated into the features covariance matrix P_{fi}.

For this scheme to work, the velocity vector of the feature formation must be either constant or change slowly when compared to the observation frequency (as discussed above), so the vehicle can have a good estimate of it. If V_F is constant, then the model equation will be $V_F(k+1) = V_F$. Otherwise, if it changes with time, then we must add a noise component q_{vF} to its model, as show in Equation (8.18), to allow its uncertainty to increase between observations, so the estimated value will not converge to a fixed value, but will be able to change the estimated velocity during the algorithm execution.

$$V_F(k+1) = V_F(K) + q_{vF} \tag{8.18}$$

Finally, we must add to the error covariance matrix $Q_F(k)$, that represents the error of the motion of the feature formation, the uncertainty of the estimated velocity

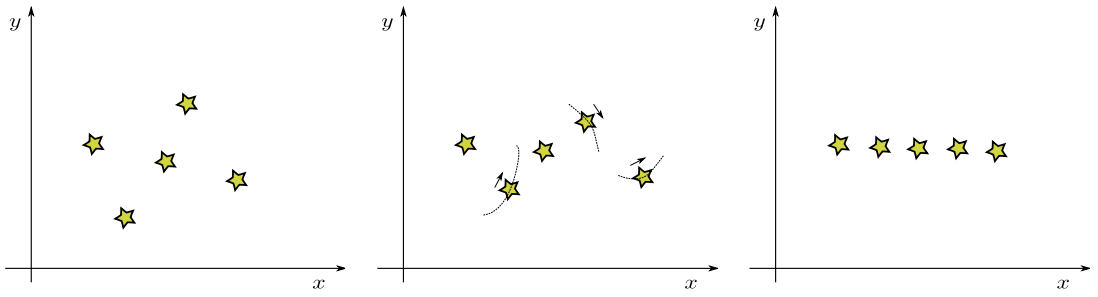


Figure 8.9: Features changing their formation: a) Features in a formation Formation1; b) Features in between formation; c)Features in a formation Formation2.

vector V_F. This is given by the EKF equations itself in the elements of matrix P relative to the positions in the vector state where V_F appears.

Now, it is possible to use the equations developed in Section 8.2.1 to once again convert the SLAM problem to a reference frame where the features are fixed. However, we must point that the convergence of this scheme is slower than the scheme showed in the previous section. This happens because, at the beginning of the algorithm, the vehicle must estimate the motion taken by the features, and also because the uncertainty of this estimated value contaminates the system.

8.2.3 Features Moving in a Dynamic Formation with Known Motion

In this case, we suppose that the features keep some certain possible formations most of the time, but it is also possible for them to change the topology of this formation during the execution of the mission. This means that the features can start the mission with a certain type of formation, lets say, Formation1, and after some time change to another formation Formation2. This process is shown in Figure 8.9.

The main idea for this case is for the vehicle to wait for the features to settle in a formation, and then apply the ideas presented in the first case scenario. Since the individual motion of each vehicles is known, it is theoretically possible to determine,

at a given moment, if the features are keeping a certain formation or changing it. However, possible errors in the known motion of the vehicles, and the minor adjustments those have to make to keep their formation, may prevent the vehicle to successfully detect a fixed formation for the features. But there is no need for the vehicle to actually wait for the features to settle in a new formation to start taking observations.

Suppose that, at time instant k, the features are in a certain formation and are already mapped by the vehicle with a certain uncertainty (covariance matrix $P(k)$). Suppose also that the features frame $\{F\}$ is fixed in one of the features. At $k+1$ the features start changing their formation, and do it until time step $k+n$. It is possible for the vehicle to use the knowledge of the motions of the features to predict their new positions while they move, and by taking observation to them, the vehicle is able to reduce the uncertainty of these predictions. In this way, at time $k+n$, the vehicle will have the estimated position of the features and a covariance matrix $P(k+n)$. From this time forward, since the features will be stationary relative to $\{F\}$, the vehicle will be able to reduce the uncertainty of the map by using the SLAM algorithm relative to the features frame $\{F\}$, until the features start to change the formation once again. This means that, while the features are changing formation (between $k+1$ and $k+n$), the uncertainty in their position relative to $\{F\}$ might increase (and the correlations between vehicles might decrease - when normalized), even if the vehicle takes observations to them. But once the formation is fixed again, the uncertainty will start to decrease as observations are made (and correlations increase), so the map starts converging to an accurate relative map.

Notice that the map relative to $\{F\}$ will converge only if the features spend enough time in a certain formation after each change. With this condition, we can guarantee the computation of a accurate relative map in between feature formation

changes. Of course, as in the first case, as the reference frame $\{F\}$ moves (contaminated by a noise $q_F(k) \neq 0$) relative world frame, we cannot guarantee a lower bound for the error in the absolute location of the features relative to the world frame.

8.2.4 Features Moving Freely with Known Motion

This case can be viewed as the previous one, where the features change from formation to formation, with the difference that now the features do not need to stay a minimum time in a certain formation. We consider that the features are in formation at any given time instant k (where the formation topology is given by the features position at k), and they keep changing their formation all the time. This means that we are always in the stage where the features are changing their formation, and therefore we cannot guarantee the convergence of the map, even relative to a frame fixed in one of the features. By the way, considering that the frame $\{F\}$ is fixed in one of the features does not help in any way, since the features do not stay in a fixed formation at any time, and thus are never static relative to $\{F\}$.

This is the case where the proposed algorithm CEKOLM fits in. In the algorithm, there is no restriction about the motion the features should take. They are able to move freely through the environment. Also, in CEKOLM, whenever a feature vehicle (FAUV) communicates with the central vehicle (CAUV), it communicates its control input, and so the motion of the feature is known to the vehicle.

Although we can not guarantee a map of relative locations with absolute precision, with some reasonable assumptions we may guarantee an upper bound for the uncertainty of the relative positions of the features. This means we can compute a relative map of the features with a certain minimum precision, defined by the noise characteristics contaminating the features motion, the noise characteristics for the exteroceptive sensor used, and the maximum time a feature can go unobserved.

Since it is of no use to define a frame $\{F\}$ fixed on the features, this time we will use the vehicles frame $\{V\}$ and represent the position of the features relative to this frame, so the state vector relative to $\{V\}$, X^V will be

$$X^V(k) = [0\ X^V_{f1}(k)\ X^V_{f2}(k)\ \ldots\ X^V_{fv}(k)]^T,$$

where the first term "0" stands for $X^V_v = (0,0,0)$ - the vehicle is always at the origin relative to its own frame. Making this change of frames and representing the position of the features in $\{V\}$ means that we are not incorporating the noise component of the vehicle q_v. The vehicle has no uncertainty about its pose in this frame. This is the same as to consider the whole system relative to the world frame $\{W\}$ but setting the vehicle process noise $q_v(k) = 0$, or enabling it to have some global positioning system. Hence, this case may be considered as a simple mapping problem! Remember that in a mapping problem the observations of the features taken from the vehicle (which is certain about its pose) are independent, and so no correlation is created between features. So we can analyze the system by examining only one feature and, then, extend the results for all features.

When the i^{th} feature is first observed at time instant k, it will be included in the map, and its initial uncertainty will be given by the sensor noise characteristics (supposing the sensor gives enough information to infer the feature position with a single measurement, and if this is not the case, we must combine several observations). If $g(X^V(k), y(k))$ is a function that gives the position of the feature based on the state vector $X^V(k)$ and the observation $y(k)$, then the initial uncertainty of this feature will be given by

$$P_{fi}(k) = D_y g R(k) D_y g^T,$$

were $R(k)$ is the noise covariance matrix of the sensor.

Notice that observing a new feature is like observing an existing feature with very high uncertainty (infinite uncertainty), and after the observation, the uncertainty of the feature is reduced to $D_y g R D_y g^T$. If the feature is observed again at time k', after moving and increasing its uncertainty, the new uncertainty after the observation $P_{fi}(k')$ must be such that $P_{fi}(k') \leq D_y g R D_y g^T$.

After the feature is observed the first time, it starts moving (actually, continues to move, but for the vehicle it is like it did not exist before the observation), with its own process noise $q_{fi} \sim \mathcal{N}(0, \sigma_{fi}^2)$. Let us assume that $\sigma_{fi}^2 < \infty$ and that $T_{max} < \infty$, where T_{max} is the maximum time steps a feature can go without being observed. While the feature is moving and not being observed, its covariance matrix starts growing, according to the Equation

$$P_{fi}(k+1) = D_{X_{fi}} f_f P_{fi}(k) D_{X_{fi}} f_f^T + D_{u_{fi}} f_f Q_{fi}(k) D_{u_{fi}} f_f^T.$$

So the uncertainty matrix $P(k+n), n = 1, 2...$, representing the uncertainty n steps after k, can be computed by some function l:

$$P(k+n) = l(P(k), u^{k,k+n}, Q^{k,k+n}),$$

where $u^{k,k+n} = [u(k), u(k+1), \ldots, u(k+n)]$, and similarly for $Q^{k,k+n}$. We know that, if the i^{th} features was observed at time k, then $P_{fi}(k) \leq D_y g R D_y g^T$, so after T_{max} steps, right before another observation, the maximum uncertainty $P_{max}(k+T_{max})$ (the criterion used for ordering covariance matrices is computing and comparing their volume, that is, their determinant) the feature i can have is

$$P_{max}(k+T_{max}) = l(D_y g R D_y g^T, u_{max}^{k,k+T_{max}}, Q^{k,k+T_{max}}), \tag{8.19}$$

where

$$\begin{aligned} u_{max}^{k,k+T_{max}} &= arg \max_{u^{k,k+T_{max}}} det(H), \\ H &= l(D_y g R D_y g^T, u^{k,k+T_{max}}, Q^{k,k+T_{max}}). \end{aligned}$$

After $k + T_{max}$, the uncertainty will drop to at least $D_y g R D_y g^T$ once again. So we can conclude that the maximum uncertainty the i^{th} feature will have at any given time is given by Equation (8.19).

Now, if we do not consider the problem relative do $\{V\}$ but to $\{W\}$, means that the vehicle has some uncertainty associated with its pose. Then, whenever the vehicle observes a feature i, it will update its own pose and also the feature position based on the observation, and will correlate its new pose with feature i new position. If the vehicle is already correlated with another features, then these features now will also be correlated with feature i, populating the covariance matrix P, just like in the regular SLAM algorithm with static features. But, unlike the SLAM with static features, the correlations will not increase monotonically, as the features move around the environment with some noise q_{fi}. So, knowing the exact position of a feature does not mean we can compute the position of any other feature with absolute certainty, since they may have moved since the last observation with some uncertainty associated with this motion. However, as demonstrated above, when considered relative to the vehicles frame $\{V\}$, this uncertainty has an upper bound. So, we can guarantee a certain quality of the relative map constructed by the vehicle. Globally speaking, although the error in the pose estimation of the vehicle and features will not be bounded, it can still be greatly reduced by the SLAM algorithm with the moving features, as shown by simulations with the CEKOLM algorithm in section 7.1.

8.3 Range Partial Observability and KF Convergence

The exteroceptive sensor used throughout this work is a range-only sensor, which gives the distance between the vehicle and a feature (or, in our case, between two

vehicles). A problem with this kind of sensor is that it gives only a partial observability, in the sense that a single measurement does not contain enough information to determine the location of the vehicle. We already discussed in Section 3.2 that, if the initial position of the vehicle is unknown, by fusing several observations it is possible to estimate this initial position. Once we have an initial estimate of the position of the vehicle we can use the EKF equations to update its position.

However, there is still one problem that might arise with a range-only sensor. It is possible to have the system in a consistent state, that is, true vehicle position inside the error ellipse of the estimated vehicle position (by error ellipse we mean the ellipse with 3σ axis that has a 99.7% probability that the true vehicle position is inside this region) and, after an observation, the updated error ellipses does not contain the true vehicle position anymore. This would give us a false confidence level about the estimated position of the vehicle. In Figure 8.10, we show such a case, where before the observation we have the right confidence level about the estimated position of the vehicle, and, after the observation, we are overconfident about its estimated position. Notice that there is a 0.3% chance that this might happen in a consistent Kalman filter. However, here, we present some cases that, if not avoided, will always lead the system to an overconfident state.

This kind of error might happen whenever the feature is situated along a line r which is equidistant between the estimated position and true position of the vehicle, or next to this line. In the example of Figure 8.11, the error will happen if the feature is situated somewhere such that its x coordinate is in between the true and estimated vehicle position, that is, $x_i > \hat{x}_v > x_v$ or $x_v > \hat{x}_v > x_i$.

To see this, consider the Kalman filter equations presented in Section 2.1.2 and repeated here:

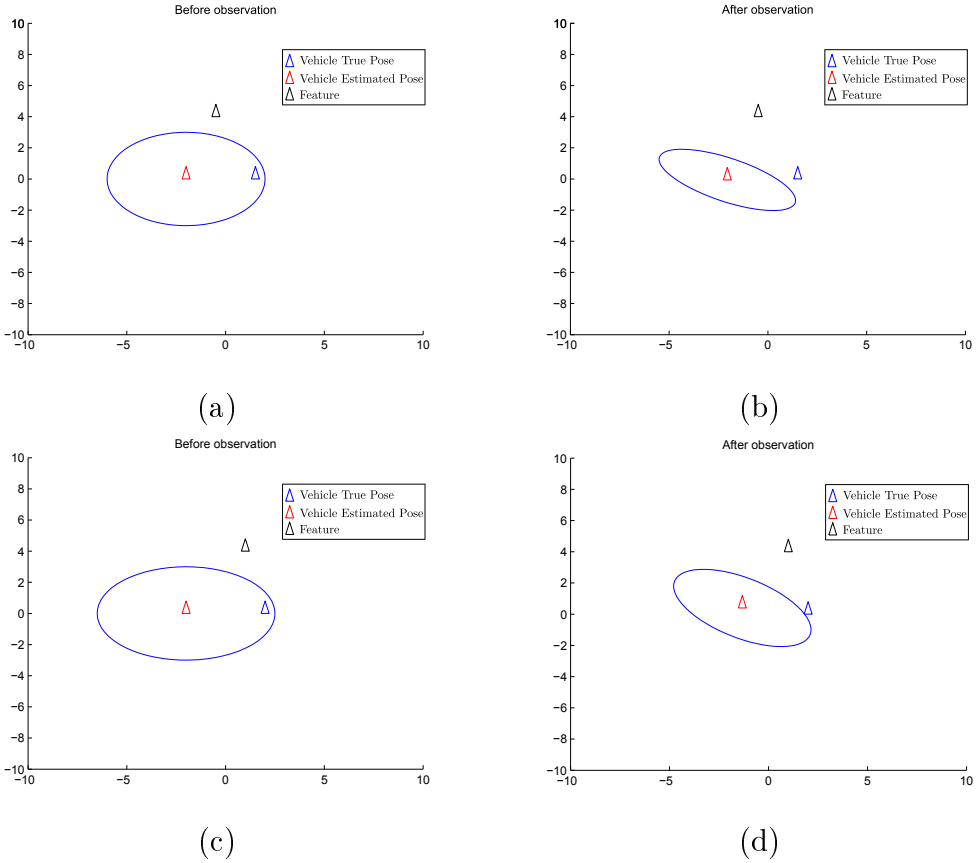


Figure 8.10: Problem with updates using partial observation: On the left (a), we can see the estimated pose of a first vehicle (red), its true position (blue), which is inside the error ellipses, and a second vehicle (black) , playing the role of a feature (it is certain about its position), which is equidistant between the true and estimated vehicle position. On the right (b), after a range-only measurement between the vehicles, the true position of the first vehicle is not inside the error ellipses centered at the new estimated position anymore. In (c) and (d) we see the same effect, but a little reduced, since the feature is not equidistant anymore (positioned at $(1, 4)$ instead of $(0, 4)$ as in (a) and (b)).

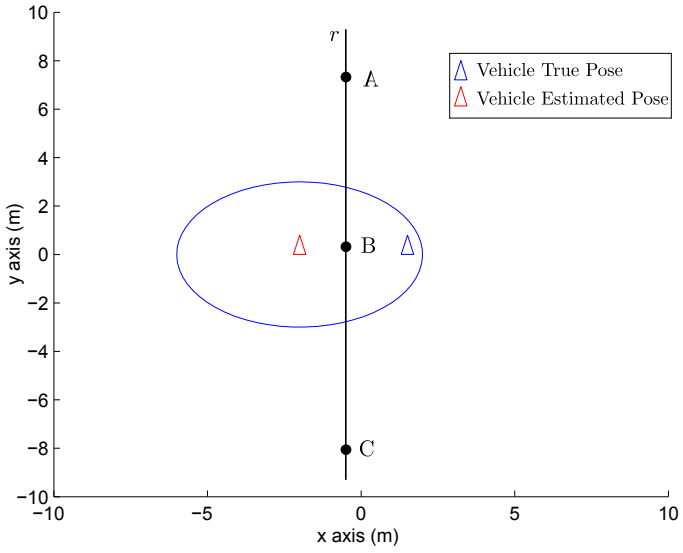


Figure 8.11: If a feature is positioned along, or next to, the line r, it will be equidistant (or almost) between true (blue) and estimated (red) vehicle position. If the vehicle observes such a feature, the error ellipses might decrease so to exclude the vehicle true position. This might happen if the error ellipses decreases along the x axis, which will happen if the feature is positioned close to the point B. If the feature move up or down, towards points A and C respectively, this effect will be reduced, as well as when the feature moves away from line r, as shown in Figure 8.13. The further the vehicle moves away from point B (when compared to the size of the uncertainty ellipses), the less the effect is observed.

$$
\begin{aligned}
e(k) &= y(k) - y_p(k), && (8.20)\\
L(k) &= D_x h P_p(k) D_x h^T + R(k).
\end{aligned}
$$

Prediciton:

$$
\begin{aligned}
\hat{x}_p(k+1) &= f(x(k), u(k), k),\\
P_p(k+1) &= D_x f P(k) D_x f^T + D_q f Q(k) D_q f^T.
\end{aligned}
$$

Update:

$$
\begin{aligned}
K(k) &= P_p(k) D_x h^T L(k)^{-1},\\
\hat{x}(k) &= \hat{x}_p(k) + K(k) e(k), && (8.21)\\
P(k) &= P_p(k) - K(k) L k(k)^T. && (8.22)
\end{aligned}
$$

Notice that, if the feature is somewhat equidistant between true and estimated vehicle position, whenever there is an observation, the estimated position of the vehicle will not change much with the update step shown in Equation (8.21), since the observation and predicted observation estimate are next to equal (remind that the measured distance is contaminated by noise), hence innovation given by Equation (8.20) $e(k) \approx 0$. However, its error ellipse will decrease, since the update in the covariance matrix shown in Equation (8.22) does not depend on $e(k)$. If it decreases enough (after one or several observations), it might not contain the true position anymore.

If the feature is not equidistant but in between true and estimated vehicle position, two situations might occur: feature closer to real vehicle position (Figure 8.12 (a)), or feature closer to estimated vehicle position (Figure 8.12 (c)). Let us analyze each case, considering one vehicle observing a feature i located at (x_i, y_i). Lets also assume, for simplicity, that $y_i = y_v$, as shown in Figure 8.12. The Jacobian of the observation function will then be given by

$$D_{X_v}h = \begin{bmatrix} \frac{x_v(k)-x_i}{d} & 0 & 0 \end{bmatrix},$$

where the position of the vehicle is given by (x_v, y_v) and d is the distance between the vehicle and the feature.

In the first case, since the measured distance between the true vehicle and the feature is probably smaller then the predicted measurement (once again, remind the noise of the measured distance), the innovation $e(k)$ will probably be negative. The Kalman gain will be given by

$$K(k) = \begin{bmatrix} \frac{d}{x_v(k)-x_i} & \frac{P_{xy}d}{P_{xx}(x_v(k)-x_i)} & \frac{P_{xy}d}{P_{xx}(x_v(k)-x_i)} \end{bmatrix}^T.$$

If $x_v < x_i$, then the x axis component of the Kalman gain will be negative, and the x component of the estimated vehicle position will increase, so that its estimated position will be corrected towards the real position. Similarly, if $x_v > x_i$, then the x axis component of the Kalman gain will be positive and the vehicle estimated position will also be corrected towards the true one.

In the second case, the innovation will probably be positive. If $x_v < x_i$ then the x axis component of the Kalman gain will be negative, so that the x coordinate of the estimated vehicle position will decrease, hence, the estimated state will move away from the true position, as depicted in Figure 8.12, which is a complete failure of the Kalman Filter. A similar failure happens if $x_v > x_i$.

Notice that, once there is an observation, the Kalman filter updates the vehicle estimated position so that the predicted measurement tends to agree with the true measurement, which, in this context, is the position in which the feature is equidistant between the true and estimated position of the vehicle. Hence, in both cases, as the vehicle takes observations, it will oscillate in the two cases while converging towards the position where the feature will be equidistant, oscillating due to the error present in the range measurement.

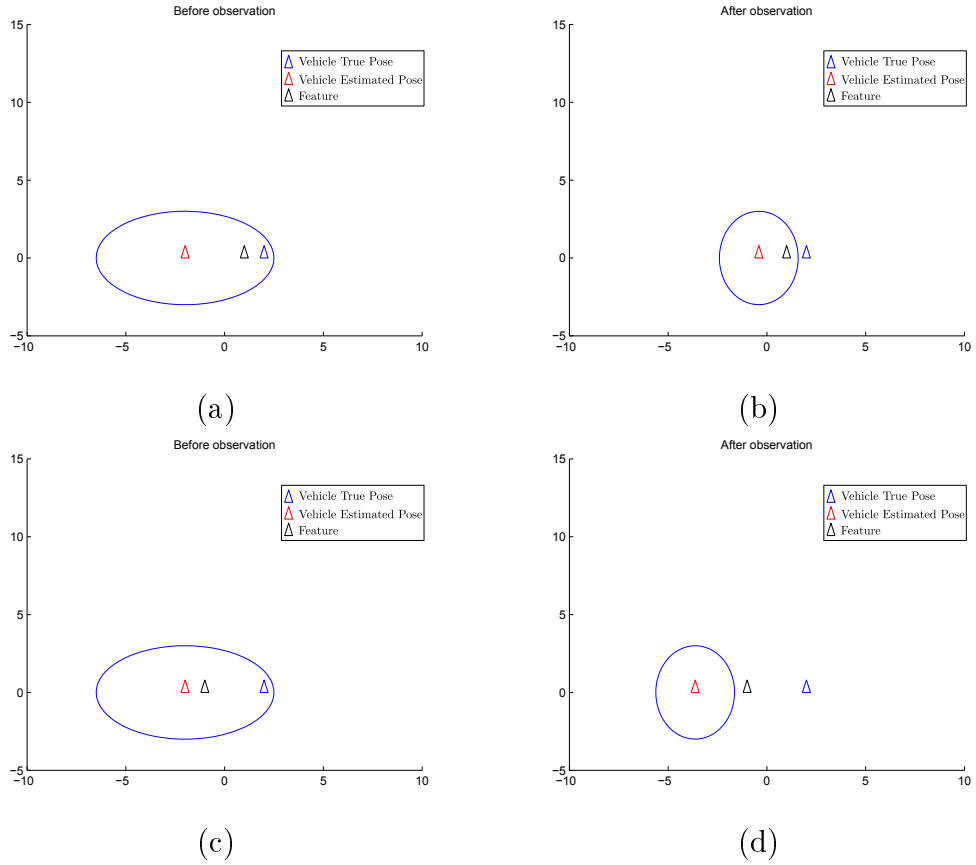


Figure 8.12: Effects of updates by observing a feature positioned somewhere in between true and estimated vehicle positions, being closer to estimated position (a) or closer to real position (c). After the update, if the feature was closer to the true position, updated estimate will move towards true position (b) (notice, however, that it is possible that the error ellipses still decreases too much). If the feature is closer to estimated position, updated estimate will move away from true position, with catastrophic results. In both cases the vehicle tends to move so that the feature is positioned at the equidistant point in between estimated and true vehicle position.

How much the error ellipse will decrease depends on how much information the observation gives. We have already shown in Section 6.1 that the error ellipse decreases along the axis joining the vehicle and the feature. So, if the feature is positioned next to the centroid of the estimated and true vehicle position, the updated state might lead to an overconfident state. This effect is reduced as the feature moves away from the centroid along the y axis (when compared with the vehicle uncertainty ellipses), as depicted in Figure 8.13. In Figure 8.13 (c), the vehicle uncertainty ellipses has a semi-minor axis of 3 m (y axis of the ellipses), and the feature is positioned at $(0, 12)$, 4 times the distance of the semi-minor axis. We can see in (d) that after the update the true vehicle position is still inside the error ellipses. Thus, one way to reduce the effect of the partial observability is to discard observations from features that are less than some distance $d_{min}(P)$ from the vehicle, where P is the vehicle uncertainty ellipse.

Since, in our work, the features are also vehicles, which have associated uncertainties, it is hard to tell if the true position of the vehicle playing the role of the feature is next to the true position of the other vehicle. So, in order to reduce the effects of this partial observability, in the algorithms presented in this work, we also do not consider any observation between two vehicles that are less then some distance $d_{min}(P_i, P_j)$ from each other, where the distance is a function of P_i and P_j, the uncertainties of both vehicles involved in the observation. As the uncertainty of each vehicle increases, so the minimum safe distance d_{min} must increase. During the simulations of the algorithms proposed in this thesis, we considered that if the two error ellipses of the vehicles involved in an observation crossed each other, the observation would be ignored.

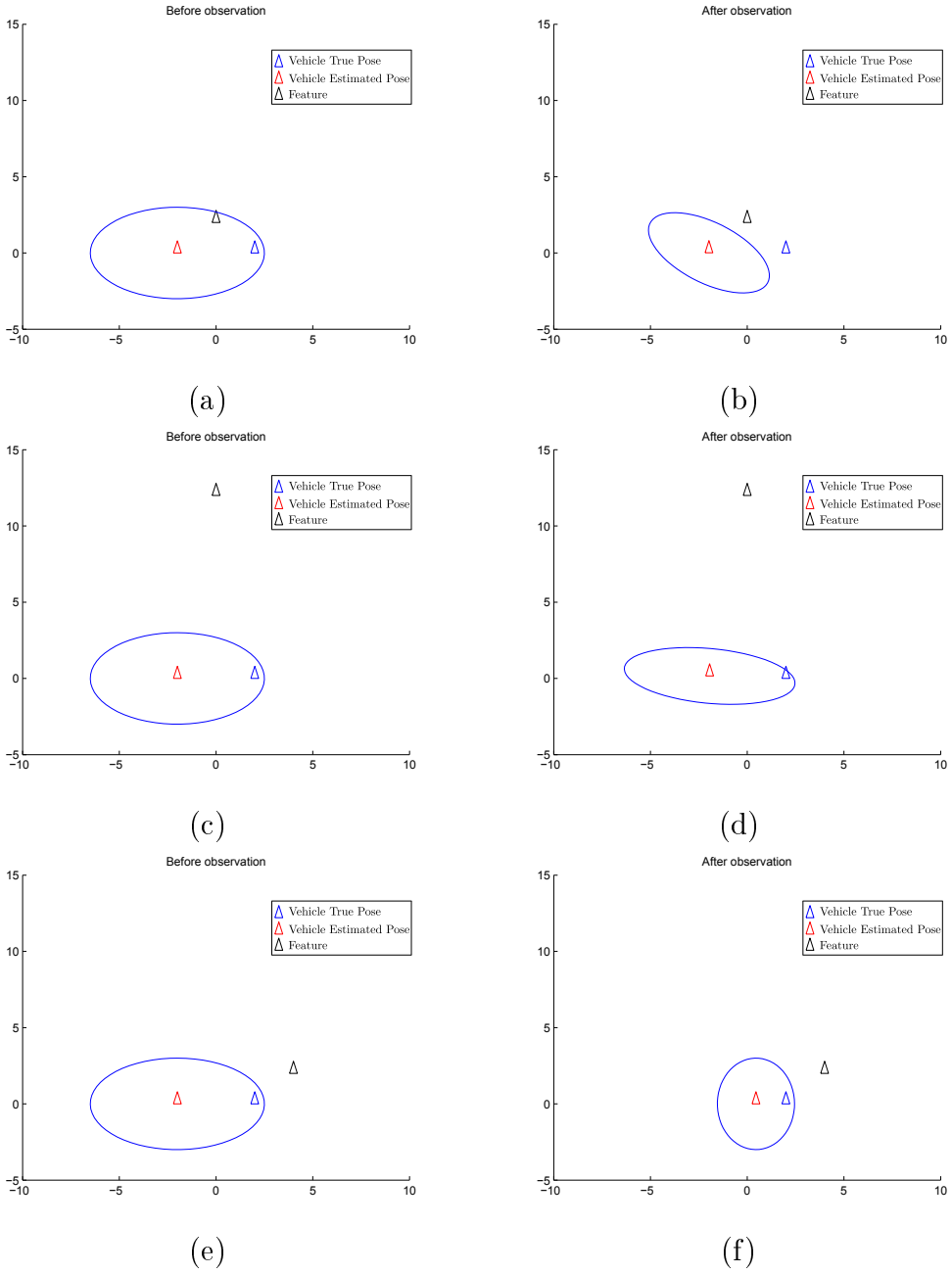


Figure 8.13: Effects of the partial observation reduced as the feature (black) moves away from the centroid, located at the origin, of estimated (red) and true (blue) vehicle position. In (a) and (b), the feature is located at $(0, 2)$, close to the origin and along the line r depicted in Figure 8.11. This leads to an overconfident state after the observation. In (c) and (d) we moved the feature up along the line r (it is now at $(0, 12)$). We can see that the error ellipses contains the true vehicle position after the update. In (e) and (f) the feature moved along the x axis, away from line r, and is at $(4, 2)$. We can see that the observation also does not lead to an overconfident estimate.

Chapter 9

Conclusions

The recent growing interest of the scientific community in exploring and studying the ocean demands the development of new technologies to support investigation in this area. The inhospitality of deep sea waters to human beings points toward the use of unmanned vehicles to perform exploration missions, which implies that the vehicle must be able to localize itself in its working area.

This thesis deals with the problem of localization for autonomous underwater vehicles. Operate unmanned vehicles in this hostile environment presents several difficulties, such as strong attenuation of electromagnetic signals, small bandwidth acoustic communications, lack of natural features, difficulties in placing and maintaining artificial features, among others. All these difficulties mean that the localization of autonomous underwater vehicles is a very hard problem.

We argued that a cooperative mission, where there is a team of vehicles operating at the same time, brings advantages in several mission scenarios. In this context, we introduced a cooperative localization scheme whereby autonomous underwater vehicles are able to take range measurements and communicate with each other. The main focus of the thesis consists in improving the estimate the pose (position and orientation) of the vehicles by using filtering techniques in a networked context, given the range measurements and communications capabilities of the team. Due to the low bandwidth, one requirement of the proposed estimation process was that

the vehicles should share as minimal as possible data with one another. Another requirement is that the algorithms should be robust to communication failures, as the acoustic underwater communication is known to be prone to packets loss and transmission delays.

With those considerations in mind, the main contribution of this thesis consists in two classes of algorithms to perform cooperative localization for underwater vehicles. They can be formulated with either the Kalman Filter (KF) - or its Extended Kalman Filter (EKF) variant - or the Particle Filter (PF). The proposed Particle Filter algorithm, called CPF, did not perform as well as did the implementation using the Kalman filtering techniques. This weaker performance was caused by the difficulties of the vehicles in sharing information. Since running the PF requires a considerable amount of data associated with a large set of particles, sharing this information using the acoustic communication was not possible. This entailed the need to compress the data to be processed and this operation revealed to be so lossy that the main advantages of the PF relatively to the KF or EKF were lost.

On the other hand, the methods based on the KF (EKF) had a good performance, and were able to greatly reduce the uncertainty about the pose of the vehicles. Since the KF (EKF) assumes a Gaussian distribution for the state vector, the amount of information to be processed is much smaller, and thus easier (faster) to share through a low bandwidth communication channel without any compression. Those methods were subdivided into two categories: the ones based on simple localization, called CEKOL algorithms, and the ones based on simultaneous localization and mapping, called CEKOLM. The CEKOL methods had an advantage of sharing less information, however, since they do not keep a cross-correlation matrix of the vehicles, they might give and overconfident estimate of the pose of the vehicles. On the contrary, The CEKOLM methods are able to create and keep an accurate cross-correlation matrix, thus giving the correct level of confidence on the estimated

pose of the vehicles. In order to maintain the associated cross-correlation matrix, CEKOLM methods must share a lot more data. However, it was possible to find a way to compress required data in a lossless way, so that the system of vehicles could use these algorithms via the available low bandwidth communication channel. As shown in simulation and experimental results, the proposed methods performed well in several different conditions and greatly improved the estimation of the pose of the vehicles in the system.

Another contribution of this thesis consists in the optimization of the estimation process of the proposed algorithms by controlling the formation of the group of vehicles or by choosing the communication protocol that they should adopt.

We also presented a scenario in which a team of vehicles that, by performing relative localization with the proposed algorithms, were able to find a minimum of a scalar field. Their cooperative activity consisted in the physical realization of the simplex downhill method. Since, in this case, the vertices are determined with uncertainties inherent to the localization method, an analysis of the convergence properties was performed for the considered context.

Through the developments of this thesis, we attempted a step forward towards the systematization of the design of multiple vehicle systems with special relevance for challenges arising in ocean observation, monitoring, surveillance, and several other applications for which cooperative underwater vehicles systems are currently regarded as a promising framework.

Obviously, a lot of research remains to be done and some research avenues can be considered as future work. The CEKOLM methods work only when the vehicles execute proprioceptive navigation (using only prediction equations of the EKF) between range observations. The authors are currently working on how the vehicles can incorporate exteroceptive information in between range observations. We also mentioned that if the application requires a large number of vehicles, several teams

of robots could be created, each group using the CEKOLM algorithm. Although this benefits each team, there will be no accurate relative position between them. We are investigating how to scale the algorithm in the number of vehicles keeping relative localization between teams of vehicles accurate.

Other interesting issues directly related with the results of this thesis should be the subject of future research. Examples of these are: the convergence behavior of SLAM schemes in the presence of map features that are non-stationary. The relationship between FAUV GPS position update and the boundedness of the error of the CEKOLM algorithm, increase the extent of decentralization by a more versatile switching of the roles played by the vehicles, and determination of communication strategies so that uncertainty is kept below a prescribed level.

For the longer term, the work developed in this thesis, provided a glimpse of challenges arising in the joint determination of control and estimation strategies. A simple example closely related with the issue of how to scale the navigation problem in order to deal with the complexity arising from a large number of vehicles, consists on how to organize the vehicles in modular teams, possibly with prescribed formations so that the performance of the overall mission is maximized. Another important issue consists in formulating and solving the general problem of optimizing the execution of missions involving multiple autonomous vehicles, for which it is understood by optimal mission performance, the one associated with a management and control of the overall resources in such a way that what is gained in improving navigation data is lost in accomplishing the overall mission goal, and, at the same time, if poorer navigation data were available then the overall system performance would decrease.

Bibliography

[1] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino. Closed loop steering of unicycle like vehicles via lyapunov techniques. *Robotics Automation Magazine, IEEE*, 2(1):27 –35, mar 1995.

[2] Peter Almström, Maben Rabi, and Mikael Johansson. Networked state estimation over a gilbert-elliot type channel. In *To appear in the 2009 IEEE CDC*, March 2009.

[3] I. Arasaratnam, S. Haykin, and R.J. Elliott. Discrete-Time nonlinear filtering algorithms using Gauss Hermite quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007.

[4] F. Arrichiello, G. Antonelli, A.P. Aguiar, and A. Pascoal. Observability metric for the relative localization of AUVs based on range and depth measurements: Theory and experiments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3166 –3171, September 2011.

[5] Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[6] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, April 2008.

[7] Joseph Djugash Athanasios Kehagias and Sanjiv Singh. Range-only slam with interpolated range data. Technical Report CMU-RI-TR-06-26, Robotics Institute, Pittsburgh, PA, May 2006.

[8] A. Bahr. *Cooperative Localization for Autonomous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2009.

[9] A. Bahr and J. Leonard. Minimizing trilateration errors in the presence of uncertain landmark positions. In *Proc. 3rd European Conference on Mobile Robots (ECMR)*, pages 48–53, 2007.

[10] A. Bahr, M.R. Walter, and J.J. Leonard. Consistent cooperative localization. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3415–3422, 2009.

[11] Alexander Bahr and John J. Leonard. Cooperative localization for autonomous underwater vehicles. In *ISER*, pages 387–395, 2006.

[12] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

[13] Tim Bailey, M. Bryson, Hua Mu, J. Vial, L. McCalman, and H. Durrant-Whyte. Decentralised cooperative localisation for heterogeneous teams of mobile robots. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2859–2865, 2011.

[14] D. Richard Blidberg. The development of autonomous underwater vehicles (auvs); a brief summary. In *ICRA - IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001.

[15] M. Bryson and S. Sukkarieh. Co-operative localisation and mapping for multiple UAVs in unknown environments. In *Aerospace Conference, 2007 IEEE*, pages 1–12, 2007.

[16] Amarjit Budhiraja, Lingji Chen, and Chihoon Lee. Nonlinear phenomena : A survey of numerical methods for nonlinear filtering problems. *Physica D*, 230(1-2):27–36, June 2007.

[17] V. A. F. Campos, D. D. S. Santana, C. M. Furukawa, and N. Maruyama. Filtro de partículas aplicados a estimação de trajetórias.

[18] Mandar Chitre. Path planning for cooperative underwater range-only navigation using a single beacon. In *Autonomous and Intelligent Systems AIS 2010 International Conference on*, pages 1–6. IEEE, 2010.

[19] Robert Christ and ScienceDirect (Online service). *The ROV manual a user guide to observation-class remotely operated vehicles.* Butterworth-Heinemann,, Amsterdam ;;Boston ;;London :, 2007.

[20] O.L.V. Costa and S. Guerra. Stationary filter for linear minimum mean square error estimator of discrete-time markovian jump systems. *Automatic Control, IEEE Transactions on*, 47(8):1351–1356, 2002.

[21] John J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Publishing Company, 1986.

[22] D. Crisan and A. Doucet. Convergence of sequential monte carlo methods, 2000.

[23] N. Cruz, A. Matos, J. Borges de Sousa, F. Lobo Pereira, J. Estrela Silva, J. Coimbra, and E. Brogueira Dias. Operations with multiple autonomous underwater vehicles: The piscis project. In *Second Annual Symposium on Autonomous Intelligent Networks and Systems*, Palo Alto, Menlo Park, CA, EUA, 2003.

[24] Jorge Estrela da Silva, Bruno Terra, Ricardo Martins, and Joao Borges de Sousa. Modeling and simulation of the lauv autonomous underwater vehicle. In *MMAR*, 2007 2007.

[25] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

[26] Arnaud Doucet, Nando de Freitas, Neil Gordon, and A. Smith. *Sequential Monte Carlo Methods in Practice*. Springer, 1 edition, June 2001.

[27] H. Durrant-Whyte. Uncertain geometry in robotics. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 851–856, 1987.

[28] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[29] H. Durrant-Whyte, D. Rye, and E. Nebot. Directed sonar navigation. In *Robotics Research: The 7th International Symposium ISRR 95*, pages 613–625, 1996.

[30] J. A. Fax. *Optimal and cooperative control of vehicle formations*. PhD thesis, Control Dynamical Syst., California Inst. Technol., Pasadena, CA, October 2001.

[31] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *Automatic Control, IEEE Transactions on*, 49(9):1465–1476, 2004.

[32] Dieter Fox, Henry Hexmoor, and Maja Mataric. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29—53, 1998.

[33] Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2000.

[34] G. Grenon, P.E. An, S.M. Smith, and A.J. Healey. Enhancement of the inertial navigation system for the morpheus autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 26(4):548–560, October 2001.

[35] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering : Theory and Practice Using MATLAB*. Wiley-Interscience, 2 edition, 2001.

[36] J. Guivant and E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2731–2736, 2002.

[37] Jose Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242—257, 2001.

[38] V. Gupta, A.F. Dana, J.P. Hespanha, R.M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *Automatic Control, IEEE Transactions on*, 54(8):1807–1819, 2009.

[39] Wenyan Hu, Tom Downs, Gordon Wyeth, Michael Milford, and David Prasser. A modified particle filter for simultaneous robot localization and landmark tracking in an indoor environment. *In Proceedings of the Australasian Conference on Robotics & Automation*, 2004.

[40] Orhan C. Imer. *Optimal Estimation and Control Under Communication Network Constraints*. University of Illinois at Urbana-Champaign, 2005.

[41] Simon J Julier and Jeffrey K Uhlmann. A new extension of the kalman filter to nonlinear systems. 3068:182—193, 1997.

[42] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[43] Tohru Katayama. *Subspace Methods for System Identification*. Springer, 1 edition, October 2005.

[44] Andreas Klöckner. Nonlinear filtering using particles and quadrature, 2007. Seminar with Boris Rozovsky.

[45] R. Kurazume, S. Hirose, S. Nagata, and N. Sashida. Study on cooperative positioning system (basic principle and measurement experiment). In *Robotics*

and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 2, pages 1421–1426 vol.2, 1996.

[46] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9:112–147, May 1998. ACM ID: 589108.

[47] Kwang Wee Lee, W.S. Wijesoma, and I.G. Javier. On the observability and observability analysis of SLAM. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3569–3574, 2006.

[48] J. Leonard and H. Durrant-Whyte. Directed sonar navigation. In *Kluwer Academic Press*, 1992.

[49] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, 1991.

[50] Hao Li and F. Nashashibi. Cooperative multi-vehicle localization using split covariance intersection filter. In *2012 IEEE Intelligent Vehicles Symposium (IV)*, pages 211–216, 2012.

[51] Xiangheng Liu and A. Goldsmith. Kalman filtering with partial observation losses. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 4180–4186 Vol.4, 2004.

[52] Zhenzhang Liu and Tien-Fu Lu. Multiple robots plume-tracing in open space obstructed environments. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 2433 –2439, dec. 2009.

[53] A. Martinelli and R. Siegwart. Observability analysis for mobile robot localization. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)*, pages 1471 – 1476, August 2005.

[54] I. Matei, N. Martins, and J.S. Baras. Optimal state estimation for discrete-time markovian jump linear systems, in the presence of delayed output observations. In *Information Theory Workshop, 2008. ITW '08. IEEE*, pages 237–242, 2008.

[55] PS Maybeck and Academic Press. *Stochastic models, estimation and control. Volume I.* 1979.

[56] Qing-Hao Meng, Wei-Xing Yang, Yang Wang, Fei Li, and Ming Zeng. Adapting an ant colony metaphor for multi-robot chemical plume tracing. *Sensors (Basel, Switzerland)*, 12(4):4737–4763, April 2012. PMID: 22666056 PMCID: PMC3355438.

[57] Mehran Mesbahi and Magnus Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, July 2010.

[58] Michael, Ling, Abhishek, and Richard M. Probabilistic performance of state estimation across a lossy network. *Automatica*, 44(12):3046–3053, December 2008.

[59] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593—598, 2002.

[60] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *In Proceedings of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151—1156, 2003.

[61] Anastasios I. Mourikis and Stergios I. Roumeliotis. Performance bounds for cooperative simultaneous localization and mapping (c-slam). In *Robotics: Science and Systems*, pages 73–80, 2005.

[62] S. Munir and W.J. Book. Internet-based teleoperation using wave variables with prediction. *IEEE/ASME Transactions on Mechatronics*, 7(2):124–133, 2002.

[63] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890–897, 2001.

[64] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308 –313, January 1965.

[65] P. Newman and J. Leonard. Pure range-only sub-sea SLAM. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1921–1926 vol.2, 2003.

[66] Paul Michael Newman. EKF based navigation and SLAM, July 2004. SLAM Summer School 2004, Toulouse.

[67] Juan Nieto, Tim Bailey, and Eduardo Mario Nebot. Scan-slam: Combining ekf-slam and scan correlation. In *FSR'05*.

[68] Johan Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Dept. Automatic Control, Lund Inst. Technol., Lund, Sweden, 1998.

[69] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 5 edition, September 2009.

[70] Edwin Olson, John J. Leonard, and Seth Teller. Robust Range-Only beacon localization. *Oceanic Engineering, IEEE Journal of*, 31(4):949–958, 2006.

[71] F. Lobo Pereira. Contributions of optimal control for a vision in decision and control engineering, 1997.

[72] Fernando Lobo Pereira and João Borges Sousa. Coordinated control of networked systems. *Automation and Remote Control*, Vol 65:pp. 1037 – 1045, 07/2002 2002.

[73] J.M. Porta. CuikSLAM: a kinematics-based approach to SLAM. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2425–2431, 2005.

[74] Jim Primbs. *Survey of Nonlinear Observer Design Techniques*. 1996.

[75] Ioannis M. Rekleitis. *Cooperative Localization and Multi-Robot Exploration*. PhD thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, February 2003.

[76] W.D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 2192–2197 vol.3, 1993.

[77] S.I. Roumeliotis and G.A. Bekey. Distributed multirobot localization. *Robotics and Automation, IEEE Transactions on*, 18(5):781–795, 2002.

[78] Stergios I Roumeliotis and Ioannis M Rekleitis. Propagation of uncertainty in cooperative multirobot localization: Analysis and experimental results. *Autonomous Robots*, 17:41–54, 2004. 10.1023/B:AURO.0000032937.98087.91.

[79] Rajnikant Sharma and Clark Taylor. Cooperative navigation of MAVs in GPS denied areas. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 481–486, 2008.

[80] Ling Shi, K.H. Johansson, and R.M. Murray. Change sensor topology when needed: How to efficiently use system resources in control and estimation over wireless networks. In *Decision and Control, 2007 46th IEEE Conference on*, pages 5478–5485, 2007.

[81] R. Sim. Stabilizing information-driven exploration for bearings-only SLAM using range gating. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3396–3401, 2005.

[82] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *Automatic Control, IEEE Transactions on*, 49(9):1453–1464, 2004.

[83] R Smith, M Self, and P Cheeseman. Estimating uncertain spatial relationships in robotics. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 850, 850, 1987.

[84] S.M. Smith, K. Ganesan, P.E. An, and S.E. Dunn. Strategies for simultaneous multiple auv operation and control. *International Journal of Systems Science*, 29(10):1045–1063, 1999.

[85] João Borges Sousa, K.H. Johansson, J. Silva, and Alberto Speranzon. A verified hierarchical control architecture for coordinated multi-vehicle operations. *International Journal of Adaptive Control and Signal Processing*, Vol 21(Issue 2-3):pp. 159 – 188, 2006.

[86] D. J Spero and R. A Jarvis. A new solution to the simultaneous localisation and map building (SLAM) problem. February 2008.

[87] Milos S. Stankovic and Dusan M. Stipanovic. Discrete time extremum seeking by autonomous vehicles in a stochastic environment. In *To appear in the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, P.R. China, December 2009.

[88] Srdjan S. Stankovic, Milos S. Stankovic, and Dusan M. Stipanovic. Consensus based overlapping decentralized estimation with missing observations and communication faults. *Automatica*, 45(6):1397–1406, 2009.

[89] S.S. Stankovic and D.M. Stipanovic. Consensus based overlapping decentralized estimator. *Automatic Control, IEEE Transactions on*, 54(2):410–415, 2009.

[90] Curtin T., J. G. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling network. *Oceanography*, 6(3):83–94, 1993.

[91] Y. T. Tan and M. Chitre. Single beacon cooperative path planning using cross-entropy method. In *IEEE/MTS Oceans'11 Conference*, Hawaii, USA, September 2011.

[92] Sekhar Tatikonda and Sanjoy Mitter. Control under communication constraints. Technical report, 2004.

[93] P.V. Teixeira, M.B. Nogueira, and J.B. Sousa. Long baseline beacon position estimation. In *OCEANS, 2011 IEEE - Spain*, pages 1 –7, June 2011.

[94] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, September 2005.

[95] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[96] Yu Tian and Aiqun Zhang. Simulation environment and guidance system for auv tracing chemical plume in 3-dimensions. In *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, volume 1, pages 407 –411, march 2010.

[97] Y. Tipsuwan and Mo-Yuen Chow. Gain scheduling middleware for networked mobile robot control. In *American Control Conference, 2004. Proceedings of the 2004*, volume 5, pages 4313–4318 vol.5, 2004.

[98] J. Vaganay, J.J. Leonard, J.A. Curcio, and J.S. Willcox. Experimental validation of the moving long base-line navigation concept. In *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pages 59–65, 2004.

[99] M. Walter and J. Leonard. An experimental investigation of cooperative SLAM. In *Proceedings of the Fifth IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[100] Chieh-Chih Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results

from a ground vehicle in crowded urban areas. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 842–849 vol.1, 2003.

[101] Y. Q. Wang, H. Ye, and G. Z. Wang. Fault detection of NCS based on eigendecomposition, adaptive evaluation and adaptive threshold. *International Journal of Control*, 80(12):1903–1911, 2007.

[102] Sarah E. Webster, Ryan M. Eustice, Christopher Murphy, Hanumant Singh, and Louis L. Whitcomb. Toward a platform-independent acoustic communications and navigation system for underwater vehicles. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, Biloxi, MS, 2009. Accepted, To Appear.

[103] Peter Whaite and Frank P Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 19:339—346, 1995.

[104] Wikipedia. Map - wikipedia, the free encyclopedia, 2009. [Online; accessed 24-November-2010].

[105] Wikipedia. Monte carlo method — Wikipedia, the free encyclopedia, 2009. [Online; accessed 27-May-2009].

[106] Wikipedia. Ocean — Wikipedia, the free encyclopedia, 2009. [Online; accessed 21-May-2009].

[107] S.B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, University of Sydney, Australian Centre for Field Robotics, Sydney, Australia, 2001.

[108] Wing Shing Wong and R.W. Brockett. Systems with finite communication bandwidth constraints. I. state estimation problems. *Automatic Control, IEEE Transactions on*, 42(9):1294–1299, 1997.

[109] Wing Shing Wong and R.W. Brockett. Systems with finite communication bandwidth constraints. II. stabilization with limited information feedback. *Automatic Control, IEEE Transactions on*, 44(5):1049–1053, 1999.

[110] Junku Yuh, Tamaki Ura, and George A. Bekey, editors. *Underwater Robots*. Springer, July 1996.

[111] D. Zarzhitsky, D. Spears, and W. Spears. Distributed robotics approach to chemical plume tracing. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 3415–3422, 2005.

[112] V.A. Ziparo, A. Kleiner, L. Marchetti, A. Farinelli, and D. Nardi. Cooperative exploration for USAR robots with indirect communication. In *Proc.of 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV'07)*, 2007.