Safe localization and control of a towed sensor

Joris Tillet

DOCTORAT BRETAGNE LOIRE MATHSTIC

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Robotique*

Par

Joris TILLET

Safe Localization and Control of a Towed Sensor

Localisation et Contrôle Sûrs d'un Capteur Tracté

Thèse présentée et soutenue à Brest, le 18 octobre 2021
Unité de recherche : Lab-STICC

Rapporteurs avant soutenance :

Lionel LAPIERRE — Maître de conférence, LIRMM, Montpellier, FR
Martine CEBERIO — Professeure, The University of Texas, El Paso, USA

Composition du Jury :

Président :	Reda BOUKEZZOULA	Professeur, Polytech Annecy-Chambéry, Annecy Le Vieux, FR
Examinateur :	Andreas RAUH	Professeur, Carl von Ossietzky Universität, Oldenburg, DE
Dir. de thèse :	Luc JAULIN	Professeur, Lab-STICC, ENSTA Bretagne, Brest, FR
Co-dir. de thèse :	Fabrice LE BARS	Maître de conférence, Lab-STICC, ENSTA Bretagne, Brest, FR

Invité(s) :

Jean-Daniel MASSON — Expert, Agence de l'Innovation de Défense, DGA, Paris, FR
Marie-Véronique SERFATY — Directrice, Agence de l'Innovation de Défense, DGA, Paris, FR

Safe Localization and Control of a Towed Sensor

Joris TILLET

Brest, December 15, 2021

À ma famille.

"*Knowledge, like air, is vital to life. Like air, no one should be denied it.*"

— Alan Moore, V for Vendetta

Acknowledgments

Quelle satisfaction d'écrire aujourd'hui ces lignes, et ainsi apporter la touche finale à ce manuscrit qui marque l'aboutissement de mon doctorat. C'est avec beaucoup de plaisir que je repense à tout ce chemin que j'ai parcouru ces trois dernières années. Bien sûr, il y a eu des moments plus difficiles, je me revois notamment pendant la rédaction, quand je rêvais justement de rédiger ce chapitre. Mais je suis fier d'être arrivé au bout, et donc d'écrire aujourd'hui ma gratitude envers toutes ces personnes sans qui rien de tout cela n'aurait pu se réaliser.

Pour commencer, je tiens à remercier mes encadrants de thèse, puisque ce sont eux qui ont logiquement tout rendu possible. Merci donc à Luc JAULIN, mon directeur de thèse, pour ses idées abondantes et variées, sa grande disponibilité, sa compréhension et son enthousiasme. Merci également à Fabrice LE BARS, co-encadrant, pour ses nombreuses opinions, ses relectures minutieuses, sa grande disponibilité, et aussi pour m'avoir emmené à différents concours de robotique.

C'est avec joie que je remercie les membres de mon jury pour le temps et l'intérêt qu'ils ont témoigné pour mon travail. Merci à Lionel LAPIERRE pour sa relecture, ses corrections et ses remarques pertinentes "pour venir me chatouiller". Merci à Martine CÉBÉRIO pour sa relecture, son intérêt et sa considération pour mon travail. Merci à Réda BOUKEZZOULA pour avoir présidé le jury, pour son enthousiasme et son soutien dans nos contributions sur la théorie de l'incertain. Merci à Andreas RAUH pour avoir assumer le rôle d'examinateur, mais aussi pour les références bibliographiques ainsi que l'invitation et l'organisation des séminaires sur les méthodes par intervalles.

Je tiens naturellement à remercier Annick BILLON-COAT et Michèle HOFMANN pour leur précieuse aide dans les démarches administratives, ainsi que leur grande sympathie au quotidien. Je souhaite remercier du fond du cœur tous les collègues du bâtiment M qui cultivent cette excellente ambiance au laboratoire et qui motivent à venir travailler tous les jours dans une atmosphère attachante. J'ai vraiment vécu des moments plaisants en votre compagnie. Je repense notamment au temps passé ensemble à Guerlédan, mais aussi aux parties de *Gobb'it*, aux sorties escalade, cinéma, restaurant, soirées diverses, plage & surf, bateau, balades, escape games, fabrication de jus de pommes, ... En particulier, merci à toi, Irène MOPIN, qui m'a tant aidé et

soutenu, tout spécialement ces derniers mois. Je te suis reconnaissant pour le temps que tu m'as donné en conseils, explications, relecture, pour les multiples services rendus, mais surtout pour ton amabilité. Bien sûr, je désire remercier les personnes avec qui j'ai partagé le bureau M025 ces trois dernières années : Thomas LE MÉZO pour tes conseils variés, tes encouragements, mais par-dessus tout ta compagnie et ta gentillesse. Également Auguste BOURGOIS et Pierre NARVOR, pour votre aide, toutes nos discussions et plus généralement votre joyeuse compagnie. Je remercie les autres doctorants, docteurs et ingénieurs de recherche dont je suis heureux d'avoir croisé la route : François CÉBRON, Maël LE GALLIC, Dominique MONNET, Simon CHANU, Pierre BENET alias *P2*, Julien DAMERS, Marie PONCHART, Fabien NOVELLA, Romain SCHWAB, Simon ROHOU. Un petit mot spécial pour Yoann SOLA alias *Gaston*, pour toutes ces discussions que nous avons eu sur des sujets très variés, et tous ces moments passés sur des activités diverses, souvent "juste pour tester". Je pense également à ceux que j'ai connu plus récemment, et dont j'ai apprécié les instants passés ensemble : Nathan FOURNIOL, Benjamin LEPERS, Christophe VIEL, Alam CASTILLO, Carlos ORTIZ, Maria COSTA, Aurélie PANETIER, Quentin FERDINAND, Morgan LOUEDEC, Quentin BRATEAU. Je tiens pareillement à remercier Benoît ZERR, Rémi RIGAL, Yoan CHEVILLOTTE. Je souhaite bon courage aux futurs docteurs ! Je veux remercier le centre de ressources pour son aide précieuse et sa bonne humeur (et l'odeur de fromage fondu le vendredi après-midi !). Chers amis, je ne vous oublierai pas.

Je laisse également un mot pour mes amis de plus longue date, qui se sont intéressés de près ou de loin à mon travail de thèse, mais avec qui je passe toujours et encore des moments riches et captivants. Merci donc à Lélio, Cyril, Jolan!, Axel, Yan, Clément, Nicolas, Madian.

Je ne saurai écrire en quelques lignes toute ma gratitude pour toi, Emeline LUIRARD, qui m'a bien sûr considérablement soutenu ces trois dernières années, mais qui m'accompagne depuis maintenant 8 ans, pour mon plus grand bonheur. Je te remercie donc de tout mon cœur pour tout ce que tu fais, tous les jours, pour moi.

J'aimerai également pouvoir exprimer toute ma reconnaissance et mon affection pour mes parents et mon frère, qui m'ont soutenu, aidé, encouragé, et par-dessus tout aimé, toutes ces années. Vous êtes et resterez sans cesse un cocon de bonheur que je retrouve à chaque fois : avec vous, c'est toujours Noël ! Je ne retiens pas une pensée pour ma nièce Léane, à qui je pensais sous la dénomination de *Pamplemousse* pendant ma rédaction, et qui a assisté à ma soutenance alors qu'elle n'avait pas encore 3 mois. Ne t'en fais pas Léane si tu n'as pas tout compris, je pourrai te réexpliquer en temps voulu ! J'étends naturellement ma reconnaissance à ma famille au sens large, avec notamment la *tribu* que j'ai plaisir à retrouver chaque année à Chanteloube ou ailleurs, ainsi que la famille TILLET que j'affectionne et dont j'apprécie toujours son sens de l'accueil.

Pour finir, je remercie le lecteur qui a ouvert ce manuscrit, j'espère qu'il saura prendre du plaisir à lire ces pages comme j'ai pu en prendre durant ces trois années, et que sa curiosité pour mon travail soit ratifiée par quelques pensées propices. Bonne lecture !

Joris.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

The exploration of the oceans becomes more and more reachable thanks to still more performing robots. Research in underwater robotics has already made possible the development of efficient unmanned underwater vehicles. The purpose of this thesis comes within the scope of this still active research field. Mainly, two different problems are studied to deal with underwater exploration.

Firstly, the control of a deported vehicle allows acquiring accurate data from quickly perturbable sensors. As an example, a magnetometer is a sensor that measures the local magnetic field. By using such a sensor, it is possible to build a magnetic map of the explored area, allowing to spot out submarine communications cable, some wrecks, or geological anomalies. It requires to be deported from the main vehicle in order to avoid perturbations from the engines. Thus the strategy consists in towing the sensor while controlling its trajectory.

Secondly, the reliable localization of an underwater system is crucial for exploiting the gathered information. Indeed, when scanning an area, it is necessary to have enough accurate localization to be sure the scan is complete. In addition, the localization allows placing the made measurements on a map to be sure one can find attractive targets back.

Finally, among the profuse mysteries to discover in the seas, wrecks hold parts of the unknown past of our ancestors. Therefore, searching wrecks is the primary motivation of the presented works.

1.1 Underwater archaeology

1.1.1 Humankind history through shipwrecks

Writing was invented for the first time between 3400 and 3100 BCE in Mesopotamia. However, it had also been developed independently in three other ancient civilizations: in Egypt, around 3250 BCE [129], in China, around 1200 BCE [13] and in lowland areas of Southern Mexico and Guatemala, around 500 BCE [52]. Before this time, it is prehistory. Therefore, there are no texts to narrate the development of humankind during this era, except for rare undeciphered symbols. Even after this invention, writings are scarce in many historical periods. Thus, there is still much unknown, and the only means to learn more about these ancient times is to find time-linked artifacts. That is the role of archaeology.

Archaeology is a discipline that requires various skills: history knowledge, exploration, chemistry aptitudes, to name a few. That is why it has been divided into several sub-disciplines whose focus can be, for instance, a specific material, a particular period, or a geographical place. Some of these sub-disciplines characterized by a thematic concern became academic disciplines in the 1970s, as maritime and nautical archaeology [20]. We can mention the formation of the NAS (Nautical Archaeology Society) in 1972 which published a guide to underwater archaeology in 1992 [34]. In underwater environments, we can find remains of boats, ships, and other vessels as expected. There are also wrecks of aircraft and many other objects or buildings, either linked

with the water or lost offshore. Plenty of such sites of interest have already been discovered such as ancient settlements, harbor vestiges, or warships from different ages [35, 138, 101].

An example of a model of an underwater shipwreck site located in southern Brazil is given in Figure 1.1. This image has been obtained using photogrammetry techniques. It consists of taking several pictures of a site and merging all these pictures into a unique figure, which is in three dimensions.

Figure 1.1: Photogrammetric model of the Lagoa do Peixe. Credit: R. Torres [166].

The discoveries of these ancient vestiges often help to bring answers to the lifestyle of our ancestries. However, finding a wreck or other remains underwater is rare, difficult, and often very expensive. Robots are already used for this task and have several advantages: they reduce human life risks, they can go in deeper water, they can work for several days autonomously, and they can be remotely operated for sensitive tasks. An example of an Autonomous Underwater Vehicle (AUV) used for deep-water archaeology is given in [12]. Another project with several robots for archaeology purposes can be found in [2].

The interests of robotics in underwater archaeology are twofold. On the one hand, robots are extremely useful in this unknown and dangerous environment as they can work longer and deeper than divers. On the other hand, the tasks of archaeological surveys raise robotic challenges whose solutions can have applications in multiple fields.

This thesis is incorporated within the framework of the exploration and search of the location of such archaeological sites. Most of the time, this task consists in scanning an area where we assume that there are objects of archaeological interest. The most commonly used sensor is the side-scan sonar, either towed by a boat or embedded on an underwater robot. It can give an acoustic image of the seabed with enough precision to spot out wrecks or other artifacts [138, Chap. 39]. An example of a wreck perceived by a side-scan sonar is given in Figure 1.2.

However, this sensor is insufficient when relics are covered with sediments or even wholly buried under sand or mud. The reason is that their acoustic frequency is too high (hundreds of kiloHertz) to get a great image resolution and consequently does not enter sediment sub-layers

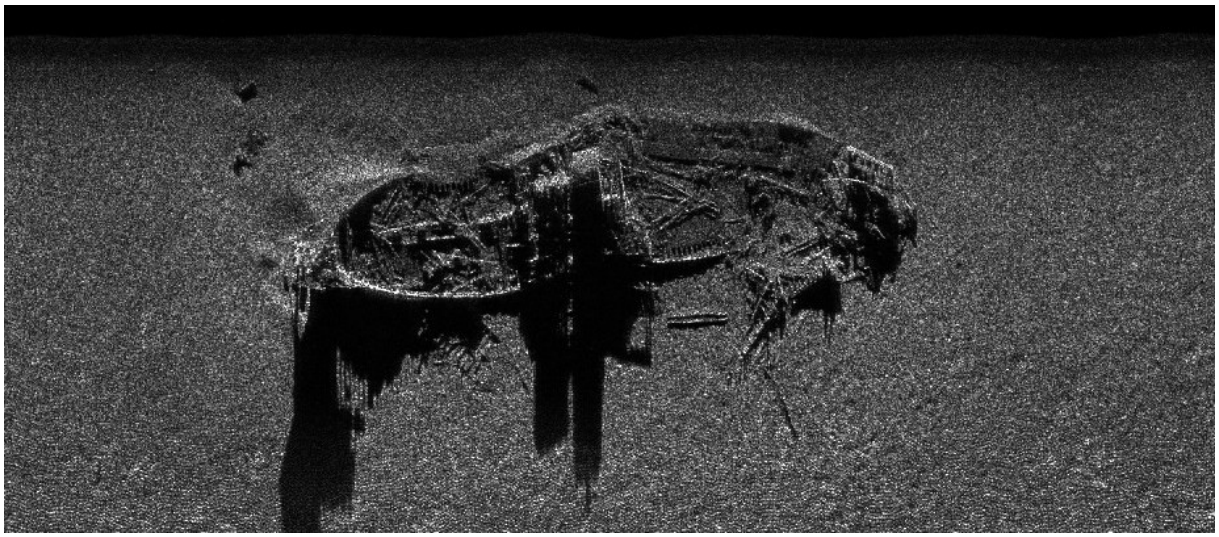


Figure 1.2: The *Swansea Vale* wreck perceived by the Klein 5400, a side-scan sonar, in high-resolution mode. Credit: DGA-TN Brest.

[105]. In that case, archaeologists must carry out subsurface surveys, which consist in probing the sediment sub-layers. They can use a sub-bottom profiler or a magnetometer when there are ferromagnetic materials. The former is a sounder that penetrates below the seabed surface by using low frequency, mainly used to examine sedimentary layers. Thus it can detect buried objects [58]. However, it seems that this sensor had never genuinely led to the discovery of a buried wreck [62, Chap. 3]. A magnetometer is a sensor that measures the intensity or the direction of a magnetic field. So it can detect the perturbation due to an iron shipwreck or any sufficiently ferromagnetic objects like anchors or canons. It has already been used for finding wrecks, firstly by treasure hunters, and then during archaeological surveys as in [3], once it has been shown to be efficient for this task [66]. The interpretation of the marine magnetic data to distinguish shipwrecks and debris is not an easy task [20, Chap. 4]. A magnetometer is a highly sensitive sensor and can detect plenty of local anomalies in some areas, with a low proportion of artifacts of interest. Finally, this sensor has a low resolution, and it is not easy to locate the detected targets with respect to the magnetometer.

The method of the survey itself comes with a bundle of issues. Indeed, the carrier of the magnetometer can highly perturb its measurements. It is typically due to the ship's motors. So the sensor is generally towed behind its carrier, at a rough estimate of 10 meters behind minimum. The intensity of a magnetic perturbation decreases with the cube of the distance. It is thus usually desirable to drag the sensor the closest to the seabed as possible. However, it is also hard to know precisely its actual depth, neither the local bathymetry.

1.1.2 *Marie-la-Cordelière*

This thesis comes within the scope of the search for the *Marie-la-Cordelière*, or simply *Cordelière*. The *Cordelière* was a Breton (French) ship of 600 tons built between 1494 and 1496. It was

armed with 200 canons and could embark on until 1000 people. It sunk in the Bay of Brest, West of France, on the 10th, August 1512, during a battle against English ships (see Figure 1.3). Archaeological researches have been carried out to try to find the ship's wreck in 1997 and 2001. New research campaigns have been conducted in 2018 and 2019 by the French *Département des Recherches Archéologiques Subaquatiques et Sous-Marines* (Department of underwater archaeological research) (DRASSM). The difficulty of this objective comes from several factors: the search area is huge (11×6 km), and the wreck is probably highly deteriorated and covered by sediments [63]. Thus, acoustic methods do not seem to be suitable for such a buried wreck. Only the magnetic research has been envisioned [64].

Figure 1.3: *Le combat de la Cordelière* by Pierre-Julien Gilbert in 1838. Painting of the battle of the *Cordelière* against an English fleet, in 1512, off the coast of Saint-Mathieu.

1.2 Robotic challenges

Searching wrecks is a challenging task for robotics. Indeed, the final goal is to scan a whole underwater area with the dedicated sensor and locate the detected targets. For instance, with the magnetometer, the objective is to build a magnetic map of the search area. It raises issues linked to sea exploration. The two following main challenges can be spotted out and are studied through this thesis:

1. Design a controller to make the sensor perform a proper scan safely;

2. Localize the robot during the scan to situate the measurements.

These challenges are rather complementary. We detail them more precisely in the next section, with special care on robotics-related issues.

1.2.1 Navigation and control

Problem statement

In the scope of the search for the *Cordelière*, only the magnetic survey is considered. Nevertheless, a magnetometer has to be deported to ensure not to interfere with measurements. Since then, the task becomes more complex: the robot should follow a trajectory such that the deported sensor follows the desired scanning pattern.

Thus, the robotic problem involves designing a non-linear controller and a navigation algorithm for a robot towing a deported system. The controller can only act on the robot as the sensor is not actuated. However, it is the trajectory of the deported system that matters: the sensor must scan a complete area. Even if the robot reaches a perfect scan, there is no guarantee that the towed system also reaches the perfect scan. Indeed, when acting on the robot, the effects on the deported system's dynamics come with a slight delay. It can be compared to the behavior of a trailer towed by a car. For instance, the trailer will cut corners. Therefore, the controller should anticipate this behavior to make the sensor do a proper scan.

Ensuring the accuracy and the safety of the navigation

In addition, we want some guarantee on the actual probed area and the safety of the system. Indeed, we want to be sure that the whole area has been scanned and that any potential target cannot have be missed. Besides, the volume taken by the system being significant, the risks of collision are higher. Thus we also need some guarantee on that issue.

These guarantees should be checked before starting a mission. They can take the form of constraints to be satisfied all through the mission. Indeed, it would be regrettable to send the robot and realize afterward that the scan of the area contains gaps. Even worse, we want to avoid losing it because of a collision, even though we could have predicted the crash. Therefore, some constraints are defined to translate the guarantees of the scan and the safety issues.

1.2.2 The underwater localization problem

Lack of information

Electromagnetic radiation constitutes the primary support for information-carrying in air. In particular, it includes visible light and radio waves which can efficiently serve to provide data on the environment and communicating between remote systems.

However, these electromagnetic waves are absorbed by water, avoiding their propagation underwater. Thus, it makes the use of a whole class of sensors unsuitable for underwater robots. They are deprived of Global Navigation Satellite System (GNSS), which can provide precious

information on localization. Note that the unavailability of GNSS has also to be handled by ground and aerial robots in indoor environments [57, 70]. Nevertheless, indoor robots still have access to light and radio sensors to gather enough information for accurate localization. The underwater localization problem is much more complex. Depending on the turbidity of the water, light can travel up to several tens of meters, but in most places, it is not more than a few meters. Additionally, colors are altered depending on the distance, see *e.g.* [6].

In order to obtain an estimation of speed and orientation, the dynamical model of the robot may be used, making the state of the robot estimated over time with respect to the inputs sent to the actuators. Yet, this approach is generally not efficient at all, as the dynamical model is an approximation that introduces errors, and the external perturbations are not taken into account (especially the currents).

Dead reckoning approach

Another approach consists in measuring the different forces acting on the robot. Then these forces are integrated to obtain an estimation of a position change between two measurements. This method is called *dead reckoning*. In this scope, many proprioceptive sensors exist, with varying quality and price depending on the embedded technology. Accelerometers and gyroscopes are inertial sensors that measure linear acceleration and rate of rotation, respectively. Magnetic sensors return the north direction but are sensitive to local perturbations, either due to the robot itself or any magnetized elements in the environment. These three types of sensors, accelerometers, gyroscopes, and magnetometers (or gyrocompass), are usually combined to obtain an estimation of the 3D orientation of the system. A non-linear estimation algorithm, which is often an Extended Kalman Filter (EKF), is also run on these data to compute the orientation. This whole system is called an Attitude and Heading Reference System (AHRS).

The fusion of sensor data allows for minimizing the errors, delaying when cumulated errors are too significant. Complete systems computing the estimations of the position, the velocity, and the orientation constitute Inertial Navigation Systems (INSs) and include, besides the same sensors as for an AHRS, some electronics, and a navigation computer.

Another sensor is also commonly used to help to limit the growth of the cumulative errors: Doppler Velocity Loggers (DVLs) can directly give an estimation of the speed over the ground of the vehicle, using acoustic waves and the Doppler effect. When data from a DVL are combined with an INS, we talk about Aided Inertial Navigation System (AINS). It can limit the growth of the errors to approximately 0.02% of the distance traveled, according to [120]. For more details on performances, a state-of-the-art of AUV positioning is given in [7], with an application to bathymetric surveys. Note that water speed sensors may be used instead of DVLs. However, although cheaper, they cannot measure the vehicle velocity component relative to the seabed due to the ocean currents, making them unsuitable for most underwater areas.

The use of high-quality INS is bulky and is thus not suitable for small-sized robots. In addition, it is very expensive. A DVL requires to be close enough (less than 500 m at 150 kHz

[99]) to either the seabed or the surface to get accurate measurements. So a DVL could be used in the scope of the search of wrecks. It is pretty expensive but relevant as the robot is supposed to be close enough to the seabed. Note that problems can still occur in a shallow water context due to acoustic sources of errors such as noise, multipath, or reverberation. Indeed, when the water level is low, an acoustic ray can be reflected by the seabed and the water surface several times, blurring the received data [26].

Finally, the dead reckoning approach provides a localization relative to an initial point. However, independently of the sensors' quality, this approach leads to an unbounded growing error on the estimated position. Thus, dead reckoning is relevant only for short missions or combined with another absolute localization system. An example of a dead-reckoning approach using intervals is given in [95] where some communications and ranging help to bound the errors. In [130], an Extended Kalman Filter (EKF) is used to track the position of an agricultural vehicle with the odometry when the GNSS signal is lost. An EKF is also used in [17] on an underwater vehicle, using data from a DVL. The errors stay bounded thanks to a sonar acquisition.

Finding an external reference

The unavailability of the GNSS causes the absence of absolute external reference, which can bound the error of localization over time. The first solution to this issue is to resurface from time to time, such that the access to GNSS remains partially available, making the errors bounded. However, even if it can be envisioned in shallow waters, most of the time, it is not desirable as it is energy costly and unconceivable for deep-water missions. Thus, it is possible to use acoustic beacons, either linked to the surface with an absolute reference or fixed underwater, to retrieve an alternative external reference. These beacons are localized and can communicate with the robot to give its relative range or bearing. Then the robot can calculate its position applying triangulation or trilateration. The two central used systems are Long Baseline (LBL) and Ultra Short Baseline (USBL) using several, or only one, beacon, respectively. The latter requires a transceiver on the robot that contains several receiving elements to compute the bearing of the coming information. So these systems seem to handle the issue but still come with their drawbacks: it is expensive, requires a complex setup which can cause problems, and implies either a limited area (less than 10 km^2) or a surface vessel in the case of non-deep-water missions [99]. Study and use case are given in [114] for example.

In the context of the search of a wreck such as the *Cordelière*, surface vessels should be avoided, making systems such as USBL unsuitable. In addition, the search area is too vast for systems such as LBL. Finally, to avoid the cost and the complexity of LBL or USBL systems, and also too expensive proprioceptive sensors as high-quality INS, other approaches should be studied.

Exploring the environment with acoustics

The only way to bound errors drifting when performing dead reckoning is to obtain an absolute position reference regularly. We have seen that resurfacing to make a GNSS fix is generally undesirable. The idea is to obtain another exteroceptive information.

Luckily, acoustic waves can propagate energy on very long distances in water, making the use of acoustic sensors perfectly appropriate for underwater robots. The travel distance depends on the transmitted frequency [105]. Thus, it is possible to communicate with surface or other underwater systems. However, the throughput of information with acoustic support is low. It is also possible to use acoustic signals to measure ranges to targets as the seabed, surface, rocks, or other obstacles.

Finally, acoustic sensors provide the required information for reaching an underwater localization. Nevertheless, processing this data is far from being straightforward.

The first strategy is called *map-based navigation*. It requires the knowledge of the local map but provides interesting results [123]. It can be based on a bathymetric map, a magnetic field, gravitational anomalies, or other geophysical parameters. The only condition is that the navigation's area variates sufficiently in the measured parameters to allow localization. However, it requires an *a priori* knowledge of the map. This issue motivates research on Simultaneous Localization And Mapping (SLAM) methods [144].

In any case, data from the environment have to be acquired but naturally contain errors: noise and outliers. For instance, echo sounders can measure distances. However, the measurement is sensitive to the sound speed, which varies with the depth and other physical parameters. In addition, acoustic scattering effects cause more errors and outliers. Rejecting outliers when dealing with acoustic data is thus crucial and has already been addressed in [157] for example.

More generally, when collecting data in the underwater environment, the difficulty relies upon extracting reliable information. There are a lot of outliers, uncertainties, and unknown parameters. Finally, the objective is to use relatively cheap sensors and still manage to bring reliable information out.

1.3 Research approach

The two challenges mentioned above have been studied in a specific context. Indeed, the issues are pretty general, and different approaches can be envisioned. We present now in more detail the objectives and the choices made for each challenge.

1.3.1 Control a towed sensor

Modelization of a car-trailer system

In order to design a satisfying controller for a system, we usually want to understand the dynamics of this system. A modelization using differential equations is then proposed to approximate the behavior of the system.

Concerning the magnetometer towed by a robot, the modelization of the rope between the two systems is complex. The management of such a cable constitutes a whole topic of research, especially in the scope of the umbilical of a ROV [154, 104, 92, 159]. To keep it simple, we chose to model it by a rigid link. This assumption makes sense in our context as the rope is almost always taut: when scanning an area, a robot should have a constant speed and move straight most of the time. In this way, the measurements are made at a constant chosen frequency.

Besides, although the controller should handle a 3D context in an underwater environment, the depth of the system can be regulated independently. Thus, we can only consider the problem in the 2D-horizontal plane.

With these two assumptions gathered, the system can be modelized more easily. The behavior can be related to the one of a car towing a trailer. At last, a non-neglectable advantage of considering the car-trailer system is that developed approaches can be tested with much more ease. Indeed, the process to carry out experimentations is more intricate when it requires to go at sea, without forgetting the dependence on weather and sea conditions.

Boatbot: the automatized boat of ENSTA Bretagne

The final objective is to scan an area at sea. In the context of the search for the *Cordelière*, the DRASSM carries out wrecks search with its research vessel: the *André Malraux*. This vessel has been designed and equipped especially for underwater archaeological exploration. However, as it has an important draft, it cannot easily lead research in shallow water, and there are shallow waters in the survey area of the *Cordelière*, close to the shore. So one objective is to imagine a system able to drag a magnetometer in shallow waters. In a second phase, the goal is to make this system entirely immersible to avoid perturbating or being perturbed by the marine traffic, which is significant in this area.

So the first developed system was an inflatable boat towing a magnetometer. This boat has been automatized, using an electrical engine behind the helm. A differential Global Navigation Satellite System (GNSS) and an Inertial Measurement Unit (IMU) with magnetometers are used to localize and access the boat's orientation. Thus, the automatized boat, named *Boatbot*, became autonomous in direction and is able to follow survey lines accurately, even with wind and currents [152]. The speed has also been automatized to obtain a constant speed during the measurements, independent of currents and other perturbations. Figure 1.4 shows Boatbot towing a kayak, and the magnetometer is immerged below the kayak. The kayak was a surface intermediary to ensure the rope could not be cut by the motor's propellers.

Figure 1.4: Boatbot towing a kayak and a magnetometer, Bay of Brest, July 2018.

The system Boatbot-magnetometer, which can be viewed as a surface vehicle towing an immerged fish, presents different issues. Indeed, even if the surface vehicle is fully autonomous, it requires someone aboard for safety (and legal) reasons. Boatbot could be replaced by an Autonomous Surface Vehicle (ASV), but there are still other issues. Indeed, the position of the fish is uncertain and hard to control (especially the depth) as we have no information on it. This uncertainty grows with the length of the cable, which should be long enough to have the magnetometer close to the seabed. So the idea is to use an AUV instead of the surface robot: it can be less concerned by the marine traffic, and it can measure and control its immersion at any time. Thus we can hope to have better control of the position of the magnetometer due to a shorter cable, and it is possible to envision missions in deep waters. However, this solution leaves the system without any link to the surface, and therefore to any GNSS. As a result, the localization of the system, especially the magnetometer, becomes much more complex in the horizontal plane.

Constraints satisfaction

Last but not least, we want some guarantees on the control of the deported sensor. Indeed, the presented challenge mentioned a “proper” scan, and the notion of safety is brought up. These additional requirements can be expressed in the form of constraints. Then, the objective is to validate the system in the sense that the constraints are always satisfied.

The Constraint Satisfaction Problems (CSPs) can describe many search problems and have thus been studied in various contexts. For instance, a problem of estimation is solved with a

maximum constraints satisfaction approach, using genetic algorithms, in [119].

Concepts behind CSPs are described in [156], and a survey of existing algorithms is proposed in [88]. Set-membership approaches have also proved to be suitable to address these problems, as in [71], where the SLAM problem is handled.

Sometimes, several constraints are involved, and they cannot be satisfied simultaneously at the same time. Then a CSP can be turned into a Constraint Optimization Problem (COP) where a maximum of constraints should be satisfied. The notion of *soft constraints* can also be introduced: a soft constraint is a non-mandatory constraint. An example is given in [140] with *fuzzy constraints*, *i.e.*, constraints that are partially satisfied; the degree of satisfaction being represented by a value included within 0 and 1.

In this thesis, set-membership approaches have been selected to address CSPs because they provide guaranteed results, and also because they allow dealing with strong non-linearities with ease.

1.3.2 Localization in pool

Underwater localization based on acoustic data

The underwater localization problem is both complex and crucial. The objective is to manage to estimate the position of an underwater robot by processing acoustic data from its environment. These data contain much noise and outliers. Thus their interpretation is far from being straightforward.

Firstly, in order to handle this intricate problem and ease the tests, the context has been chosen as follows. The objective is to localize an underwater robot in a pool using an echosounder. In this way, we can focus on the data interpretation part by considering a controlled environment. In addition, it is easy to carry out experimentations with data acquisition and test the different approaches.

Representing information

The underwater localization problem highly depends on the type, the quality, and the quantity of collected data. For instance, with an accurate LBL system correctly deployed in good conditions, the robot has direct and frequent access to its relative position with good precision. Then the information is not handled in the same way as with a low-cost echosounder merged with a pressure sensor that approximates the local water level compared to a water level reference. There are much more uncertainties and a more complex representation of the information in the latter case.

So, the objective is to find an estimation of the position of the robot. Usually, probabilistic approaches are used for estimation problems. These methods compute a unique solution that represents the estimated position. This solution comes with covariance matrices to represent the uncertainties [121]. We can mention the well-known Kalman filter [82], which has been proved to

be efficient in linear and Gaussian contexts. Extensions to this filter have been primarily studied, with linearization, Gaussian mixtures, and other Bayes filters to deal with non-linearities and multitarget [106, 160]. In the non-linear case, stochastic methods have also been largely studied and provide relevant results [151]. For instance, solutions to the SLAM problem have been dealt with [50, 4, 147]. There are also examples of underwater localization using bathymetry, as in [5]. However, these approaches may fail from the moment there are strong non-linearities or not enough reliable data. The latter example of bathymetric SLAM requires the use of a multibeam echosounder, which is an expensive sensor that gives many data. The obtained results may be incorrect as soon as this sensor is substituted for an inexpensive echosounder. Regardless, there are very few studies of underwater localization using a single-beam echosounder and a Digital Elevation Model (DEM): [29], or [149] with multiple cooperative vehicles. We can also mention a bathymetric SLAM approach using sparse data (see [11]).

Set-membership approaches have shown to be particularly adapted to deal with strong non-linearities and considerable uncertainties. Thus, the estimation problem can be solved as soon as the uncertainties are bounded by known values, such that the actual value is guaranteed to belong within these bounds. Instead of computing a unique solution with covariance matrices, like with probabilistic approaches, the result consists of a *solution set* that contains the actual solution [161, 158, 107]. So this solution set contains all the feasible solutions, and the computations are guaranteed never to remove such a possible solution. The drawbacks of this approach are that an infinite number of solutions are considered, some pessimism can be introduced due to the guarantee's property, and less information on the values is considered compared to probabilistic approaches: the probability distributions used to represent the values are no longer considered. Another difference between a probabilistic and a set-membership approach is the determinism of the latter. The obtained result is always the same, which is not guaranteed when using a probabilistic approach.

Among the drawbacks of the set-membership approaches, the lack of probability distribution is often pointed out. Actually, it would not be a *probability distribution*, but a *possibility distribution*, as all computations are deterministic. This distribution is completely binary: an element either belongs to the solution set or does not belong to it. Its membership is either equal to 1 or 0. There is no notion of *partial belonging*, or *possible belonging*. An answer to this issue is given by the *fuzzy logic* approaches. The principle is precisely to break this binarity by choosing a possibility distribution to represent a value. Thus, instead of considering two bounds that contain the possible values, a *fuzzy set* is defined such that the membership of a given value can be within 0 and 1.

1.3.3 Contributions and outlines

This thesis comes within the scope of the search for the *Cordelière*. Therefore, the main contributions are linked to the exploration of the seabed with a deported sensor, namely a

magnetometer, and especially its control, safety, and localization.

Firstly, Chapter 2 introduces some tools from dynamical systems and non-linear control theory to handle control issues. The feedback linearization is mainly studied. Then, the set-membership approach is presented with interval analysis. Finally, the fuzzy logic is also studied to complete the interval analysis approach. These tools are directly illustrated with some basic examples. They will be used through the following chapters with applications linked to the challenges.

Afterward, Chapter 3 uses the presented tools from non-linear control theory to deal with the problem of the car with a trailer. Notions of controllability and flatness are first introduced, and the approach is illustrated on a Van der Pol vector field. A controller for the trailer is designed. The follow set notion is also defined. It represents the set of all the states in which the robot can be without violating some constraints. The previously designed controller is validated under these constraints by characterizing its follow set using interval analysis.

In Chapter 4, we address the localization problem. We present how acoustic data are collected and how they can be processed. Intervals are used to handle them, and different examples are given dealing with the localization of a robot in a pool. An approach by constraints is proposed, with particular care on the reliability of the chosen constraints. New contractors are designed for this purpose.

In order to go further, Chapter 5 also handles the localization problem but brings tools from fuzzy logic to propose a new approach. It consists in combining sets, allowing to deal finely with estimation problems. The principle is to use a constraints approach and provide a means to prioritize the constraints so that as many constraints as possible are satisfied. Interval analysis is used to characterize the different α-*cuts* of the fuzzy set,and thus obtain the values that *possibly* belong to the solution set. This possibility is represented by α, which can be chosen lower or higher depending on degree of membership we want, which can translate a degree of accuracy or guarantee for example. The interest in the new fuzzy tools is illustrated on an actual robot in a pool.

Finally, Chapter 6 concludes this document by summarizing the contributions and detailing some prospects.

CHAPTER 2

Theoretical tools

Contents

2.1 Introduction

In order to handle the mentioned robotic challenges, some theoretical tools require to be presented. The first challenge is about the control of a deported sensor. Thus, tools from the control theory field need to be introduced to deal with the design of a controller. More precisely, the dynamics of a towed object being non-linear, these tools belong to *non-linear control theory*, and we focus on the *feedback linearization* approach. This theory goes along with the study of dynamical systems, whose fundamentals are recalled beforehand.

In a second part, the theory and concepts of *interval analysis* are presented. Tools from this field are necessary to develop our approaches for both challenges. Indeed, the control of the deported sensor and the localization problem raise non-linear problems. On top of that, we would like some guarantee on the analysis and the results of the proposed approaches. Therefore, tools from interval analysis are perfectly adapted to handle non-linearities and lead computations while keeping a guarantee on the result. Especially, contractors are introduced and studied as they will be used all through this document. They are operators allowing to reduce the domains of the studied variables while keeping all the feasible solutions. They can be used for solving various problems with a set-membership approach.

Finally, intervals are exploited to deal with uncertainties: they provide a result that is reliable as soon as the uncertainties have been bounded in the intervals representing the studied variables. This point is significant in our context where we want reliable results and where noise is omnipresent and non-neglectable. Thus, in order to go into uncertainty representation in-depth, *fuzzy logic* is introduced. This other branch of mathematics provides the tools to deal with fuzzy sets, more flexible and appropriate than intervals to represent uncertainties. Interval analysis can still be used to carry out the computations and thus keep guaranteed results.

Therefore, this chapter starts with notions of dynamical systems and non-linear control theory with some basic examples. Especially, feedback linearization is introduced. Then, an introduction to intervals is proposed. The essential elements are presented, especially the contractors, which are powerful and very useful tools. Finally, fuzzy logic concepts are depicted, and their benefits are underlined on the same examples as for interval analysis.

2.2 Non-linear control theory

Control theory began in the 19th century. Its purpose is the control of dynamical systems. We can divide it into two branches: linear control and non-linear control theories. The first branch proposes general techniques to study and solve linear systems [89]. Concerning non-linear control theory, existing approaches often offer solutions to specific classes of problems, so two systems may be studied with two different methods, depending on their stability properties. Nevertheless, this field has been widely studied, and various tools are available today, as presented in [143, 69, 85], among other books. Many applications in robotics can be found as in [145, 32,

59, 53]. Sometimes, only solutions near a stable point are searched for, and the system can then be *linearized* in the neighborhood of this point. It means that the system is approximated by a linear system, and thus techniques from the first branch can be used. As linear techniques are well-known and efficient, we often want to find a way to linearize non-linear systems. That is the case with the *feedback linearization* approach, which consists of making a change of variables in order to obtain an equivalent linear system.

2.2.1 Dynamical systems

First of all, the formalism of dynamical systems is quickly presented as it is fundamental for dealing with control theory.

A robot, or any mechanical system, can be modeled by state equations. These equations represent the dynamics of the system with its inputs and outputs. Most of the time, for real systems, it is (strongly) non-linear. Such systems are described as follow:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t)\right) + \mathbf{g}\left(\mathbf{x}(t)\right) \cdot \mathbf{u}(t) \qquad \textit{(evolution equation)} \qquad (2.1)$$

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{x}(t)\right) \qquad \textit{(output equation)} \qquad (2.2)$$

where $\mathbf{x}$ is the state of the system and $\dot{\mathbf{x}}$ its derivate with respect to time. Its dimension is denoted by n. The vector $\mathbf{u}(t)$ is the input, and $\mathbf{y}(t)$ the output of the system. The functions $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ are assumed to be smooth. Each of these variables depends on the continuous-time variable t. For the sake of clarity, we will not always write the time dependence in the following. For instance, we shall write $\mathbf{x}$ for $\mathbf{x}(t)$.

The evolution equation describes the dynamics of the system: it is a differential equation that corresponds to physical behavior. It can be obtained using the laws of physics or any other means to model these dynamics.

In this section, we are interested in finding a way to control such systems. The output $\mathbf{y}(t)$ is used for obtaining an estimation of the current state of the system $\hat{\mathbf{x}}(t)$. This estimation is made from measurements and is more or less uncertain, depending on whether the state is easily and precisely measurable: $\hat{\mathbf{x}}(t) \approx \mathbf{x}(t)$. A *state observer* is the name of the tool that returns the state estimation from the output. Depending on the outputs, its design can be a complex task, requiring the use of the least-squares approach [9], a Kalman filter [82, 146] or a Bayes Filter [160]. Interval analysis can also be used for this task as in [51, 128, 127].

However, the design of a state observer is not really the topic of control theory. To keep it simple, we assume that we have a perfect observer, such that we know the state of the system: $\hat{\mathbf{x}}(t) = \mathbf{x}(t)$. It means that $\mathbf{y}(t)$ corresponds directly to the variables to be controlled. This perfect observer simplification does not make what follows less general, as the method used requires the state's knowledge anyhow and is independent of the state observer. Thus, in the following, we assume that the state is known at any time: we study a state feedback controller.

2.2.2 Feedback linearization

In order to take on non-linear dynamical systems and give an intuition on how feedback linearization works, we start straight away with a small example.

Lotka–Volterra equations

This example is not taken from a robotics application. Another simple example with a robotic system is given further, and the whole study of a complex robotic system is the purpose of Section 3.3.

The Lotka–Volterra equations [10], or predator-prey equations, form a non-linear system that can be used to represent the dynamics of the populations of two species: a predator and its prey. The populations evolve with respect to these differential equations:

$$\begin{cases} \dot{x_1} = \alpha x_1 - \beta x_1 x_2 \\ \dot{x_2} = \delta x_1 x_2 - \gamma x_2, \end{cases} \tag{2.3}$$

where x_1 and x_2 represent the population (in thousands) of preys and predators, respectively, over time t (in years). The constants $\alpha, \beta, \delta, \gamma$ are parameters depicting the birth and death rates for each species.

Let consider the following values for parameters:

$$\begin{cases} \alpha = 1 \\ \beta = \dfrac{1}{5} \\ \delta = \dfrac{1}{3} \\ \gamma = 1 \end{cases} \tag{2.4}$$

An input u, which depends on time, is also introduced for adding or removing some preys in the system. This latter variable allows us to control the system: we can choose its value over time to act on the system. Then, finding the control law that makes the system converge to the desired state amounts to find an expression for u. Finally, the system is described by

$$\begin{cases} \dot{x_1} = x_1 - \dfrac{1}{5} x_1 x_2 + u \\ \dot{x_2} = \dfrac{1}{3} x_1 x_2 - x_2. \end{cases} \tag{2.5}$$

A simulation of 20 years of the system with the initial state of ten thousand preys and two thousand predators $\begin{pmatrix} 10 \\ 2 \end{pmatrix}$ and with no command ($u = 0$ on $[0, 20]$) gives the evolution represented in Figure 2.1. It is called an *open-loop control* because it does not take into account the output of the system. It can give a first overview of the stability of the system, especially with the representation in the (x_1, x_2)-plane that shows a stable cycle.

Finding a control law

The objective is to find an expression of u to control the number of preys over time. We assume that x_1 and x_2 are known at any time.

By choosing u such that

$$u = -x_1 + \frac{1}{5}x_1x_2 + v \tag{2.6}$$

and replacing it in the first equation of (2.5), we obtain the following linear equation:

$$\dot{x_1} = v. \tag{2.7}$$

Here we have made a change of variable that leads to a linear equation and whose new input is v. That is how we obtain a *feedback linearization*.

Now we can use standard linear methods to regulate the system. Thus, if we want the population x_1 to follow a dynamics Ψ, we can use a proportional-derivative controller:

$$v = \Psi - x_1 + \dot{\Psi} \tag{2.8}$$

which leads to:

$$\dot{x_1} = \Psi - x_1 + \dot{\Psi}. \tag{2.9}$$

So it means that the error $e = \Psi - x_1$ between the value of x_1 and the set-point is solution to the differential equation:

$$\dot{e} + e = 0. \tag{2.10}$$

This differential equation guarantees that the error converges exponentially towards 0, and this is precisely the expected behavior for an error.

The control law u can then be written as

$$u = -2x_1 + \frac{1}{5}x_1x_2 + \Psi + \dot{\Psi}. \tag{2.11}$$

For instance, if we want the number of thousands of preys to oscillate around 3 over time, it means that we want $x_1(t)$ to be equal to $\sin(\frac{t}{2}) + 3$. It yields

$$\Psi(t) = \sin(\frac{t}{2}) + 3 \tag{2.12}$$

$$e(t) = \sin(\frac{t}{2}) + 3 - x_1(t). \tag{2.13}$$

The wanted expression for u is thus

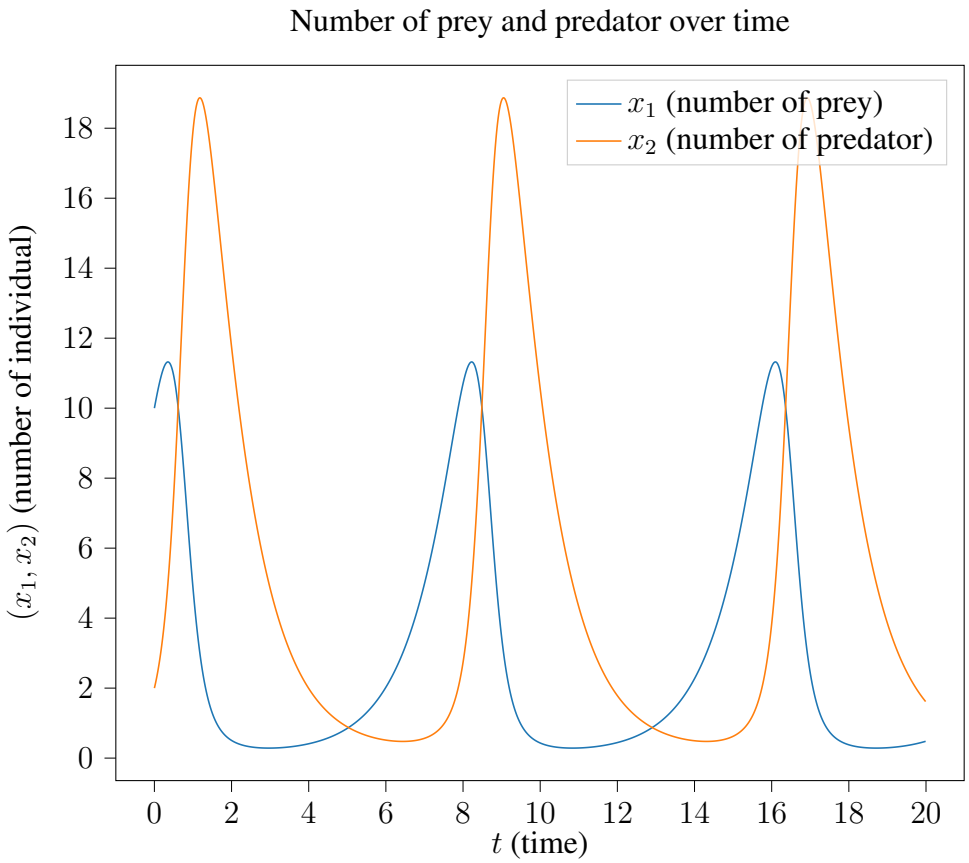
$$u = -2x_1 + \frac{1}{5}x_1x_2 + \sin(\frac{t}{2}) + \frac{t}{2}\cos(\frac{t}{2}) + 3. \tag{2.14}$$

We remark that the knowledge of x_1 and x_2 is indeed required to compute u. If these values were not precisely known, then the obtained control law may not work as expected. Now, if we implement this expression in the simulation, we obtain the Figure 2.2, with the number of prey converging towards a sine curve shifted around ordinate 3.

Note that the controller depends on time and is not linear. Although a linearization has been done to find the controller, no approximation has been introduced. It is more advantageous to use this method, contrary to a linearization near a stable point which adds approximation errors.

Finally, we have succeeded in finding a control law that makes our system converge towards the desired dynamics. The method used here is called *feedback linearization* and consists in computing the successive derivatives of the expression of the error until it depends on the input. In the example, only one derivative had to be calculated, but if the new objective is to regulate the number of predators, instead of preys, then the second derivative has to be computed, to make u appear in the analytical expression.

This example gives an intuition on this method and broadly shows the assumptions and the different steps. In the following, we enter more into details and present the mathematical tools used behind this approach.



Number of predator with respect to number of prey (representation in the (x_1, x_2)-plane)

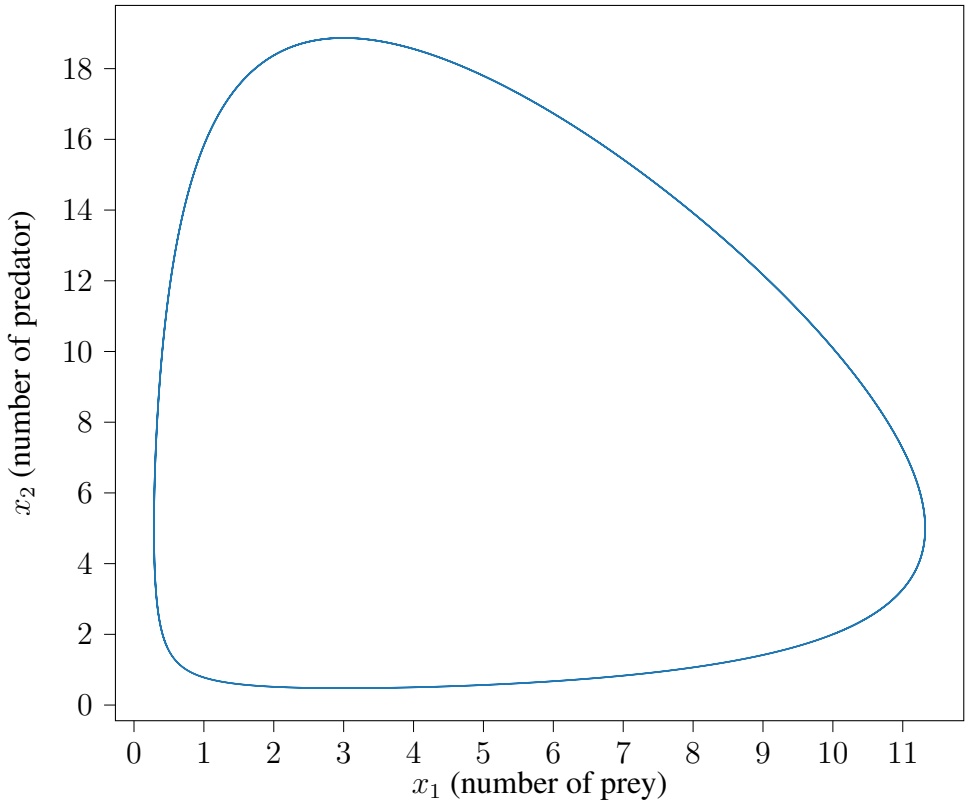
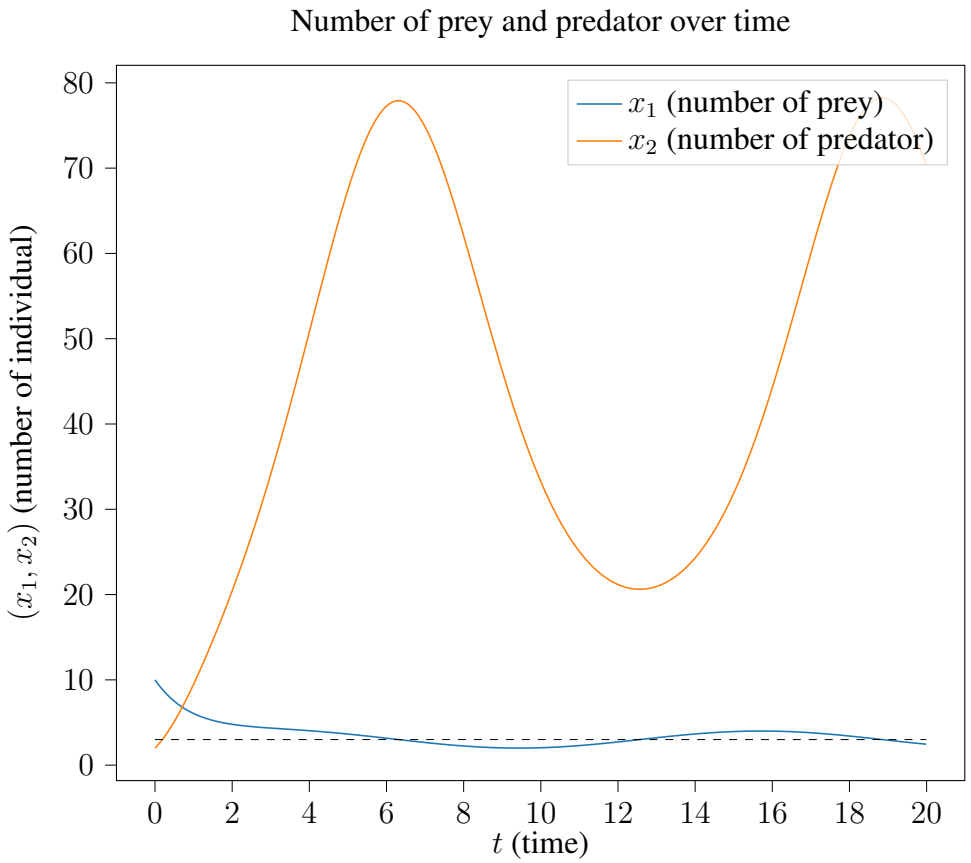


Figure 2.1: Simulation of the evolution of the populations of prey and predator without input.



Number of predator with respect to number of prey (representation in the (x_1, x_2)-plane)

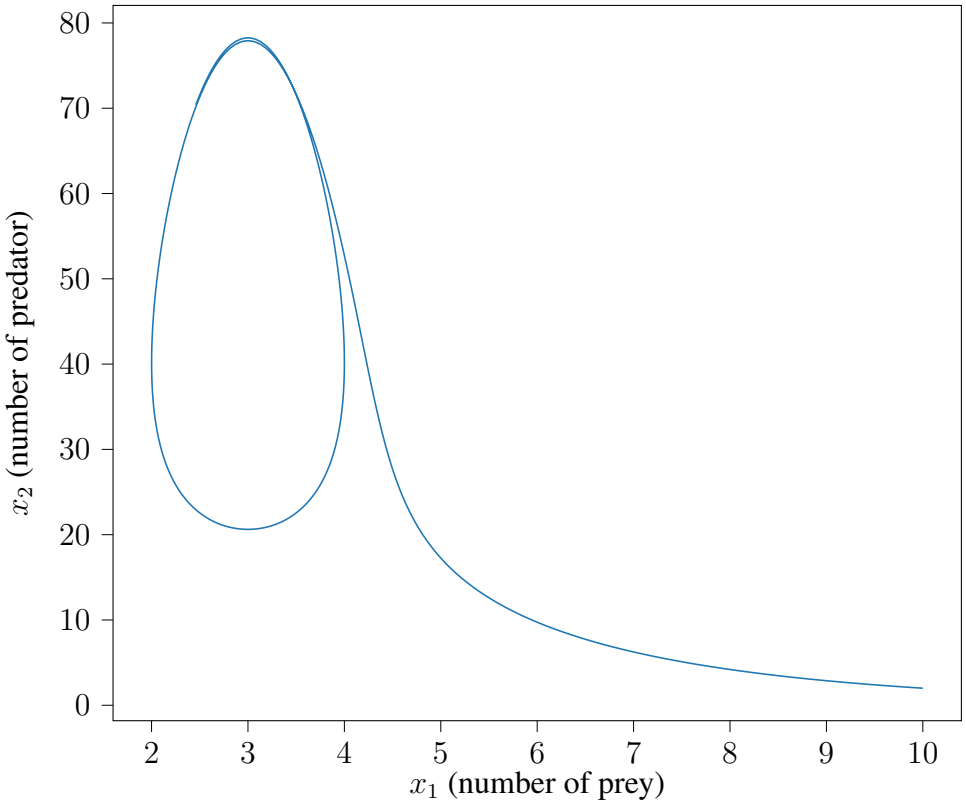


Figure 2.2: Simulation of the evolution of the populations of prey and predator. The input has been designed to make the number of preys converge towards a sine curve around ordinate 3 (identified by the dashed line).

2.2.3 Differential geometry

Let us tackle the mathematical concepts hidden behind feedback linearization. In order to control a dynamical system, described by (2.1), we usually need to study the output equation (2.2) and, more especially, its dynamics. It means that the successive derivatives of the output $\mathbf{y}$ have to be computed. This task is relatively easy and immediate on simple systems. However, most of the time, dynamical systems used to model real mechanical systems are complex. It leads to intricate computations, and this is why we use tools from differential geometry to abstract these computations and automatize this step.

Let introduce these tools without defining too many concepts to keep it simple (see [143, Chapter 6] for more details). The evolution function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ described in (2.1) is actually named a *vector field* to match the terminology of differential geometry. Its dimension is n.

Note that we use here the $\mathbb{R}^n$ space as, in our applications, all the state variables are real. However, in differential geometry, we work in a more general topological space named *manifold*. In the following, $\mathbf{M}$ denotes an n-dimensional manifold. Then, we usually defined a *coordinate patch* denoted by $\mathbf{P}$ which is a connected, open set around a point in the manifold $\mathbf{M}$. This patch allows to define a coordinate system and finally to define tools as Lie derivatives properly.

The state $\mathbf{x}$ of the robot evolves on the vector field $\mathbf{f}$, which corresponds to its dynamics. In other words, at a time t, the robot is in a state $\mathbf{x}(t)$, and since the robot obeys to laws of physics, the evolution of the state is determined by the vector field $\mathbf{f}$, modeling these laws. So the derivative (with respect to time) of $\mathbf{y} = \mathbf{h}(\mathbf{x})$ is the rate of change of $\mathbf{h}$ on the vector flow (trajectory on the vector field $\mathbf{f}$) at the point $\mathbf{x}(t)$. It is named a *Lie derivative* and is denoted by $\mathcal{L}_\mathbf{f}(\mathbf{h})$ or simply $\mathcal{L}_\mathbf{f}\mathbf{h}$.

More formally, we have

$$\mathcal{L}_\mathbf{f}\mathbf{h} = \nabla\mathbf{h} \cdot \mathbf{f}, \tag{2.15}$$

where $\nabla\mathbf{h}$ denotes the gradient of $\mathbf{h}$: $\nabla\mathbf{h} = \frac{d\mathbf{h}}{d\mathbf{x}}$.

Now if we consider once again the equation of (2.2), when we derive $\mathbf{y}$, we get:

$$\begin{aligned}
\dot{\mathbf{y}} &= \frac{d\mathbf{h}}{d\mathbf{x}}(\mathbf{x}) \cdot \dot{\mathbf{x}} \\
&\overset{(2.1)}{=} (\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \cdot \mathbf{u}) \cdot \frac{d\mathbf{h}}{d\mathbf{x}}(\mathbf{x}) \\
&= \underbrace{\frac{d\mathbf{h}}{d\mathbf{x}}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})}_{=\mathcal{L}_\mathbf{f}(\mathbf{h})(\mathbf{x})} + \underbrace{\frac{d\mathbf{h}}{d\mathbf{x}}(\mathbf{x}) \cdot \mathbf{g}(\mathbf{x})}_{=\mathcal{L}_\mathbf{g}(\mathbf{h})(\mathbf{x})} \cdot \mathbf{u}.
\end{aligned} \tag{2.16}$$

We can see that we are using Lie derivatives as soon as we want to derive an expression depending on a state variable. We can now define the successive Lie derivatives recursively: the ith derivative is

$$\mathcal{L}_\mathbf{f}^{(i)}\mathbf{h}(\mathbf{x}) = \mathcal{L}_\mathbf{f}\mathcal{L}_\mathbf{f}^{(i-1)}\mathbf{h}(\mathbf{x}) = \frac{d\left(\mathcal{L}_\mathbf{f}^{(i-1)}\mathbf{h}(\mathbf{x})\right)}{d\mathbf{x}} \cdot \mathbf{f}(\mathbf{x}). \tag{2.17}$$

The feedback linearization method consists in deriving the output of the system until the input $\mathbf{u}$ appears in the equation. The number of derivatives needed to be computed depends on the system and is called the relative degree. It is denoted by r, and it is the smallest integer such that

$$\forall i \in \{0, \dots, r-1\},\ \mathcal{L}_{\mathbf{g}}\mathcal{L}_{\mathbf{f}}^{(i)}\left(\mathbf{h}(\mathbf{x})\right) \neq 0. \tag{2.18}$$

In addition we necessarily have $r \leq n$.

So, for each non-linear system that is feedback linearizable [85, p. 508], these tools can be directly used to find a control law.

2.3 Interval analysis

Notions of non-linear control that have been presented above will be used later to design a controller of a towed sensor. Another theoretical tool that is presented now will help to validate such a controller under some constraints. This tool is interval analysis, and it will also be used to deal with localization and, more generally, set estimation.

Interval analysis is a branch of mathematics that provides tools to handle various problems. The first issue, which was one of the main motivations to carry out researches on intervals, was guaranteeing numerical computations despite rounding errors due to the finite representation of numbers in machines [125]. Since then, numerous applications have been found and have developed a new interest in this field. In robotics, we can mention relevant uses of this approach in localization [39, 133], control [112], or planning [65, 123], to name a few.

Intervals are specially adapted to deal with uncertainties. These uncertainties can barely be ignored, and different tools have already been designed to address them. Apart from interval analysis, we can mention Bayesian approaches, which are widely used and have no more need to prove their efficiency. Concerning robotics, the article [150] can give a first overview and references. The fuzzy logic field should also be mentioned, as it brings one additional point of view on representing these uncertainties.

For now, let us focus on intervals. Concretely, instead of considering a real number x for representing a value, we consider an interval $[a, b]$ that contains x: $x \in [a, b]$, with $a, b \in \mathbb{R}$. This interval is denoted by $[x]$. The variables a and b are called the *lower bound* and the *upper bound* of the interval, respectively, and are often denoted by x^- and x^+.

2.3.1 Definitions

Intervals

More formally, an interval is a connected subset of $\mathbb{R}$. Call $\mathbb{IR}$ the set of all intervals, and $[x]$ an element of this set. The interval $[x]$ is defined by its lower and upper bounds, denoted by x^- and x^+, respectively:

$$[x] = \left\{ x \in \mathbb{R} \mid x^- \leq x \leq x^+ \right\}. \tag{2.19}$$

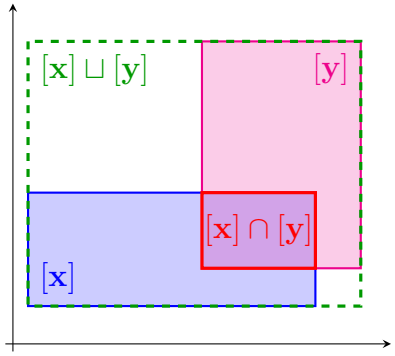


Figure 2.3: Example of operations on two boxes $[\mathbf{x}]$ and $[\mathbf{y}]$.

For example, $[2,4], [-\frac{1}{3}, 10], [-\pi, \sqrt{2}]$ are intervals. Intervals can also be *degenerate*, when $x^- = x^+$. For instance, $[1,1] = \{1\}$ is a degenerate interval. We can mention two more examples of special intervals: the empty set $\emptyset$ and $[-\infty, +\infty]$.

As an interval is a set, set operators can be extended to intervals. The intersection of two intervals $[a]$ and $[b]$ is

$$[a] \cap [b] = \left\{ x \in \mathbb{R} \mid \max(a^-, b^-) \leq x \leq \min(a^+, b^+) \right\}. \tag{2.20}$$

The only subtlety is for the union of intervals: the interval hull operator, denoted by $\sqcup$, is used to keep the integrity of intervals. Indeed, the union of two intervals can be a non-connected set – it is not any more an interval. That leads to the definition

$$[x] \sqcup [y] = \left\{ x \in \mathbb{R} \mid \min(x^-, y^-) \leq x \leq \max(x^+, y^+) \right\}. \tag{2.21}$$

For example, $[-3,-1] \cup [17,42]$ is not an interval, but $[-3,-1] \sqcup [17,42] = [-3,42]$ is one.

Now we can define *interval vectors* as the cartesian product of intervals, to extend the definitions to multiple dimensions. For instance, a two-dimensional interval vector is the cartesian product of two intervals: $[\mathbf{x}] = [0,1] \times [-1,2]$. It is also called a *box* or a *two-dimensional box* (*2D-box*).

Set operators are also well defined on interval vectors, with the same operations as for (1D-)intervals applied on each dimension. Figure 2.3 gives an illustrative example.

Besides, as an interval represents a value as a real, all the arithmetics operations are also extended to intervals. All the implementation details of arithmetics are not presented in this document, but the principle is to carry out the computations using each boundary of the intervals. The reader may refer to [41] for further information. For instance, the addition of two intervals $[x]$ and $[y]$ is defined by

$$[x] + [y] = [x^- + y^-, x^+ + y^+]. \tag{2.22}$$

The subtraction of two intervals is defined similarly. The multiplication and division operations should be applied with care, especially when 0 is included in one of the terms. More complex operations are also defined, as trigonometric functions ($\cos$, $\sin$, etc), exponential, minimum, and so on. Note that computations with intervals are thus non-limited to linear operations. Since

then, an essential advantage of interval analysis is to deal with strong non-linear problems still guaranteeing the result.

In the same way, this arithmetic is also extended to interval vectors.

Constraint satisfaction problem

Many problems can be formalized into a *Constraint Satisfaction Problem (CSP)*. *Constraint Programming (CP)* is a field of research that addresses these CSPs. Details of this field can be found in [136], for example. Note that we can also speak of Constraint Networks (CN) [98]. The formalism is described in [139, Chapter 6] as follows.

A CSP is a triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where each set is: the *set of variables*, the *set of domains* associated with these variables, and the *set of constraints*, respectively. An element of the latter is a *constraint* and is defined by a pair $\langle scope, rel \rangle$ where $scope$ is the domains of the variables involved in the constraint, and rel is the relation between these variables constraining their possible values.

In this document, we use a function to represent the constraints set. The formalism becomes as follows.

Proposition 2.3.1. *Fix* $\mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$ *where* m *is the number of constraints. Then, a CSP* $\mathcal{H}$ *is described by*

$$\mathcal{H} : \begin{cases} \mathbf{x} \in [\mathbf{x}] \\ \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{cases} \tag{2.23}$$

Proof. Indeed, we can find back the original formalism by defining the triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ such that:

- $\mathbf{x}$ is the vector of all elements of $\mathcal{X}$;
- $[\mathbf{x}] = \mathcal{D}$;
- $\mathcal{C}$ is a set of m elements such that the ith element is defined by the pair

$$\langle scope, rel \rangle = \langle \text{domain}(h_i), \{h_i(\mathbf{x}) = 0\} \rangle ,$$

  with h_i the function that returns only the ith output of the function $\mathbf{h}$.

Therefore, we have a CSP. □

Example 2.3.1. For instance, for $n = 3$, assume that we have the following constraints ($m = 2$):

$$\begin{cases} x_1 - x_2 = 1 \\ x_1 \cos(x_3) \leq 0, \end{cases} \tag{2.24}$$

with $[\mathbf{x}] = \mathbb{R} \times \mathbb{R} \times [-\pi, \pi]$.

Let show that it is a CSP. In order to match the formalism, an additional variable is introduced in the following equivalent system of equations:

$$\begin{cases} x_1 - x_2 = 1 \\ x_1 \cos(x_3) + x_4 = 0, \end{cases} \tag{2.25}$$

with $x_4 > 0$ (so now $[\mathbf{x}] = \mathbb{R} \times \mathbb{R} \times [-\pi, \pi] \times [0, +\infty]$). In this way, all the constraints are expressed as equalities, and we thus have a definition for the function $\mathbf{h}$:

$$\begin{array}{rccc} \mathbf{h}: & \mathbb{R}^4 & \to & \mathbb{R}^2 \\ & \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} & \mapsto & \begin{pmatrix} x_1 - x_2 - 1 \\ x_1 \cos(x_3) + x_4 \end{pmatrix}. \end{array} \tag{2.26}$$

Such a CSP is associated to a *solution set* $\mathbb{S}$ which is defined by

$$\mathbb{S} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{h}(\mathbf{x}) = \mathbf{0}\}. \tag{2.27}$$

The solution set of this class of problems can be computed in a reasonable time using heuristic combinations or combinatorial optimization. Otherwise, in general, this is an NP-hard problem; that is, there is no available algorithm able to compute the result with a polynomial complexity in the number of variables. Interval analysis also brings methods to solve these problems efficiently, with the advantage of offering a guarantee on the result. Indeed, intervals are particularly adapted to address set membership problems, even in a non-linear context.

2.3.2 Contractors

Definition

A method using interval analysis is presented to solve CSPs. A *contractor* [23] is an operator that contracts the box $[\mathbf{x}]$ into a smaller box $[\mathbf{x}']$ that still satisfies the constraints [80]. In other words, we have

$$[\mathbf{x}'] \subset [\mathbf{x}], \tag{2.28}$$

and

$$\mathbb{S} \cap [\mathbf{x}] = \mathbb{S} \cap [\mathbf{x}']. \tag{2.29}$$

So $[\mathbf{x}']$ is a better approximation of the researched solution set $\mathbb{S}$.

An example is given in Figure 2.4 with a contractor $\mathcal{C}$ that is associated with a set S. It means we can rewrite (2.29) with

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbb{S} \cap [\mathbf{x}] = \mathbb{S} \cap \mathcal{C}([\mathbf{x}]). \tag{2.30}$$

Note that this contractor is *minimal*. It means that it returns the smallest box that wraps the set without eliminating any of the solution points [38].

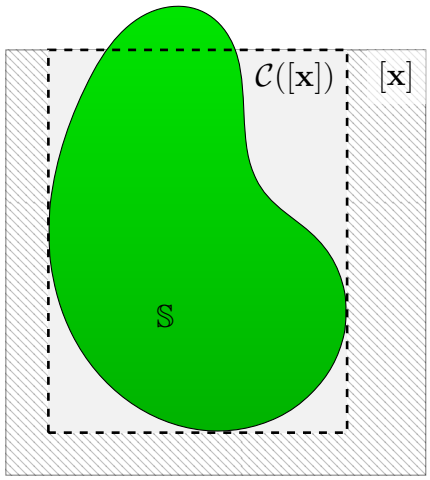


Figure 2.4: Example of a (minimal) contraction of the box $[\mathbf{x}]$ with the contractor $\mathcal{C}$ associated with the set $\mathbb{S}$.

More formally, a contractor $\mathcal{C}$ is defined by the three following properties:

$$\begin{array}{lll} (i) & \forall [\mathbf{x}] \in \mathbb{IR}^n, \quad \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)} \\ (ii) & [\mathbf{x}] \cap \mathbb{S} \subset \mathcal{C}([\mathbf{x}]) & \text{(consistancy)} \\ (iii) & [\mathbf{x}] \subset [\mathbf{y}] \implies \mathcal{C}([\mathbf{x}]) \subset \mathcal{C}([\mathbf{y}]). & \text{(monotonicity)} \end{array} \tag{2.31}$$

The first property is the *contractance* property. It guarantees that no candidate from outside the input box can be considered.

The second one is the *consistency* property. It is equivalent to (2.30) and guarantees that no solution point can be eliminated.

The last one is the *monotonicity* property. It is not strictly required to define a contractor, but here we use only monotonic contractors. Keeping this property allows having some guarantees on the convergence time, as described in [24].

In order to obtain a better approximation of the solution set, it is possible to create a pavage composed of non-overlapping sub-boxes covering the initial box $[\mathbf{x}_0]$. Indeed, the solution set is rarely a box, so the over-approximation is often too pessimistic, especially for non-convex sets. The idea is to use smaller boxes to get closer to the researched set. Figure 2.5 gives an example on the same set $\mathbb{S}$ as for the contraction example.

Separators

When using contractors, we can obtain an approximation of the outer of the solution set $\mathbb{S}$. It means that we have the guarantee that none of the solutions are removed by the contractor. An idea to better characterize the solution set is to compute its complementary set using the complementary contractor, which is independent of the first one. This second contractor gives the approximation of the outer of the complementary solution. It is thus guaranteed that the result set contains every non-solutions (and often some solutions). It is used to remove points inside

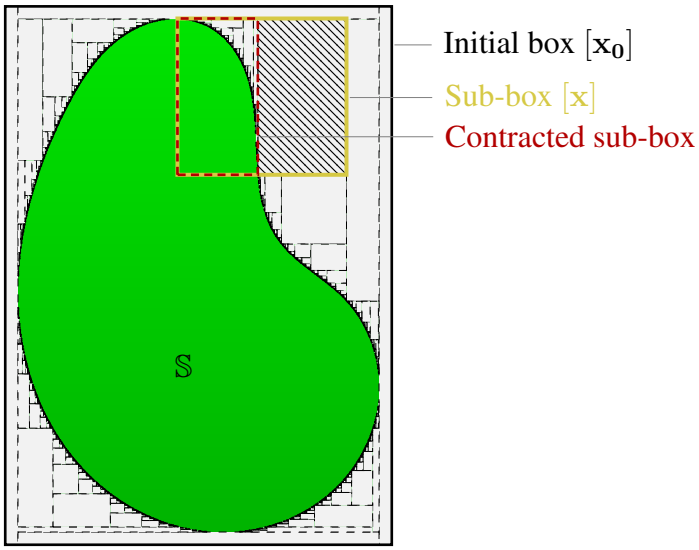


Figure 2.5: Paving of the box $[\mathbf{x_0}]$ with the contractor $\mathcal{C}$ associated with the set $\mathbb{S}$. The contraction of one sub-box is illustrated as example.

the solution set. In this way, it allows for computing an inner approximation of the solution set. This pair of contractors is then called a *separator* [74].

Concretely, a separator takes as input a box $[\mathbf{x}]$ and returns as output two sub-boxes $[\mathbf{x}_{\text{in}}]$ and $[\mathbf{x}_{\text{out}}]$ such that

$$([\mathbf{x}] \setminus [\mathbf{x}_{\text{in}}]) \subset \mathbb{S} \tag{2.32}$$

$$([\mathbf{x}] \setminus [\mathbf{x}_{\text{out}}]) \cap \mathbb{S} = \emptyset. \tag{2.33}$$

An example of an inner contractor is given in Figure 2.6 with a contractor $\mathcal{C}_{\text{in}}$ that is associated with a set $\mathbb{S}$. It is equivalent to the contractor $\mathcal{C}_{\text{out}}$ that is associated with the complementary set $\bar{\mathbb{S}}$. It means we have $\forall [\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{x}] \setminus \mathcal{C}([\mathbf{x}]) \subset \mathbb{S}$.

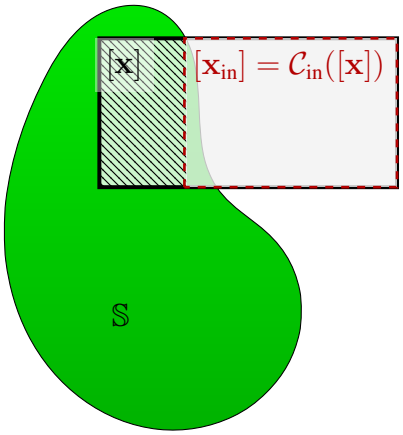


Figure 2.6: Example of an inner contraction of the box $[\mathbf{x}]$ with the contractor $\mathcal{C}_{in}$ associated with the set $\mathbb{S}$.

Besides, contractors (and by extension separators) can be easily combined to create more complex contractors. For instance, we may have different constraints for the same problem.

Then each contractor associated with each of these constraints can be defined. A combination of these contractors is made of unions and intersections of these contractors. A union (respectively an intersection) of contractors is the union (intersection) of the contracted boxes.

2.3.3 Combining sets

Set inversion problem

In many situations, not only in robotics, we have access to the effects, and we want to characterize the cause that produced them. Indeed, the effects can often be measured with sensors. More formally, it amounts to find the preimage set $\mathbb{X}$ knowing the image set $\mathbb{Y}$ of a function f such that $f(\mathbb{X}) = \mathbb{Y}$. The researched set is then $X = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\}$. For example, CSPs belong to set inversion problems: the solution set is a preimage of a function representing the constraints. So interval analysis is particularly adapted to solve this kind of problem, even with strong non-linearities, as in [111], where the localization of a robot in a known structured environment is dealt with. Using separators and an algorithm of set inversion, outer and inner approximations wrapping the researched set are computed.

The Set Inversion Via Interval Analysis (SIVIA) algorithm [76] fulfills these tasks: it returns subpavings enclosing the set to be characterized, and an indeterminate subpaving corresponding to the boundary between the two first subpavings. Its operation is similar to the one already described for the paving (in Section 2.3.2): it consists of bisecting boxes until each sub-boxes is either small enough to entirely belong to one approximation set (inner or outer), or too small to continue computations and then belong to the indeterminate set. This second case depends on a factor ϵ: the smaller, the closer the approximation. However, naturally, the number of calls to the separator depends on this factor, so choosing a smaller ϵ leads to a longer computation time. So all the sub-boxes too small to belong to one of the two approximation sets belong to the third indeterminate set: the boundary. This set is the boundary between the two others, and it contains all the values on which we have no information. In a context where we want a guarantee on the result (at the risk of adding pessimism), this boundary should be considered as we can not conclude on the belonging of its elements.

This approach may be compared to a gridding approach. Indeed, it is precisely the same reasoning, but instead of taking boxes, the gridding method proposes to take points covering the space and then to compute the belonging of these points. The more significant difference is the guarantee linked to interval analysis, which is not approached by a gridding algorithm due to the finite precision of machines.

Relaxed intersection

In the scope of estimation problems, sets most often have a physical signification. A set can represent the state (or part of the state) of a system, an estimation of its parameters, or some measurements. Depending on the strategy used, we can rapidly have a significant number of

sets to manage. This number is all the more critical when the sensors return a lot of valuable data. Combining these sets means using arithmetic and set operations to satisfy some constraints and thus reduce the uncertainties as much as possible. In other words, the problem amounts to characterizing a physical value (a state variable or a model parameter) from different data (coming from sensors or constraints). These data represent the most often partial pieces of knowledge of the researched set, sullied with uncertainty, hence the need to combine them. They are represented by sets $\mathbb{Z}_i$ that are called *granules* to underline the fact that they represent granules of knowledge.

When these granules are intervals, we have seen that we can use contractors to combine them and compute the researched set. A restriction that comes relatively quickly with this approach is that implemented constraints are assumed to be undeniably and always true. When inputting data from sensors, there are often a few *outliers*, that is, aberrant data. Only one occurrence of such an outlier and the associated constraint becomes incompatible with the other measurements. When using contractors, it often results in a contraction to the empty set, which is an undesired situation.

To handle this pitfall and robustify the approach, an algorithm has been developped to compute the *relaxed intersection*, or *q-intersection* [75]. This operator is denoted by $\bigcap^{\{q\}}$, and is such that for the granules $\mathbb{Z}_i, i \in [1, m]$, and $q \in [0, m]$ the set

$$\mathbb{X} = \bigcap_{i=1}^{m}{}^{\{q\}} \mathbb{Z}_i \tag{2.34}$$

is the set of all $\mathbf{x}$ that belong to all the $\mathbb{Z}_i$ except q of them. An illustrative example is given Figure 2.7 with 5 sets including 2 outliers. A classical intersection would have resulted in an empty set, whereas a 2- or 3-intersection, respectively painted red and yellow gives exploitable information.

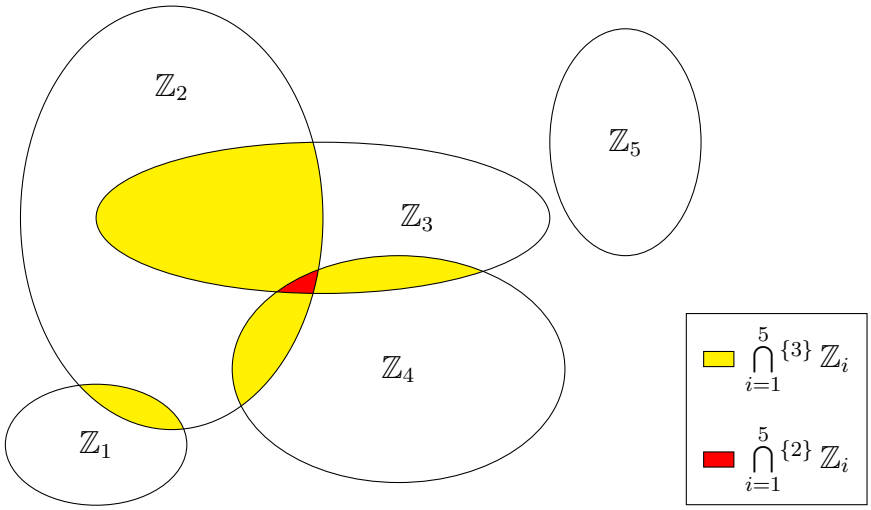


Figure 2.7: Examples of relaxed intersections on 5 sets including 2 outliers: $\mathbb{Z}_1$ and $\mathbb{Z}_5$.

We can notice the three following trivial cases that reduce the actual scope of q to $[1, m-2]$:

- if $q = 0$, then it amounts to a classical intersection;
- if $q = m - 1$, then it amounts to a classical union;
- if $q = m$, then it amounts to the universal set.

So the relaxed intersection allows further robustness to outliers, assuming that the number of outliers is known. Indeed, the setting of the parameter q is not always obvious and can be a complex task. To handle the latter issue, one can compute the relaxed intersection for different values of q with the idea to find the smallest q such that the q-intersection is not empty. This idea has already been studied, and estimators have been developed as the Outlier Minimal Number Estimator (OMNE) [91], or the guaranteed version Guaranteed Outlier Minimal Number Estimator (GOMNE) applied in localization [78].

Example 2.3.2. To illustrate the approach, suppose that we have four ring-shaped granules, $\mathbb{Z}_1, \mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_4$, corresponding to the position for a robot consistent with some measured distances d_j to different landmarks $\mathbf{m}(j) = (m_1(j), m_2(j))$. More precisely, we have

$$\mathbb{Z}_j = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \sqrt{(x_1 - m_1(j))^2 + (x_2 - m_2(j))^2} \in [d_j] \right\}, \tag{2.35}$$

where the intervals $[d_j]$ and the landmark coordinates $\mathbf{m}(j)$ are given by the Table 2.1.

j	1	2	3	4
$[d_j]$	$[2.2, 4.2]$	$[4.4, 6.4]$	$[7.1, 9.1]$	$[4.1, 6.1]$
$\mathbf{m}(j)$	$(-1, 3)$	$(5, 2)$	$(8, -1)$	$(1, -5)$

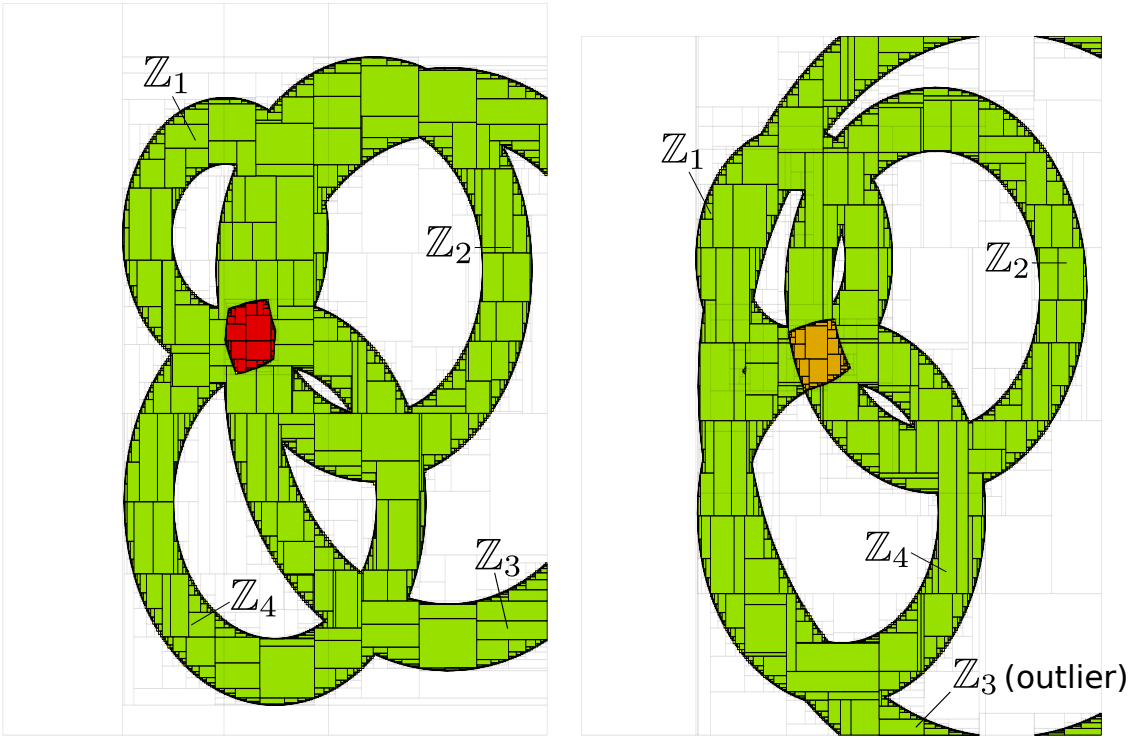
Table 2.1: Measured distances and coordinates of the landmarks.

We associate a separator for representing each granule. Then we can easily combine these granules by characterizing their union, intersection, or relaxed intersection. The results of these operations are represented in Figure 2.8. In both figures, the union of the rings is represented in green. On the left (Figure 2.8a), the intersection is painted red and is not empty as there is no outlier. On the right (Figure 2.8b), the granule $\mathbb{Z}_3$ is an outlier: the interval $[d_3]$ is equal to $[11.1, 13.1]$, which does not include the actual range to the third landmark. Then, the intersection of all the granules would be the empty set. Thus, the relaxed 1-intersection is computed and depicted in orange: the outlier has been rejected.

Figure 2.8 has been obtained by using the SIVIA algorithm. The visible boxes are due to the interval method, which bisects and tests interval values for x_1 and x_2.

2.4 Fuzzy logic

Fuzzy logic is a branch of fuzzy mathematics where a logical variable, instead of taking values only in the set $\{0, 1\}$ (as a boolean variable: $\{\textit{false}, \textit{true}\}$), can take any values in the interval



(a) without outlier (b) with one outlier ($\mathbb{Z}_3$)

Figure 2.8: Representation of different combinations of the four ring-shaped granules, with no outlier (left) or one outlier (right).

$[0, 1]$. It means that a logical variable is not binary anymore and can be *partially true*. Fuzzy logic had been studied since the 1920s as *infinite-valued logic*, and then had been widely studied primarily by Lotfi Zadeh since 1965 [168, 167].

2.4.1 Representing uncertainties

Uncertainties in estimation problems

Localization problems, and more generally, estimation problems, consist in computing some values such that they are the closest as possible to the actual value they represent. Concretely, when considering a system, we have access to the outputs of this system resulting from measurements, and we also generally have a model that gives its dynamics. Then the objective is to estimate either some parameters intrinsic to the system or state variables like the position.

Since then, several questions can be raised. On the one hand, the measurements provided by the sensors are the basis for working. Consequently, their interpretation is crucial for the following. They contain uncertainties and, sometimes, can even be outliers, *i.e.* completely wrong. How to represent these uncertainties, how to properly reject the outliers, how to divide the influence of each data on the result; all are issues for which different approaches exist.

On the other hand, the result of an estimation algorithm should precisely represent its uncertainty. Thus, some questions deserve to be studied, especially about the balance between

reliability and precision. Indeed, depending on the application of the result, two extremes can be envisioned. Either the result should be imperatively guaranteed. In that case, the lesser uncertainty should be taken into account, leading to a potentially unprecise result but totally reliable. Otherwise, the result is directly used to take a concrete decision, and a choice must be made: only one value is kept. The most probable or possible value is taken, but there is no more guarantee on it.

In order to address all these issues, several approaches exist with their own strength. However, when uncertainties are significant and when outliers occur frequently, *fuzzy logic* seems to suit particularly. This chapter starts with a small comparison of the different approaches to handle uncertainties and introduces the fuzzy set theory. Tools to deal with fuzzy sets are presented with some arithmetics.

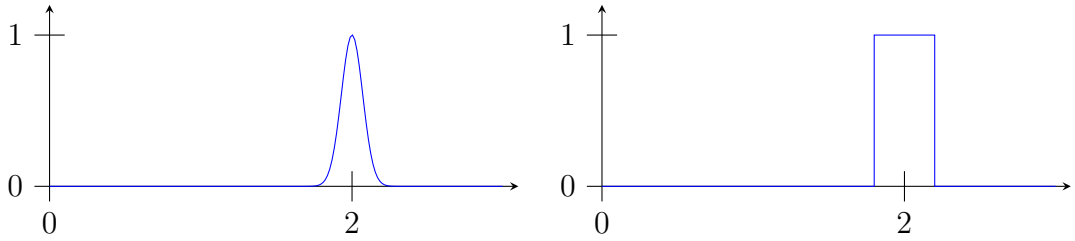
Approaches overview

When dealing with real robots, or more generally real systems, we need to handle uncertainties. These uncertainties come from different sources: external perturbations as weather parameters (such as wind and temperature), environment as uneven ground with unknown grip, or internal perturbations as sensors sensibility or motors efficiency. Well-representing uncertainties allow to finer understand the system and its dynamics, and thus better controlling or observing it.

The mainstream approaches are based on probability theory [151], with many efficient probabilistic estimators [162, 82] to characterize the system state, even in non-linear context as in [81]. However, these methods require a complete representation of the errors, and this is rarely possible. Hence some assumptions are made, more or less realistic, to define and characterize the posterior density function of the parameters to be estimated. For instance, the dependence between errors deeply complexifies probabilistic representation and thus is often ignored. Although results are often the expected ones, especially when data abound, they can lack integrity; the computed confidence region is then over-optimistic and even may be far from the actual system's state. When expecting some reliability and guarantee on the results, we can consider moving towards other approaches, more flexible concerning the strict requirements on the statistical properties of the manipulated data that are not always verified in practice.

In previous section, set membership methods have been presented with the interval analysis tools. It is particularly adapted when the representation of uncertainties and the correlations between them are unknown [56, 122, 110]. Instead of considering a density function, the approach consists in considering intervals [115] to represent uncertainties. It benefits from contractor programming [23, 133] to characterize the set of all parameters that are consistent with all the data. The corresponding interval estimators have been shown to be very reliable on many different types of localization problems [31, 96, 135, 38, 131]. It can address insufficient data and allow more flexible manipulations in complex and non-linear computations by avoiding strict statistical requirements, [90, 1, 77, 60, 30].

Nevertheless, it can be seen as less specific than probabilistic methods. Figure 2.9 shows different representations for the same measurement with the two different approaches. It illustrates the fact that interval representation lacks specificity. Indeed, the vertical dimension is not used at all. Either a point belongs to the interval or not, it is binary. So the probabilistic approach has this advantage: a point can have a probability to belong to the set within $[0, 1]$. To remedy this lack of specificity of set membership methods, we can envision: fuzzy sets [49], possibility theory [47] or belief theory [36, 37]. They all add this vertical dimension, allowing them to manage the distribution of the variables. These distributions are *possibility distributions*, contrary to probability distribution with the probabilistic approach. So the objective is not limited to extend the set membership approach to probabilistic context as in [73]: the convenience of fuzzy sets combined with tools from interval analysis can handle various contexts.



(a) Using probabilistic representation (b) Using interval representation

Figure 2.9: Example of representations for the measurement of a distance of 2 meters.

These conceivable approaches, fuzzy sets, possibility theory or belief theory, mainly been used when the variables are discrete, are particularly suited to deal with estimation problems where the uncertainty cannot be completely modeled. Moreover, when the variables are continuous, as vectors of $\mathbb{R}^n$, interval methods can still be used with a graduality: see for instance, [116] in the context of belief functions, [14, 16, 15] for fuzzy estimation, [46] in the context of possibility theory or [117, 43] when combined with probabilistic methods.

2.4.2 Fuzzy sets

Definitions

A *fuzzy set* is a set such that an element $\mathbf{x}$ can belong to this set only partially. We thus introduce the concept of *grade of membership*, or *degree of membership*, which is a real number of the unit interval $[0, 1]$. Then, when $\mathbf{x}$ is not included (at all) in the fuzzy set, its grade is equal to 0, when $\mathbf{x}$ is (fully) included in the set, its grade is equal to 1, and $\mathbf{x}$ is said to be partially included in the set when its grade is in $]0, 1[$.

From now on, we differentiate fuzzy sets with *crisp sets* that is the name given to classical sets. A crisp set is a particular case of a fuzzy set, with grades of membership only equals to 0 or 1 (respectively, non-belonging and belonging). To define properly such a set, we need to extend

the definition of a characteristic function. As a recall, for a crisp set $\mathbb{X}$ included in a reference set (or universal set) $\mathbf{\Omega}$, a membership function $\chi_{\mathbb{X}}$ is classically defined as follow:

$$\begin{aligned} \chi_{\mathbb{X}} : \mathbf{\Omega} &\to \{0,1\} \\ \mathbf{x} &\mapsto \begin{cases} 1 & \text{if } \mathbf{x} \in \mathbb{X} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.36}$$

Similarly, for a fuzzy set, we define the membership function $\mu_{\mathbb{X}} : \mathbf{\Omega} \to [0,1]$ which returns the grade of membership of its input, that is, any real between 0 and 1.

An α-*cut* is the set defined by a threshold $\alpha \in]0,1]$ and such that

$$\mathbb{X}_\alpha = \{\mathbf{x} \in \mathbf{\Omega} \mid \mu_{\mathbb{X}}(\mathbf{x}) \geq \alpha\}. \tag{2.37}$$

We also define the set $\text{core}(\mathbb{X}) = \mathbb{X}_1$ as the biggest set such that

$$\forall \mathbf{x} \in \text{core}(\mathbb{X}), \quad \mu_{\mathbb{X}}(\mathbf{x}) = 1 \tag{2.38}$$

and the set $\text{support}(\mathbb{X}) = \mathbb{X}_{0^+}$ as the biggest set such that

$$\forall \mathbf{x} \in \text{support}(\mathbb{X}), \quad \mu_{\mathbb{X}}(\mathbf{x}) > 0. \tag{2.39}$$

Figure 2.10 illustrates these notions on a one-dimensional fuzzy set.

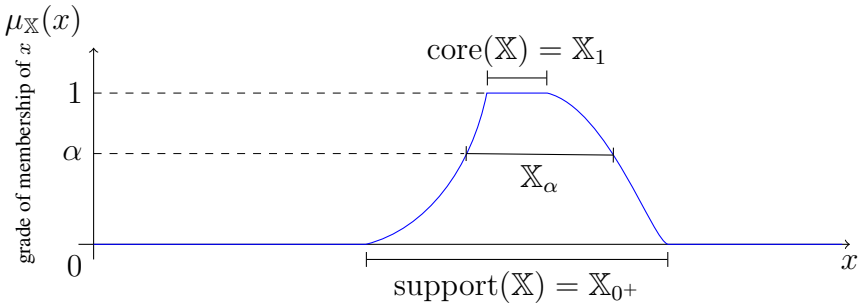


Figure 2.10: Representation of the fuzzy membership function of a one-dimensional fuzzy set.

So a fuzzy set $\mathbb{X}$ is defined by its μ-function $\mu_{\mathbb{X}}$ with the following "nested" property. Note that we consider only convex fuzzy sets.

$$\forall (\alpha_1, \alpha_2) \in [0,1]^2, \quad \alpha_1 < \alpha_2 \implies \mathbb{X}_{\alpha_1} \supseteq \mathbb{X}_{\alpha_2}. \tag{2.40}$$

Especially:

$$\text{core}(\mathbb{X}) \subseteq \text{support}(\mathbb{X}). \tag{2.41}$$

Note that we have the following proposition.

Proposition 2.4.1. *Let $\mathbb{X}$ be a fuzzy set defined by its μ-function $\mu_{\mathbb{X}}$. Then, we have*

$$\text{core}(\mathbb{X}) = \text{support}(\mathbb{X}) \iff \forall \alpha \in]0,1], \mathbb{X}_\alpha = \text{core}(\mathbb{X}) \iff \mathbb{X} \textit{ is a crisp set.}$$

Proof. $\Longrightarrow$: Assume that core(A) = support(A). Fix $\alpha \in]0,1]$. We have $0 < \alpha \leq 1$ and then $(A)_1 \subset (A)_\alpha \subset (A)_0$. Yet $(A)_1 = (A)_0 = \text{core}(A)$. So $(A)_\alpha = \text{core}(A)$.
$\Longleftarrow$: Assume that $\forall \alpha \in]0,1]$ $(A)_\alpha = \text{core}(A)$. Especially, $(A)_{0^+} = \text{core}(A)$. □

So intervals are particular cases of fuzzy sets. Every α-cuts are all equals. The fuzzy membership function of an interval is represented in Figure 2.11.

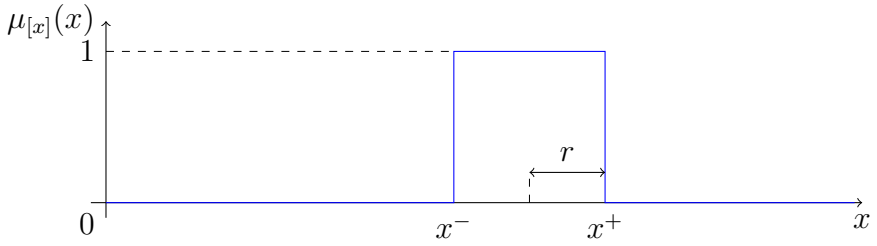


Figure 2.11: Representation of the fuzzy membership function of a classical interval $[x]$, or crisp set, with a radius of r.

Example 2.4.1. To give a first intuition of the interest of such fuzzy sets, let take a very simple example. Consider the harvest of apples in an orchard. Let take a fuzzy set representing the number of red apples harvested among all the apples. It is a fuzzy set because the notion of "red" for apple is fuzzy, and we choose here to accept that an apple can be only 50% red, for example. So we can represent the μ-function of this fuzzy set, as in Figure 2.12.

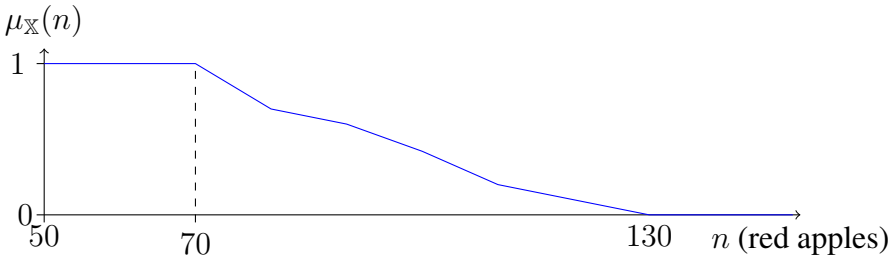


Figure 2.12: Fuzzy set representing the number of "red" apples harvested.

This fuzzy set indicates that 70 apples are undeniably red. However, there are 60 $(130 - 70)$ other apples that are more or less red and whose "more or less", which is a fuzzy quantifier, is represented by the grade value. Naturally, all this information contained in the fuzzy set could not have been represented by a probabilistic approach neither by an interval.

Operations on fuzzy sets

There are three basic operations on crisp sets: the complement, the intersection, and the union. These operations can be extended to fuzzy sets by the *standard fuzzy set operations*[1], presented in [48, 87] among others.

[1] As there are several ways to generalize these operations, the word *standard* denotes the usual way. In the following, as only standard operations are considered, the word standard will be omitted.

Let $\mathbb{X}$ and $\mathbb{Y}$ be two fuzzy subsets of the reference set $\mathbf{\Omega}$. Their associated fuzzy membership functions are denoted by $\mu_{\mathbb{X}}$ and $\mu_{\mathbb{Y}}$ respectively. Let $\mathbf{x}$ be an element of $\mathbf{\Omega}$. The complement of $\mathbb{X}$ is denoted by $\bar{\mathbb{X}}$, and its membership function is defined by

$$\mu_{\bar{\mathbb{X}}}(\mathbf{x}) = 1 - \mu_{\mathbb{X}}(\mathbf{x}) \qquad \text{(complement)} \tag{2.42}$$

The intersection and the union of $\mathbb{X}$ and $\mathbb{Y}$ are denoted by $\mathbb{X} \cap \mathbb{Y}$ and $\mathbb{X} \cup \mathbb{Y}$ respectively, and their membership functions are defined by

$$\mu_{\mathbb{X}\cap\mathbb{Y}}(\mathbf{x}) = \min\left(\mu_{\mathbb{X}}(\mathbf{x}), \mu_{\mathbb{Y}}(\mathbf{x})\right) \qquad \text{(intersection)} \tag{2.43}$$

$$\mu_{\mathbb{X}\cup\mathbb{Y}}(\mathbf{x}) = \max\left(\mu_{\mathbb{X}}(\mathbf{x}), \mu_{\mathbb{Y}}(\mathbf{x})\right) \qquad \text{(union)} \tag{2.44}$$

Note that we can easily check the consistency of these definitions on crisp sets. Assume that codomains of $\mu_{\mathbb{X}}$ and $\mu_{\mathbb{Y}}$ are restricted to the set $\{0, 1\}$. Thus $\mathbb{X}$ and $\mathbb{Y}$ are crisp sets, as an element $\mathbf{x}$ either belongs or not to these sets. There is no more fuzzy consideration. Table 2.2 shows that the definitions of each fuzzy set operations coincide with the truth table of crisp set operations, associating the digit 0 to *false* and 1 to *true*.

$\mu_{\mathbb{X}}(\mathbf{x})$	$\mu_{\mathbb{Y}}(\mathbf{x})$	$\mu_{\bar{\mathbb{X}}}(\mathbf{x})$	$\mu_{\mathbb{X}\cap\mathbb{Y}}(\mathbf{x})$	$\mu_{\mathbb{X}\cup\mathbb{Y}}(\mathbf{x})$
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

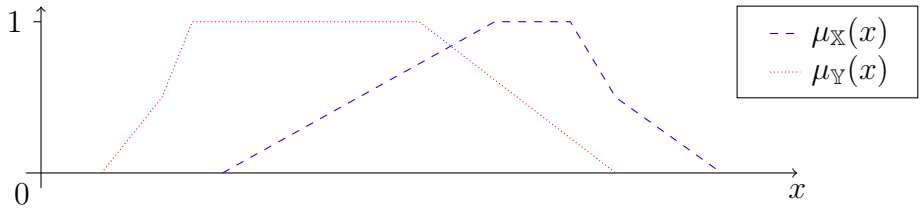
Table 2.2: Values of the μ-function for the different fuzzy set operations on two crisp sets. The results are the same as for crisp set operations.

Example 2.4.2. Figure 2.13 illustrates these operations on two given fuzzy sets $\mathbb{X}$ and $\mathbb{Y}$. The last example is a combination of two of these operations.

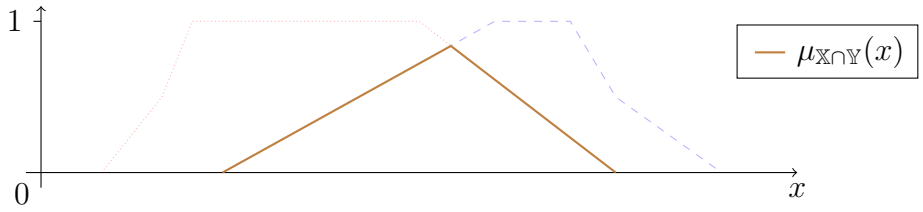
Remark. The union and complementary operations can define non-convex fuzzy sets, as in the two last examples. To keep the convexity through these operations, we can extend the square union ($\sqcup$) as defined on intervals, and use the same approach with the complementary operation.

Arithmetic on fuzzy sets

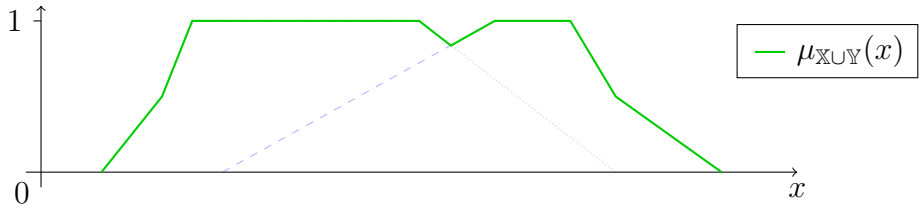
As fuzzy sets can represent a value or a vector of values, as it was the case for intervals, we want to define some arithmetical operations. For example, what if we want to compute the sum of two fuzzy sets, like to obtain the fuzzy set representing the number of red apples plus the number of green apples. Different approaches are described in [49, 83, 67]. The basic concept is the following.



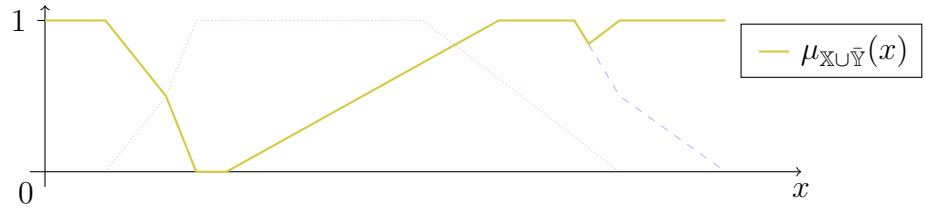
(a) Representation of the membership functions of the one-dimensional fuzzy sets $\mathbb{X}$ and $\mathbb{Y}$.



(b) Representation of the membership function of the intersection $\mathbb{X}\cap\mathbb{Y}$. It is the minimum of the two membership functions: $\forall x \in \Omega, \mu_{\mathbb{X}\cap\mathbb{Y}}(x) = \min\left(\mu_{\mathbb{X}}(x), \mu_{\mathbb{Y}}(x)\right)$.



(c) Representation of the membership function of the union $\mathbb{X}\cup\mathbb{Y}$. It is the maximum of the two membership functions: $\forall x \in \Omega, \mu_{\mathbb{X}\cup\mathbb{Y}}(x) = \max\left(\mu_{\mathbb{X}}(x), \mu_{\mathbb{Y}}(x)\right)$.



(d) Representation of the membership function of the fuzzy set $\mu_{\mathbb{X}\cup\bar{\mathbb{Y}}}(x)$. It is a combination of two operations: $\forall x \in \Omega, \mu_{\mathbb{X}\cup\bar{\mathbb{Y}}}(x) = \max\left(\mu_{\mathbb{X}}(x), 1 - \mu_{\mathbb{Y}}(x)\right)$.

Figure 2.13: Example of the operations on two fuzzy sets.

Set $\mathbf{x} \in \boldsymbol{\Omega}$. To compute the fuzzy set $\mathbb{Y}$ resulting of the binary operation (denoted by $\Diamond$) of two fuzzy sets $\mathbb{X}_1, \mathbb{X}_2$, its membership function is defined as

$$\forall (\mathbf{x_1}, \mathbf{x_2}) \in \boldsymbol{\Omega}^2, \quad \mu_{\mathbb{Y}}(\mathbf{x}) = \sup_{\mathbf{x}=\mathbf{x_1}\Diamond\mathbf{x_2}} \min\left(\mu_{\mathbb{X}_1}(\mathbf{x_1}), \mu_{\mathbb{X}_2}(\mathbf{x_2})\right). \tag{2.45}$$

However, this definition cannot be directly implemented, as there are generally an infinite number of combinations of $\mathbf{x}_1, \mathbf{x}_2$ such that $\mathbf{x}_1 \Diamond \mathbf{x}_2 = \mathbf{x}$. One approach to address this implementation issue is actually to use the interval analysis, as studied in [83]. Indeed, a fuzzy set can be viewed as a sequence of α-cuts:

$$\forall \mathbf{x} \in \boldsymbol{\Omega}, \quad \mu_{\mathbb{X}}(\mathbf{x}) = \sup_{\alpha\in[0,1]} \alpha \cdot \mu_{\mathbb{X}_\alpha}(\mathbf{x}), \tag{2.46}$$

where $\mu_{\mathbb{X}_\alpha}$ is the membership function of the α-cut of the set $\mathbb{X}$. This function thus represents a crisp set: it is equal to the characteristic function of $\mathbb{X}_\alpha$. With the convexity of fuzzy sets assumption, $\mathbb{X}_\alpha$ is then an interval. A fuzzy set is then fully represented by its successive α-cut, and the arithmetic can be implemented using the one from interval analysis on each α-cut. Naturally, it works only when the number of α-cuts is finite. When there is an infinite number of α-cuts, which is the general case since $\alpha \in [0, 1]$, we can still achieve the representation of a fuzzy set by subdividing the segment $[0, 1]$ in m intervals and carrying the computations on each of these intervals.

That said, in the general case, we do not need to compute every α-cuts anyhow. It can serve to provide a graphical representation of the studied set, but most of the time, for estimation problems, only a few α-cuts are used at the end.

Score function

The relaxed intersection and the estimators as OMNE allow a parameters estimation particularly robust to outliers. However, the limits of these tools can be reached, and a more general estimator is sometimes needed. This is the case, for example, when the outlier occurrences are interdependent or when the combination of the sets requires weighting them differently. For instance, granules can come from different sensors, and we may have more trust in one sensor than the other ones. So we may want to give more weight to data from this sensor.

The idea of the approach is to use the fuzzy set theory to combine the granules of knowledge. The combination is described by a *score function*, which allows great flexibility in this combination.

Formally, a score function σ is a mapping from $\{0, 1\}^m$ to $[0, 1]$ with the following properties:

$$\begin{aligned}
&\text{(i)} \quad \sigma(0, \dots, 0) = 0; \\
&\text{(ii)} \quad \sigma(1, \dots, 1) = 1; \\
&\text{(iii)} \quad \forall i \in [0, m], \quad a_i \leq b_i \implies \sigma(a_1, \dots, a_m) \leq \sigma(b_1, \dots, b_m) \quad \text{(monotonicity).}
\end{aligned} \tag{2.47}$$

The score function takes as inputs the memberships of a vector to the granules, and returns the (fuzzy) belonging of this vector to a fuzzy set. In other words, this function becomes the frame of the membership function of the wanted fuzzy set $\mathbb{X}$ from the granules $\mathbb{Z}_1, \dots, \mathbb{Z}_m$:

$$\begin{aligned}
\mu_{\mathbb{X}} : \boldsymbol{\Omega} &\to [0, 1] \\
\mathbf{x} &\mapsto \sigma\left(\zeta^1(\mathbf{x}), \dots, \zeta^m(\mathbf{x})\right),
\end{aligned} \tag{2.48}$$

with ζ^j the characteristic functions of the granule $\mathbb{Z}_j$: $\zeta^j(\mathbf{x}) = \chi_{\mathbb{Z}_j}(\mathbf{x})$ for $j \in [1, m]$. Note that μ is a piecewise constant function, so the number of α-cuts is finite and lower than 2^m, the number of values that can be taken by the vector $(\zeta^1(\mathbf{x}), \dots, \zeta^m(\mathbf{x}))$.

Example 2.4.3. To illustrate the approach, let us take once again the four ring-shaped granules, $\mathbb{Z}_1, \mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_4$, corresponding to the position for a robot consistent with some measured distances d_j to different landmarks $\mathbf{m}(j) = (m_1(j), m_2(j))$ (see Example 2.3.2). More precisely, we still have

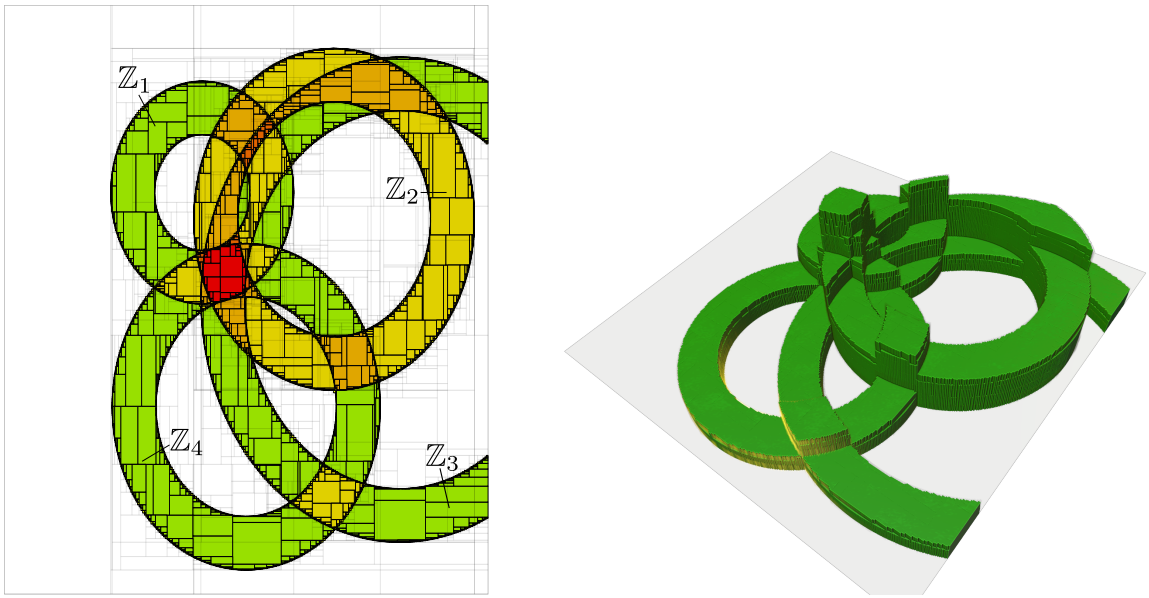
$$\mathbb{Z}_j = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \sqrt{(x_1 - m_1(j))^2 + (x_2 - m_2(j))^2} \in [d_j] \right\}, \tag{2.49}$$

where the intervals $[d_j]$ and the landmark coordinates $\mathbf{m}(j)$ are given by the Table 2.1.

We define the fuzzy set $\mathbb{X}$ by its set membership function

$$\mu_{\mathbb{X}}(\mathbf{x}) = \frac{\zeta^1(\mathbf{x}) + 2\zeta^2(\mathbf{x}) + \zeta^3(\mathbf{x}) + \zeta^4(\mathbf{x})}{5}, \tag{2.50}$$

where $\zeta^j(\mathbf{x})$ are the characteristic functions of the granules $\mathbb{Z}_j$. The expression for $\mu_{\mathbb{X}}(\mathbf{x})$ translates the fact that the position of the robot should satisfy the measured distance intervals such as the confidence or the reliability associated with $\mathbf{x} \in \mathbb{Z}_2$ is twice the reliability of $\mathbf{x} \in \mathbb{Z}_j, j \neq 2$. The fuzzy set $\mathbb{X}$ can be represented by its α-cuts $\mathbb{X}_\alpha$, $\alpha \in \{\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\}$ as illustrated by Figure 2.14.



(a) Top view with α going from green to red

(b) 3D view

Figure 2.14: Representation of the fuzzy set $\mathbb{X}$ with its vertical dimension. (a): The value of α goes from $\frac{1}{5}$ (green) to 1 (red). (b): 3D view of $\mathbb{X}$ in the (x_1, x_2, α)-space.

Figure 2.14 has been obtained by using the SIVIA algorithm that is described in Section 2.3.3. We observe that the function μ is made with plateaus due to the specific form of our function μ. The boxes that are visible in Figure 2.14, left, is due to the interval method, which bisects and tests interval values for x_1 and x_2.

The corresponding Python program can be downloaded and executed online at the following link:

https://replit.com/@TilletJ/Alpha-cut-characterization.

Finally, some links can be made with the evidence theory. Evidence theory, or belief theory, or Dempster-Shafer theory [141] can be viewed as an interconnection between probability theory, possibility theory, and fuzzy theory [102]. For instance, the score function we have defined could be interpreted as a combination function in the evidence theory in the case where we define every mass of the granules to 1. In our formalism, the masses can be directly set in the coefficients of the score function.

2.5 Conclusion

In this chapter, different theoretical tools from three mathematical fields have been introduced. Firstly, notions from non-linear control theory can be applied to a large range of systems. These notions will mainly be used in the next chapter, allowing to deal with controlling a robot towing a subsystem using the feedback linearization method.

Then, the interval analysis field has been presented, focusing on resolving Constraint Satisfaction Problem with contractors. Many problems can be formalized such that it amounts to a CSP, and thus be solved by an interval analysis method. This set-membership approach is applicable even when strong non-linearities are involved, and allow keeping a guarantee on the results in spite of uncertainties and outliers.

Fuzzy logic is the last mathematical tool studied in this chapter. It provides a formalism totally adapted to handle uncertainties and outliers in any form whatsoever. Thus, fuzzy sets can be seen as a generalization of intervals. α-cuts are intervals on whose all tools from interval analysis can be applied, and the vertical dimension of fuzzy sets allows to do without the strict binarity of intervals.

CHAPTER 3

Non-linear control with state constraints

Contents

3.1 Introduction

Nowadays, mobile robotics is mature enough to be used in a very large variety of contexts and environments. To give a quick overview, we can name examples from tiny robots, with microrobots primarily used in medical applications as in [33], to huge ones, as work class Remotely Operated underwater Vehicles (ROVs) as described in [28, Chapter 1]. We can also mention humanoid robots with NAOs from Aldebaran-Robotics [61], rovers such as *Discovery* [54] or autonomous sailboats like VAIMOS [94]. So the applications are multiple. But if we take a step back, plenty of these applications consist in embedding a payload, either composed by a packet to deliver, or a sensor to collect data or even both. Sometimes, there is no choice: this payload must be deported. Indeed, towing a packet can simplify the delivery, and some kinds of sensors are sensitive to the robot itself. The latter reason is the motivation for looking at these kinds of systems.

If we consider using a magnetometer, like in the research of buried wrecks, it requires to deport the sensor far from every source of perturbation. Actually, a magnetometer measures the variations in the magnetic field, and a mobile robot is composed of motors, batteries, and other ferromagnetic materials whose magnetometers are sensitive. This is the case with *Boatbot*, presented in Section 1.3.1.

In order to deal with such a context, the robotic system's design must obey this constraint. The solution of the deported sensor seems mechanically appropriate and straightforward. However, from an automatic point of view, it brings some new complexifications: the control of the system should make the sensor follow the desired trajectory, as this is the position of the sensor that is important in the context of a survey.

The purpose of this chapter is to apply tools from non-linear control theory in the scope of such a context. These tools have been introduced in Section 2.2.2. They are applied to a car towing a trailer, whose behavior is close to a deported vehicle. Thus, the design of a controller for a trailer-car system is proposed, knowing that it will be applied to a maritime system. The other advantage of considering this system as an application is that it is easy to model and thus simulate. Experiments on a real system can also be carried out more quickly.

Finally, the objective is to find a controller for this car to make the trailer follow a specific trajectory. Nevertheless, the safety of the system requires to be taken into account. Therefore, we want to validate the designed controller with respect to some state constraints defined to guarantee the system's safety. This is why we need to deal with *follow sets* which depict the safe areas for our robot with the dedicated controller.

This chapter begins with an introduction to concepts of controllability and flatness. They allow handling the problem of the control of the car-trailer system. A controller is then proposed and tested in simulation and on a real robot. Finally, the interval analysis is used to characterize the desired follow sets for the system's safety.

3.2 Controllability and flatness

Before applying the feedback linearization method and find a control law, we can check if the system is *differentially flat*. It is the non-linear equivalent of the controllability of linear dynamical systems [55]. Flatness theory brings valuable tools for this task, helping to understand better the physical meaning and links between the variables of the studied system. The interests are twofold. On the one hand, the flatness property assures that we can find a controller that stabilizes the system at a desired state. On the other hand, a flat system has a *flat output* which can be used to express the whole state analytically with respect to this output and its derivatives. This last point is essential for us as it allows the analysis of the controller. It will be used to express the follow sets.

3.2.1 Theory

Consider once again the non-linear system given by the general equations (2.1) and (2.2). The first step is to draw the *graph of differential delays*. This tool quickly shows the variables linked by a differential delay and those linked by an analytical non-differential relation. The objective is to understand the differential delays existing between the inputs and the outputs. It is crucial to obtain an intuition on the potential dynamics of the inputs on the outputs, *i.e.* the observed state of the system.

In this way, the relative degrees of each output can be easily read. Finally, the sum of these relative degrees gives an intuition on the system's controllability. It is in the same manner as the rank of the controllability matrix in a linear context. If this sum is lower than the dimension of the system, then a feedback-based linearization method leaves some state variables without any control. That is generally not wanted unless the system is stable and the result leads to correct behavior anyhow. When the sum is equal to the dimension of the system, we are sure that the system is controllable, and the output is then called a *flat output*.

This flat output corresponds to the output with the following additional property: the whole state of the system can be expressed with respect to this output and its successive derivatives. In other words, there exists a function ϕ such that $\phi(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(r)}) = \mathbf{x}$. Note that all the inputs can also be expressed with the same manner.

3.2.2 Cart example

Dubins' car model

In order to illustrate these notions, an example is now proposed with a robotic application. The cart model [45], also called the Dubins' car model, is a classic example of a mobile robotic system. The control of the heading of this system is proposed to see the application of the feedback linearization method. Figure 3.1 gives the physical correspondence of each variable.

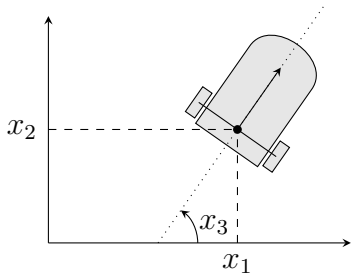


Figure 3.1: Model of the Dubins' car with the representation of state variables.

The system is described by the following state equations:

$$\begin{cases} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{pmatrix} = \begin{pmatrix} \cos(x_3) \\ \sin(x_3) \\ u \end{pmatrix} \\ y = x_3 \end{cases} \tag{3.1}$$

with (x_1, x_2) the position of the car, and x_3 its heading. The speed of the car is assumed to be constant and equal to 1 m.s^{-1}. The output is the heading, and the input acts directly on its derivative: $\dot{y} = \dot{x_3} = u$. The latter relation is the only one that would appear in the graph of differential delays. Nevertheless, the dimension of the system is 3. So it means that at most 2 variables are let without any control. Indeed, as the speed is stuck at the constant value 1, the position is not controllable, and only the heading x_3 is. So this is not a flat system. It is not possible to find back the whole state of the car (especially its position) with only the output x_3 and its derivatives. However, we can still find a controller for the heading using the feedback linearization method, as the position cannot make the heading unstable even if it is not controllable.

Now if we consider choosing the control

$$u = -x_3, \tag{3.2}$$

then we have

$$\dot{y} = -x_3 \tag{3.3}$$

and

$$\dot{y} + y = 0. \tag{3.4}$$

This last differential equation implies that $y = x_3$ will follow the dynamics given by the solution of the differential equation which is

$$y(t) = y(0) \cdot e^{-t}. \tag{3.5}$$

It means that this control law leads the car to follow the heading 0, as we have $y(t) \underset{t\to\infty}{\to} 0$.

More generally, let say we want the heading to follow a specific dynamics $\Psi(t)$ with Ψ a smooth and derivable function. So the objective is to find a controller $u(t)$ such that $Y(t) = y(t) - \Psi(t) \underset{t\to\infty}{\to} 0$.

We have

$$\dot{Y}(t) = u(t) - \dot{\Psi}(t) \tag{3.6}$$

and we want to find u such that

$$\begin{aligned} \dot{Y}(t) + Y(t) &= 0 \\ \underbrace{\dot{y}(t)}_{u(t)} - \dot{\Psi}(t) + y(t) - \Psi(t) &= 0. \end{aligned} \tag{3.7}$$

to make $Y(t)$ converges towards 0.

So the expression for the controller is

$$u(t) = -x_3(t) + \dot{\Psi}(t) + \Psi(t). \tag{3.8}$$

For instance, if we take

$$\Psi(t) = t + \cos(t), \tag{3.9}$$

then the controller is

$$u(t) = -x_3(t) - \sin(t) + \cos(t) + t + 1. \tag{3.10}$$

Moreover, the system converges exponentially towards the desired dynamics in a few seconds.

Control the speed

This example was very immediate, so we can further consider the same car with a controllable speed. The previous system was a SISO (Single-Input and Single-Output) system, and now we deal with a MIMO (Mulitple-Input and Multiple-Output) system. The state equations describing this new system are

$$\begin{cases} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{pmatrix} &= \begin{pmatrix} x_4\cos(x_3) \\ x_4\sin(x_3) \\ u_1 \\ u_2 \end{pmatrix} \\ \mathbf{y} &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \end{cases} \tag{3.11}$$

We introduce the new state variable x_4, which corresponds to the speed of the car. The input vector $\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ is now of dimension 2, as the output, which is now the (2D-)position of the car.

The graph of differential delays is drawn in Figure 3.2. The output happens to be a flat output: the sum of the relative degrees $(2 + 2)$ is equal to the dimension of the system (*i.e.* 4). The

system is therefore controllable, and it means that knowing the position and its derivatives, we are able to compute the whole system state (that is the heading and the speed, respectively the arc-tangent and the norm of the derivative of the position).

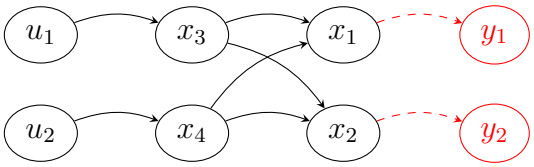


Figure 3.2: Graph of the differential delays for the new system. Dashed lines represent a non-differential relation between two nodes, and the red color is associated with the output.

When we study the output by computing its derivatives, we obtain

$$\dot{\mathbf{y}} = \begin{pmatrix} x_4 \cos(x_3) \\ x_4 \sin(x_3) \end{pmatrix} \tag{3.12}$$

$$\ddot{\mathbf{y}} = \begin{pmatrix} u_2 \cos(x_3) - u_1 x_4 \sin(x_3) \\ u_2 \sin(x_3) + u_1 x_4 \cos(x_3) \end{pmatrix} \tag{3.13}$$

$$= \underbrace{\begin{pmatrix} -x_4 \sin(x_3) & \cos(x_3) \\ x_4 \cos(x_3) & \sin(x_3) \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{3.14}$$

If we assume that $x_4 \neq 0$, that is, the speed is not null, then the matrix $\mathbf{A}$ is invertible. We can now make the change of variable

$$\mathbf{u} = \mathbf{A}^{-1} \cdot \mathbf{v}. \tag{3.15}$$

Then we have

$$\ddot{\mathbf{y}} = \mathbf{A} \cdot \mathbf{u} \overset{(3.15)}{=} \mathbf{A}\mathbf{A}^{-1} \cdot \mathbf{v} = \mathbf{v}, \tag{3.16}$$

with $\mathbf{v}$ the new input of the linearized system. Now standard linear techniques can be used to stabilize and control the system.

3.2.3 Implementation

Symbolic computation

In the previous example, we had to compute two successive Lie derivatives of the output. It has been done by hand in (3.12) and (3.13). This computation by hand of Lie derivatives was quite immediate but can quickly become intricate. It can make the feedback linearization method a long and complex task. The idea here is to present a means to compute these derivatives automatically, avoiding loss of time and possible human calculus errors. Classically, there are two main approaches to obtain derivatives: automatic differentiation and symbolic computation. Each one has its advantages and drawbacks. Automatic differentiation consists more or less of

computing approximations of the solution on some discretized points. See [126] for more details on this method.

We choose to use symbolic computation because we need to keep symbolic variables to compute the relative degrees of the outputs. In addition, we do not want to introduce approximations. Indeed, the feedback linearization method is sensitive to approximations, so we rather stay with the symbolic computation approach, although it can be slower. Another advantage of using symbolic computation is that it can solve the equation at the end, giving the analytical expression of the controller u. Every step is thus automatized, and the output of the symbolic computation can be directly implemented in the system. Finally, one last advantage of symbolic computation for our application is that the full symbolic expression can be used to compute some conditions on desired constraints. However, the latter point will be the object of Section 3.4.

Tools from differential geometry presented in Section 2.2.3 give the formalism to implement feedback linearization method into algorithms, using only symbolic computation [148]. For instance, the `SymPy` library [113] implements these tools for the Python language.

Example

We propose an example to illustrate how it can be helpful with the last studied system: the cart with speed input described by (3.11). Table 3.1 provides the Python program that computes the two inputs that lead to linearized feedback. The code is explained hereinbelow.

In a first while, we define every `SymPy` objects required to compute Lie derivatives for our system.

- **Lines 5 to 11:** the evolution function $\mathbf{f}$ of the system is defined using (3.11). The input is the state $\mathbf{X}$, composed by x_1, x_2, x_3, x_4, and the output is the derivative $\dot{\mathbf{X}}$.
- **Line 13:** we search to compute both inputs u_1 and u_2. We thus define the associated symbols to make them appear in the equations while not knowing their expressions. They are symbolic variables.
- **Line 15:** we define the manifold, which is the topological space in which we work. Its dimension is 4 as the number of state variables of our studied system.
- **Line 16:** on the defined manifold, we can now introduce a coordinate patch $\mathbf{P}$ which is required to establish a coordinate system.
- **Lines 17 to 18:** the coordinate system is constituted by the four state variables x_1 to x_4.
- **Line 19:** the variable $\mathbf{X}$ stores the vector of state variables.
- **Line 22:** the vector field associated with the evolution function is denoted by $\mathbf{f\_vf}$ and is the dynamics defined by the function $\mathbf{f}$ on the patch $\mathbf{P}$.

Then, we apply the feedback linearization method:

- **Line 24:** we define the arrays y_1 and y_2 to store the outputs and their (Lie) derivatives.
- **Line 25:** we check that the last computed derivative of y_1 is not dependant of the input (u_1 or u_2). Otherwise, the algorithm directly jumps to line 27.
- **Line 26:** we compute the next Lie derivative of y_1 and store it at the end of the array y_1. Then, the algorithm executes line 25 again.
- **Lines 27-28:** it is the same as lines 25-26, but for y_2.
- **Lines 29-30:** when these lines are executed, the arrays y_1, y_2 both contain the list of the successive derivatives of each output of the system, with the input involved in the last computed derivative. Thus, the length of these arrays is equal to the relative degree of each output.
- **Line 31:** we introduce two new symbolic variables v_1 and v_2 which will be the new inputs of our linearized system.
- **Line 32:** we ask `SymPy` to solve the system of two equations where u_1 and u_2 are the unknown variables.

The returned expressions for u_1 and u_2 are

$$\begin{cases} u_1 = \dfrac{-v_1 \sin(x_3) + v_2 \cos(x_3)}{x_4} \\ u_2 = v_1 \cos(x_3) + v_2 \sin(x_3) \end{cases} \tag{3.17}$$

which gives $\ddot{\mathbf{y}} = \mathbf{v}$. Then an expression for $\mathbf{v}$ can be straightforwardly found with usual linear control methods.

Link with interval analysis

The interest in using `SymPy` to compute Lie derivatives is undeniable. It is of precious help for applying the feedback linearization method. Nevertheless, the library can be used further: when building symbolic expressions, `SymPy` creates a *syntax tree* to store expressions. This syntax tree can be reclaimed to be used with other libraries. The goal is to separate the symbolic part, which corresponds to the formalization of the problem, and the computation part, which can be done with several approaches.

In our context, we may like to use interval analysis for the computation part and thus obtain guaranteed numerical results. It is possible by getting the syntax tree of the wanted expression and then injecting it into an interval library like `Ibex`. For instance, `Ibex` can take an expression of a constraint as input and define the associated contractor.

As an example, this approach is used in Section 3.4 to validate the controller.

```python
from sympy import Matrix, symbols, cos, sin, solve
from sympy.diffgeom import Manifold, Patch, CoordSystem, LieDerivative
from sympy.matrices import matrix_multiply_elementwise

# Evolution function
def f(X):
    x1, x2, x3, x4 = X
    return Matrix([x4*cos(x3),
                   x4*sin(x3),
                   u1,
                   u2])

u1, u2 = symbols('u_1 u_2') # command u

M = Manifold('M', 4) # State space of dimension 4
P = Patch('P', M)
coord = CoordSystem('coord', P, ['x_1', 'x_2', 'x_3', 'x_4'])
x1, x2, x3, x4 = coord.coord_functions() # state variables
X = Matrix([x1, x2, x3, x4]) # state vector

# vector field of the function f
f_vf = sum(matrix_multiply_elementwise(f(X), Matrix(coord.base_vectors())))

y1, y2 = [x1], [x2] # output y and its derivatives
while {u1, u2}.isdisjoint(y1[-1].atoms()):
    y1.append(LieDerivative(f_vf, y1[-1]))
while {u1, u2}.isdisjoint(y2[-1].atoms()):
    y2.append(LieDerivative(f_vf, y2[-1]))
print("Relative degree for y1:", len(y1))
print("Relative degree for y2:", len(y2))

v1, v2 = symbols('v_1 v_2') # command v for the linearized system
command = solve([y1[-1] - v1, y2[-1] - v2], (u1, u2))
# command contains the expressions of u1 and u2
# with respect to state variables, v1 and v2
```

Table 3.1: Example of a Python program using `SymPy` to find the linearized feedback of a system.

3.3 Application: trailer command

We propose here a concrete robotic application using all the notions that have been presented. In the context of finding buried wrecks, like the *Cordelière*'s one that was presented in Section 1.1.2, we would like to use magnetometers. Indeed, it is the only kind of sensor to our knowledge that is able to detect a relatively deep (a few meters) buried object in the sea, assuming that it contains ferromagnetic parts. The drawback of this sensor is that it is sensitive to the robot itself, as the robot contains ferromagnetic materials and often perturbates the local magnetic field. A simple solution to avoid these perturbations is to move the magnetometer a little further from the robot and tow it with a rope. The goal is to control such a system in order to make the magnetometer follow the desired path.

3.3.1 Robotic system for searching of wrecks

Figure 3.3 illustrates the system with the deported magnetometer.

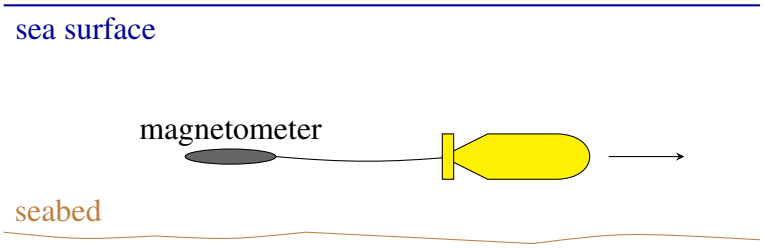


Figure 3.3: Schema of a robotic system using a deported magnetometer to search wrecks.

Therefore, the control law becomes a little more complex. Indeed, the objective is to realize a magnetic map of the researched area. So the idea is to make the sensor scan the whole area and associate data to their localization. However, the trajectory of the towed sensor can be relatively different from the trajectory of the robot itself, especially when the distance between the two is significant and when the robot makes turns. In that way, we want to find a control law that makes the towed system follow a specific trajectory, knowing that we can only act on the towing robot.

To avoid complexifying too much the problem, we choose to make the following assumption: the rope between the two vehicles is always taut. Therefore, we avoid modelizing the complex behavior of a rope in the water. This assumption becomes wrong as soon as the robot makes too tight turns or when it slows down too quickly. Nevertheless, both of these conditions can be avoided. We can choose a scanning trajectory without too tight turns, and the objective is to have a constant speed for the robot in order to obtain the desired frequency of measurements. This assumption makes possible to identify the system to a car-trailer system: the car tows the trailer with a rigid link. There is another simplification with this identification: the problem is considered as a two-dimensional one, although there is a third dimension in the context of the underwater systems. However, the problem of regulating the magnetometer to the correct depth

can be treated independently to the horizontal one and is thus not addressed in this document, even if the same approach could be used.

3.3.2 Formalism of the car-trailer system

Let consider the car-trailer system. It is described by these equations:

$$\begin{cases} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{pmatrix} = \underbrace{\begin{pmatrix} x_5 \cos(x_3) \\ x_5 \sin(x_3) \\ 0 \\ x_5 \sin(x_3 - x_4) \\ 0 \end{pmatrix}}_{\mathbf{f(x)}} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ u_1 \\ 0 \\ u_2 \end{pmatrix}}_{\mathbf{g(x)\cdot u}} \\ \mathbf{y} = \begin{pmatrix} x_1 - \cos x_4 \\ x_2 - \sin x_4 \end{pmatrix} \end{cases} \tag{3.18}$$

The variables' correspondences are shown in Figure 3.4. The chosen output is the heading of the trailer center's position, which can be computed from state variables.

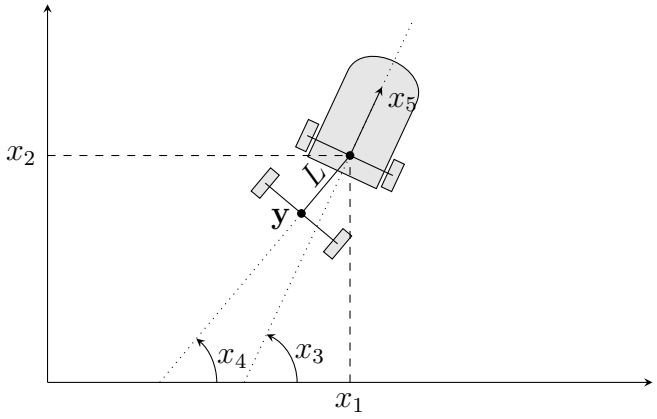


Figure 3.4: Model of the car-trailer system with a representation of state variables. The variables x_1, x_2, x_3 have the same meaning as for the Dubins' car. The heading of the trailer is denoted by x_4, and the speed of the car is the variable x_5.

3.3.3 Flattened feedback

Graph of differential delays

As presented in Section 3.2, the first step for applying the feedback linearization method to a system is to draw its graph of differential delays. It has been done in Figure 3.5. The relative degrees of the two outputs can be read by counting the (minimal) number of continuous arcs separating them from the inputs. Here it is 2 for both outputs. The sum of the relative degrees

is thus equal to 4. However, the system is of dimension 5. So we cannot conclude on the controllability of the system.

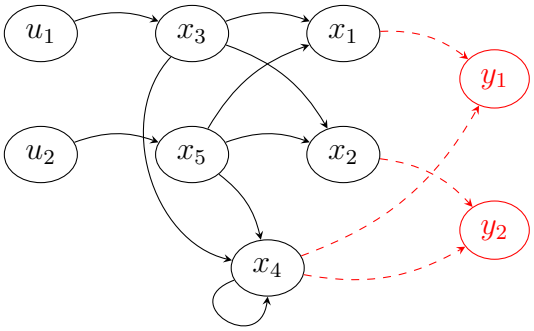


Figure 3.5: Graph of the differential delays of the car-trailer system, described by (3.18).

A feedback-based linearization method would leave one state variable without any control. If we are lucky, this floating state variable is stable, and the resulting behavior is correct. Now, if we push the method up to the simulation, we observe that for our system, the floating state variable is unstable. This instability makes the approach inappropriate.

New modelization

However, it is possible to change the system's modelization to find flattened feedback for the robot. The following theorem, illustrated by Figure 3.6, provides this flattened feedback. This modelization is taken from [137] where further details can be found. In the paper, the trailer is controlled to follow a specific trajectory. The difference here is that we want the output to follow particular dynamics. The proposed modelization means that the sum of the relative degrees corresponds to the dimension of the system.

Theorem 3.3.1. *Consider the controller*

$$\begin{aligned} \dot{v}_1 &= a_1 \\ \mathbf{u} &= \mathbf{A}^{-1}(\mathbf{x}) \cdot \left(\begin{pmatrix} v_1 \\ a_2 \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right), \end{aligned} \tag{3.19}$$

where

$$\mathbf{A}(\mathbf{x}) = \begin{pmatrix} -x_5 \sin(x_3 - x_4) & \cos(x_3 - x_4) \\ x_5 \cos(x_3 - x_4) & \sin(x_3 - x_4) \end{pmatrix}$$

and

$$\mathbf{b}(\mathbf{x}) = \begin{pmatrix} x_5^2 \sin^2(x_3 - x_4) \\ -x_5^2 \sin(x_3 - x_4) \cos(x_3 - x_4) \end{pmatrix}.$$

In the new coordinate system given by

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} = \underbrace{\begin{pmatrix} x_1 - \cos x_4 \\ x_2 - \sin x_4 \\ x_5 \cos(x_3 - x_4) \\ v_1 \\ x_4 \\ x_5 \sin(x_3 - x_4) \end{pmatrix}}_{\boldsymbol{\varphi}(\mathbf{x}, v_1)} \tag{3.20}$$

we get the closed-loop system:

$$\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \\ \dot{z}_5 \\ \dot{z}_6 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} z_3 \cos z_5 \\ z_3 \sin z_5 \\ z_4 \\ a_1 \\ z_6 \\ a_2 \\ z_1 \\ z_2 \end{pmatrix} = \mathbf{f}_z(\mathbf{z}) + \mathbf{g}_z(\mathbf{z}) \cdot \mathbf{a} \qquad = \mathbf{h}_z(\mathbf{z}). \tag{3.21}$$

Remark. This theorem suggests a better coordinate system to represent the state: (y_1, y_2) is the center of the trailer, z_5 the heading of the trailer, (z_3, z_6) the speed vector of the front car expressed in the trailer frame. It also suggests controlling the acceleration of the trailer (via a_1) and its rotation rate (via a_2) directly.

Proof. We have

$$\begin{aligned} \dot{\mathbf{y}} &= \mathcal{L}_{\mathbf{f}}\mathbf{h}(\mathbf{x}) + \underbrace{\mathcal{L}_{\mathbf{g}}\mathbf{h}(\mathbf{x})}_{=\mathbf{0}} \cdot \mathbf{u} \\ &= x_5 \cos(x_3 - x_4) \begin{pmatrix} \cos x_4 \\ \sin x_4 \end{pmatrix} \\ &\overset{(3.20)}{=} z_3 \begin{pmatrix} \cos z_5 \\ \sin z_5 \end{pmatrix}. \end{aligned} \tag{3.22}$$

Moreover, we can write

$$\begin{cases} \dot{z}_3 &= \mathcal{L}_{\mathbf{f}} z_3 + \mathcal{L}_{\mathbf{g}} z_3 \cdot \mathbf{u} \\ \ddot{z}_5 &= \mathcal{L}_{\mathbf{f}}^2 z_5 + \mathcal{L}_{\mathbf{g}}\mathcal{L}_{\mathbf{f}} z_5 \cdot \mathbf{u} \end{cases}$$

or equivalently

$$\begin{pmatrix} \dot{z}_3 \\ \ddot{z}_5 \end{pmatrix} = \mathbf{A}(\mathbf{x}) \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \mathbf{b}(\mathbf{x}), \tag{3.23}$$

where

$$\mathbf{A}(\mathbf{x}) = \begin{pmatrix} \mathcal{L}_{\mathbf{g}_1} z_3 & \mathcal{L}_{\mathbf{g}_2} z_3 \\ \mathcal{L}_{\mathbf{g}_1}\mathcal{L}_{\mathbf{f}} z_5 & \mathcal{L}_{\mathbf{g}_2}\mathcal{L}_{\mathbf{f}} z_5 \end{pmatrix}$$

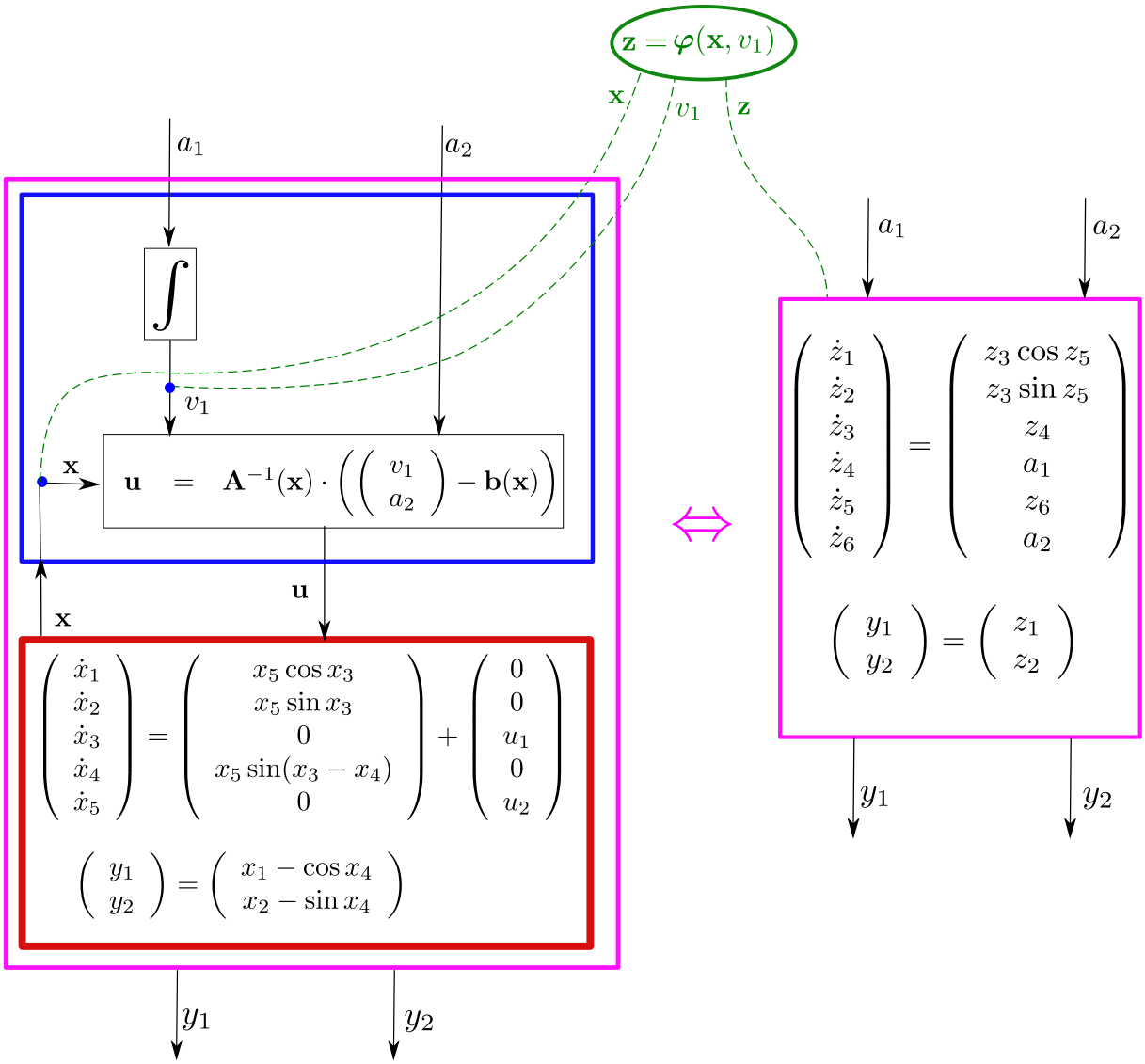


Figure 3.6: The two systems in the magenta boxes are equivalent.

and

$$\mathbf{b}(\mathbf{x}) = \begin{pmatrix} \mathcal{L}_{\mathbf{f}} z_3 \\ \mathcal{L}^2_{\mathbf{f}} z_5 \end{pmatrix}.$$

It is trivial to check that $\mathbf{A}(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ are the matrices given in the theorem. The matrix $\mathbf{A}(\mathbf{x})$ is singular if and only if $x_5 = 0$, that is if the speed is zero. If we take the linearizing feedback

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \left(\mathbf{v} - \mathbf{b}(\mathbf{x}) \right),$$

then (3.23) becomes

$$\begin{pmatrix} \dot{z}_3 \\ \ddot{z}_5 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}. \tag{3.24}$$

Finally, we have

$$\begin{aligned}
\dot{z}_1 &\overset{(3.22)}{=} z_3 \cos z_5 \\
\dot{z}_2 &\overset{(3.22)}{=} z_3 \sin z_5 \\
\dot{z}_3 &\overset{(3.24)}{=} v_1 \\
\dot{z}_4 &\overset{(3.20)}{=} \dot{v}_1 \overset{(3.19)}{=} a_1 \\
\dot{z}_5 &\overset{(3.20)}{=} \dot{x}_4 \overset{(3.18)}{=} x_5 \sin(x_3 - x_4) \overset{(3.20)}{=} z_6 \\
\dot{z}_6 &= \ddot{z}_5 \overset{(3.24)}{=} v_1,
\end{aligned}$$

which corresponds to (3.21). □

As illustrated by Figure 3.7, the sum of the relative degrees of each output $(3 + 3)$ is now equal to the dimension of the system (*i.e.* 6).

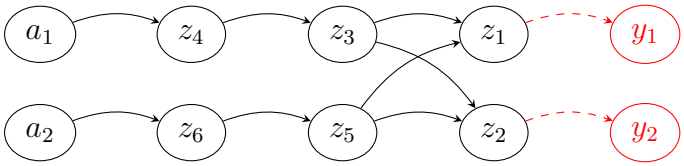


Figure 3.7: Graph of the differential delays of the flattened system, described by (3.21).

3.3.4 Simulation and experimentation

Van der Pol vector field

Now that the model of the system has its flat output, we can control it. We choose here to make the trailer follow a specific dynamics: the Van der Pol vector field. This theoretical application has several advantages. It allows verifying that the controller behaves as expected on a quite complex vector field. Indeed, although the equations of this vector field are simple to implement, the resulting dynamic is complex, with strong non-linearities. Besides, this vector field admits a limit cycle, so it is simple to check whether the followed trajectory matches the vector field, and this limit cycle has no analytical expression, so it is not possible to use it in the controller. This vector field is illustrated Figure 3.8, with some flows (trajectories following the vector field) to make the limit cycle evident.

We consider the flattened system defined by (3.21)

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}_z(\mathbf{z}) + \mathbf{g}_z(\mathbf{z}) \cdot \mathbf{a} \\ \mathbf{y} = \mathbf{h}_z(\mathbf{z}). \end{cases} \tag{3.25}$$

We want $\mathbf{y}$ to follow a desired dynamics $\dot{\mathbf{y}} = \mathbf{\Psi}(\mathbf{y})$, which is, in this application, the Van der Pol equation given by

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} y_2 \\ -(y_1^2 - 1)\, y_2 - y_1 \end{pmatrix}}_{\mathbf{\Psi}(\mathbf{y})}. \tag{3.26}$$

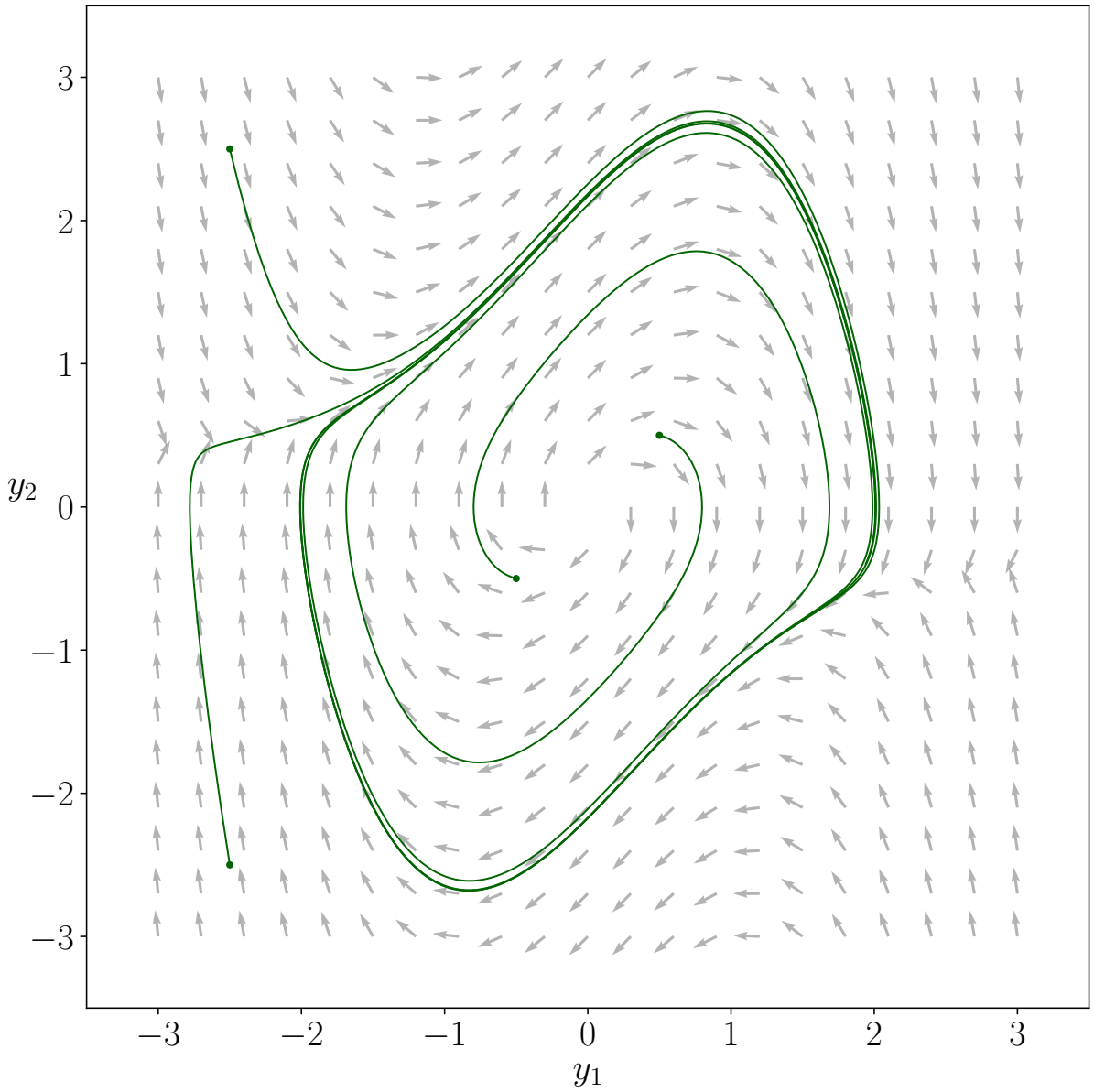


Figure 3.8: Different flows, in green, on the Van der Pol vector field. They all converge to the limit cycle.

The error $\mathbf{e}$ that we want to cancel is the difference between the course of the trailer and the direction given by the vector field:

$$\begin{aligned} \mathbf{e}(\mathbf{z}) &= \dot{\mathbf{y}} - \mathbf{\Psi}(\mathbf{y}) \\ &= \mathcal{L}_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) - \mathbf{\Psi}(\mathbf{h}_z(\mathbf{z})) \\ &= \begin{pmatrix} z_3\cos z_5 - z_2 \\ z_3\sin z_5 + (0.01\cdot z_1^2 - 1)z_2 + z_1 \end{pmatrix}. \end{aligned} \tag{3.27}$$

We have

$$\begin{aligned} \dot{\mathbf{e}}(\mathbf{z}) &= \mathcal{L}^2_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) - \mathcal{L}^1_{\mathbf{f}_z}\mathbf{\Psi}(\mathbf{h}_z(\mathbf{z})) \\ &= \begin{pmatrix} -z_3 z_6\sin z_5 - z_3\sin z_5 + z_4\cos z_5 \\ (z_6 + \frac{z_1 z_2}{50} + 1)z_3\cos z_5 + (\frac{z_1^2 z_3}{100} - z_3 + z_4)\sin z_5 \end{pmatrix} \end{aligned} \tag{3.28}$$

and

$$\ddot{\mathbf{e}}(\mathbf{z}) = \mathcal{L}^3_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) + (\mathcal{L}_{\mathbf{g}_z}L^2_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}))\cdot\mathbf{a} - \mathcal{L}^2_{\mathbf{f}_z}\boldsymbol{\Psi}(\mathbf{h}_z(\mathbf{z})). \tag{3.29}$$

We do not give the full expressions of all quantities with respect to the z_i's for the sake of clarity, and as these expressions are automatically computed using symbolic computation. We have $\deg(e_1) = \deg(e_2) = 2$, this is why the dependency with respect to $\mathbf{a}$ occurs only at the second derivative $\ddot{\mathbf{e}}$. Let us choose the error equation

$$\ddot{\mathbf{e}} + 2\dot{\mathbf{e}} + \mathbf{e} = \mathbf{0} \tag{3.30}$$

that makes $\mathbf{e}$ converge to zero. We get

$$\begin{aligned} \underbrace{\mathcal{L}^3_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) + (\mathcal{L}_{\mathbf{g}_z}\mathcal{L}^2_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}))\cdot\mathbf{a} - \mathcal{L}^2_{\mathbf{f}_z}\boldsymbol{\Psi}(\mathbf{h}_z(\mathbf{z}))}_{\ddot{\mathbf{e}}(\mathbf{z})} &+ \\ 2(\underbrace{\mathcal{L}^2_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) - \mathcal{L}^1_{\mathbf{f}_z}\boldsymbol{\Psi}(\mathbf{h}_z(\mathbf{z}))}_{\dot{\mathbf{e}}(\mathbf{z})}) &+ \\ \underbrace{\mathcal{L}_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) - \boldsymbol{\Psi}(\mathbf{h}_z(\mathbf{z}))}_{\mathbf{e}(\mathbf{z})} &= \mathbf{0} \end{aligned} \tag{3.31}$$

or equivalently

$$\begin{aligned} \mathbf{a} &= \boldsymbol{\beta}(\mathbf{z}) \\ &= -\left(\mathcal{L}_{\mathbf{g}_z}\mathcal{L}^2_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z})\right)^{-1}\cdot\left(\mathcal{L}^3_{\mathbf{f}_z}\mathbf{h}_z(\mathbf{z}) - \mathcal{L}^2_{\mathbf{f}_z}\boldsymbol{\Psi}(\mathbf{h}_z(\mathbf{z})) + 2\dot{\mathbf{e}}(\mathbf{z}) + \mathbf{e}(\mathbf{z})\right). \end{aligned} \tag{3.32}$$

Combining this expression with the controller (3.19), as illustrated by Figure 3.9, we get the trailer center following exactly the required vector field. Figure 3.10 illustrates the behavior of the controller.

The feedback linearization method leads to a successful controller of the car-trailer system. The latter system stays relatively simple compared to huge mechanical systems involving complex dynamics or multiple subsystems. Nevertheless, the equations are already barely readable. That is why the formalism of this method brings a genuine interest, allowing the use of symbolic computation.

The flattening operation of the output has also shown its interest by helping to understand the links between the inputs and the outputs and, especially, by ensuring the controllability of a system's modeling. The other interest of having a flat output is the property that allows to find back the whole state just from the output and its derivatives. That is what will be used later in Section 3.4 to validate the controller under state constraints using interval analysis.

Experimentations

After validating the behavior of the controller on simulation, the next step is to test it on real robots. Firstly, it was more convenient to implement it on a ground robot towing a sled. A picture of this experimentation is shown in Figure 3.11. A magnetometer was placed into the sled, which was following a boustrophedon to realize a magnetic map of the field. The result of this practical

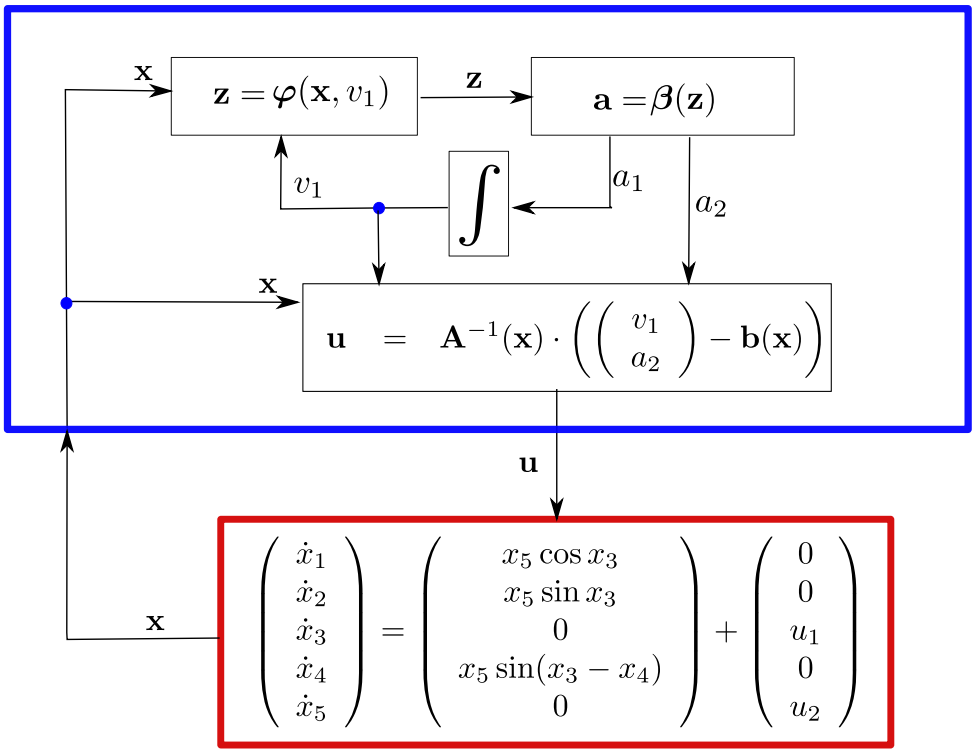


Figure 3.9: With the designed controller, the output $\mathbf{y}$ follows exactly the Van der Pol dynamics.

work is given in Figure 3.12. It is part of the "Magmap" project carried out by students of the robotic engineering training program [124]. A video of the robot towing the sled is also available at `https://youtu.be/yrC3tHgXiOo`.

Then, some tests have been carried out in the scope of the search for the *Cordelière*. The magnetometer was towed by *Boatbot* and followed isobaths to avoid regulating the depth of the sensor. A picture of the immerged magnetometer is proposed in Figure 3.13.

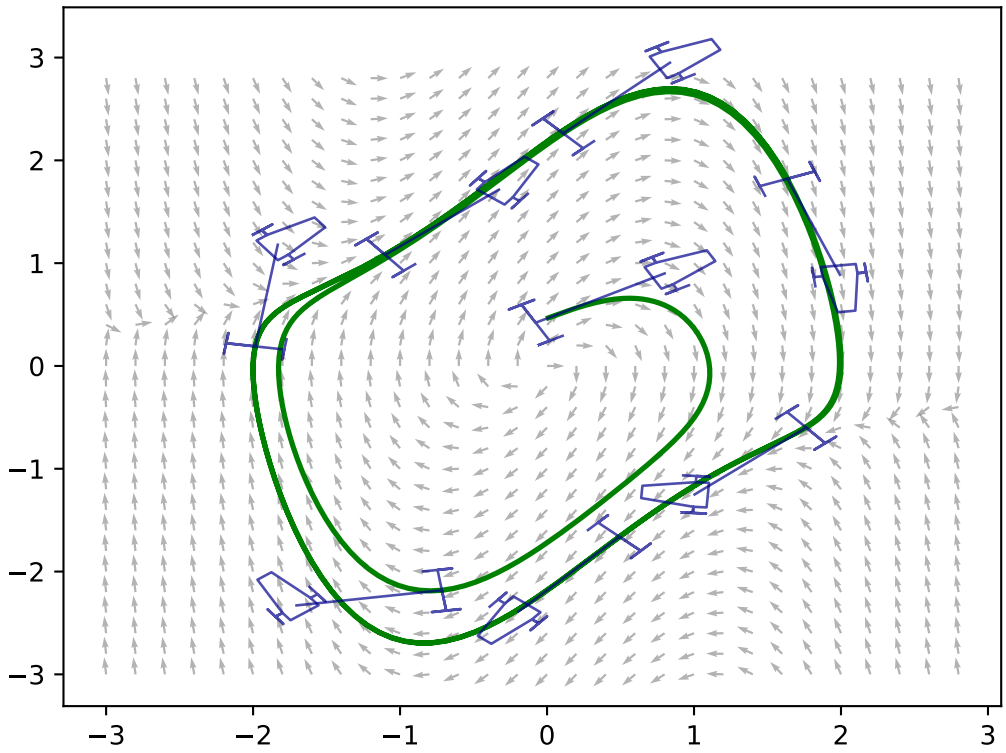


Figure 3.10: Simulation of the tank-trailer system following the Van der Pol vector field, using the designed controller. The trajectory followed by the trailer is depicted green and converges to the limit cycle, as expected.

Figure 3.11: The green sled is towed by *Saturne*, an Unmanned Ground Vehicle (UGV).

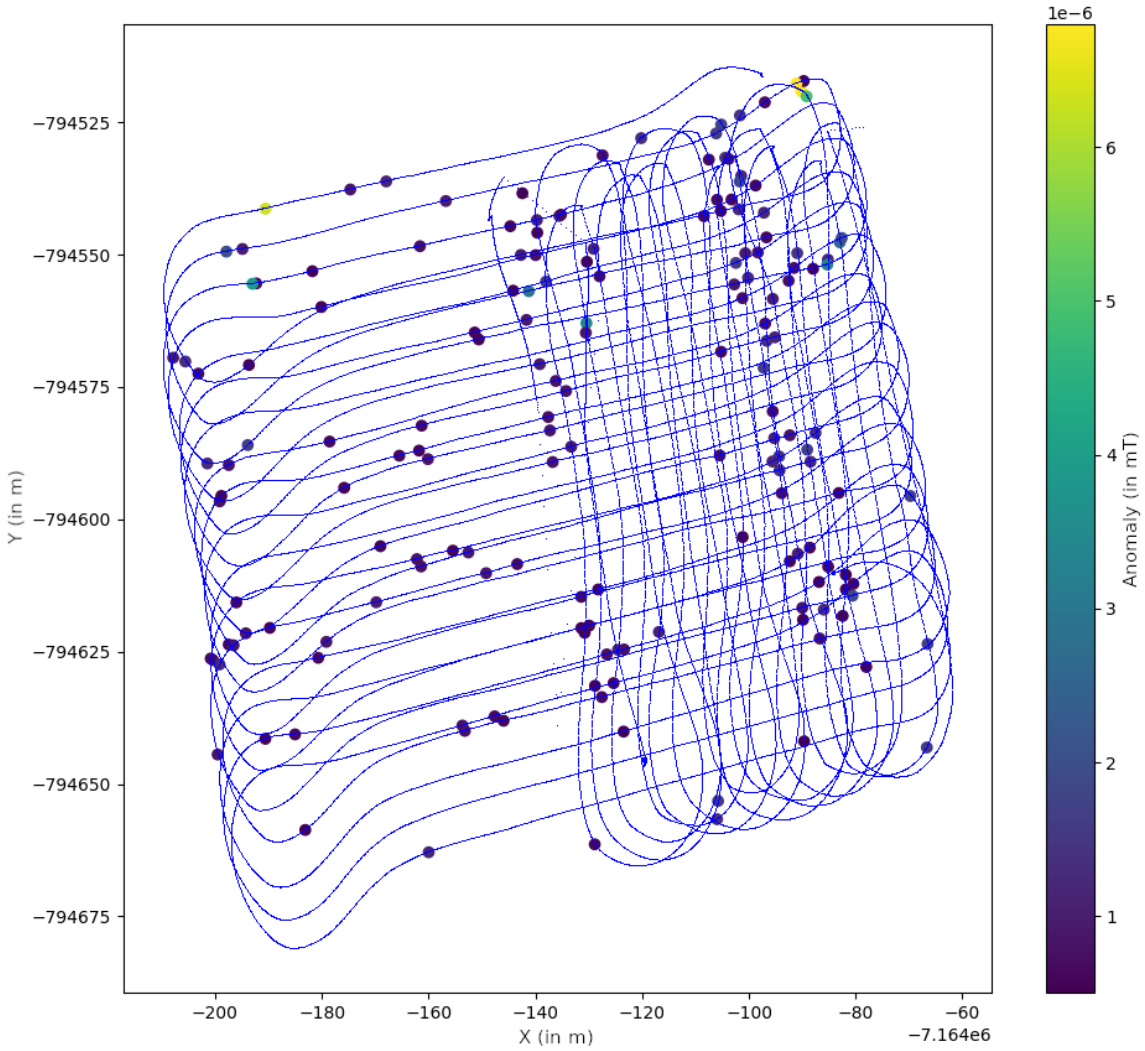


Figure 3.12: Magnetic map of the field of ENSTA Bretagne.

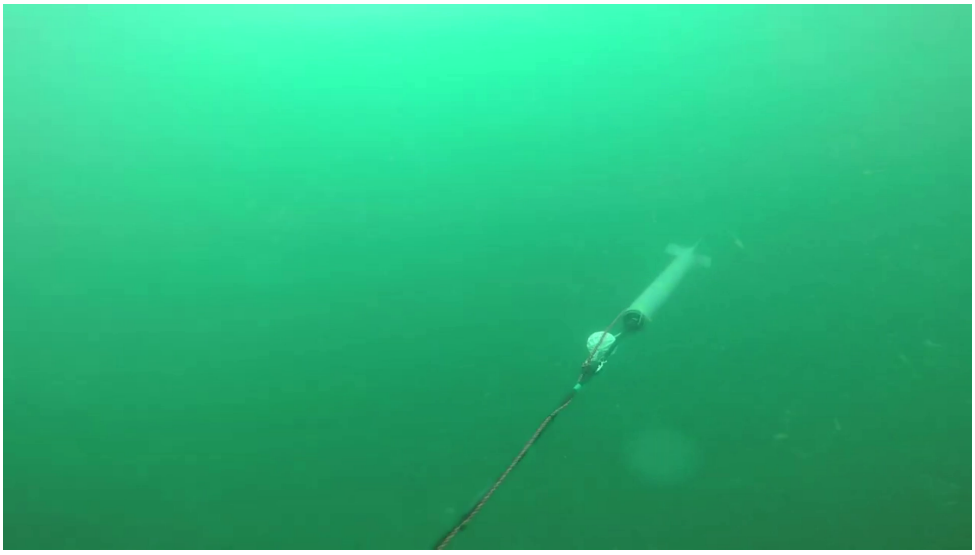


Figure 3.13: The magnetometer is immerged a dizain of meters behind *Boatbot*.

3.4 Validating the trajectory of the trailer with follow sets

In the previous section, the control of a car-trailer system has been studied. We propose now to guarantee the safety of the system by defining state constraints and verifying that these constraints are never violated while using the previously designed controller.

Thus, the objective is to take into account a state constraint. This task amounts to solving non-linear inequality constraints mixed with non-linear differential equations. Even if planning methods [93, Part IV] have provided some interesting results, the problem can still be considered as open as soon as some guarantee is required.

Let consider the car-trailer system again. We have already proposed a controller such that the output follows exactly the desired vector field in Section 3.3.

Now we would like to study issues related to the safety of the robot. To ensure its safety, we define some constraints on the state of the system. Indeed, there are some states either physically impossible or too hazardous to be considered. For instance, the angle between a car and its trailer should remain within an interval to avoid a self-collision. These constraints can also express the fact that the robot should not move into an obstacle. Thus, the objective is to find the *follow set*, the set containing all the "safe" positions, with respect to a given controller. In other words, knowing the controller and the dynamics of the robot, it is possible to predict the state of the robot for each position, and thus know whether the constraints are violated.

3.4.1 Follow set

Observability

In order to properly define a follow set, we introduce the notion of *observability*. A system is said to be *observable* [42] if there exist a function $\mathbf{\Phi}$ and integers $r_1, \ldots, r_m$ such that

$$\mathbf{x} = \mathbf{\Phi}\left(y_1, \dot{y}_1, \ldots, y_1^{r_1}, \ldots, y_m, \dot{y}_m, \ldots, y_m^{r_m}\right). \tag{3.33}$$

The integers r_i generally correspond to the relative degrees of derivation for the outputs y_i, $i = 1, \ldots, m$, but this is not mandatory. This assumption is valid for many systems as soon as it is flat with the output $\mathbf{y}$, which is then called a *flat output*. We have seen that we do have a flat output for the proposed model of the car-trailer system in Section 3.3.3. So in what follows, we can assume that we have an observable system, and thus the function $\mathbf{\Phi}$ is available.

Defining follow sets

In the case where $\mathbf{y}$ follows exactly the dynamics $\mathbf{\Psi}$, we can write

$$y_j^{(i)} = \mathcal{L}_{\mathbf{\Psi}}^i y_j, \tag{3.34}$$

using once again the Lie derivatives.

We define the *follow set* $\mathbb{Y}$ as

$$\mathbb{Y} = \boldsymbol{\gamma}^{-1}(\mathbb{X}), \tag{3.35}$$

where

$$\begin{aligned}\boldsymbol{\gamma} : \mathbb{R}^m &\to \mathbb{R}^n \\ \mathbf{y} &\mapsto \boldsymbol{\Phi}(y_1, \mathcal{L}_{\boldsymbol{\Psi}} y_1, \ldots, \mathcal{L}_{\boldsymbol{\Psi}}^{\rho_1} y_1, \ldots, y_m, \mathcal{L}_{\boldsymbol{\Psi}} y_m, \ldots, \mathcal{L}_{\boldsymbol{\Psi}}^{\rho_m} y_m)\end{aligned}$$

and $\mathbb{X}$ is the set of state constraints that should be satisfied for the state $\mathbf{x}$. The set $\mathbb{Y}$ corresponds to the set of all $\mathbf{y}$ such that if $\mathbf{y}$ follows the dynamics $\boldsymbol{\Psi}$, then all state constraints are satisfied. Most of the time, the set $\mathbb{Y}$ cannot be computed exactly because of the non-linearities between the output $\mathbf{y}$ and the state vector. So, using a set inversion approach, an inner and an outer approximation for $\mathbb{Y}$ can be obtained using the Set Inversion Via Interval Analysis (SIVIA) algorithm, as introduced in Section 2.3.3. Finally, the exact solution set is bracketed between the obtained inner and the outer approximation.

Once this follow set has been computed, a reachability analysis could be performed to find viable domains in $\mathbb{Y}$ [97]. Thus, all found $\mathbf{y}$ is such that all state constraints are always satisfied as long as the dynamics $\boldsymbol{\Psi}$ is followed.

Formalism with the car-trailer system

Consider the car-trailer system, which is controlled so that $\mathbf{y}$ follows the dynamics $\boldsymbol{\Psi}$. If we know y_1, y_2, $\dot{y}_1$, $\dot{y}_2$, $\ddot{y}_1$, $\ddot{y}_2$ then we can find the corresponding $\mathbf{x}$. This is illustrated by Figure 3.14 where we can understand that to follow the desired trajectory properly in the $\mathbf{y}$ space, there exists a unique possibility for $\mathbf{x}(t)$. The following proposition clarifies this point.

Proposition 3.4.1. *For the car-trailer system, we have* $\mathbf{x} = \boldsymbol{\Phi}(y_1, \dot{y}_1, \ddot{y}_1, y_2, \dot{y}_2, \ddot{y}_2)$ *as given by Figure 3.15.*

Proof. We have

$$\begin{pmatrix}\dot{y}_1 \\ \dot{y}_2\end{pmatrix} = \mathcal{L}_{\mathbf{f}}(\mathbf{y}) = \begin{pmatrix}x_5 \cos(x_3 - x_4)\cos x_4 \\ x_5 \cos(x_3 - x_4)\sin x_4\end{pmatrix}. \tag{3.36}$$

Thus

$$x_4 = \text{atan2}(\dot{y}_2, \dot{y}_1) \tag{3.37}$$

and from (3.18), we get

$$\begin{pmatrix}x_1 \\ x_2\end{pmatrix} = \begin{pmatrix}y_1 \\ y_2\end{pmatrix} - \begin{pmatrix}\cos x_4 \\ \sin x_4\end{pmatrix}. \tag{3.38}$$

Moreover, differentiating (3.37), we obtain

$$\dot{x}_4 = \frac{\dot{y}_1 \ddot{y}_2 - \ddot{y}_1 \dot{y}_2}{\dot{y}_1^2 + \dot{y}_2^2}. \tag{3.39}$$

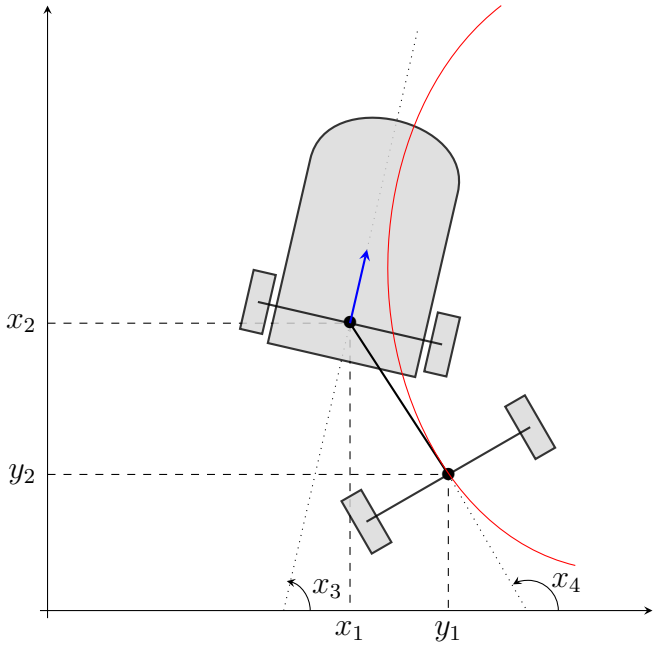


Figure 3.14: From $\mathbf{y}$ and its derivatives, we can find $\mathbf{x}$.

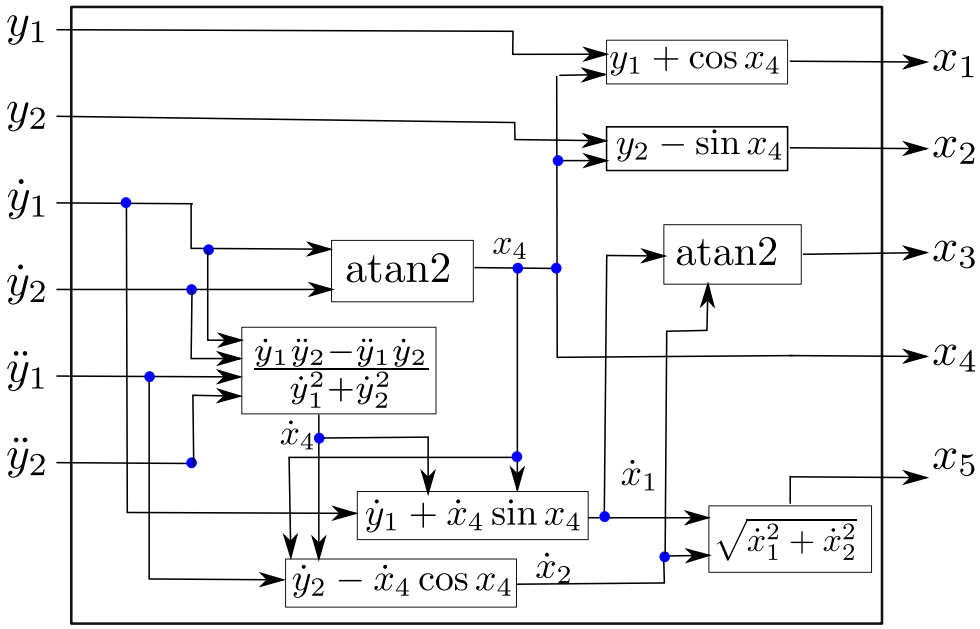


Figure 3.15: Expression of $\mathbf{\Phi}\left(y_1, \dot{y}_1, \ddot{y}_1, y_2, \dot{y}_2, \ddot{y}_2\right)$.

Thus

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} + \dot{x}_4 \begin{pmatrix} \sin x_4 \\ -\cos x_4 \end{pmatrix} \tag{3.40}$$

and

$$x_3 = \text{atan2}(\dot{x}_2, \dot{x}_1) \tag{3.41}$$

$$x_5 = \sqrt{\dot{x}_1^2 + \dot{x}_2^2}. \tag{3.42}$$

□

Finally, from the flat output of the system, it is possible to compute the whole state and validate the respect of the constraints. With the example of the car-trailer system, the output is the trailer position. Therefore, it is possible to know every place the robot can go safely by validating these places for the constraints.

3.4.2 Applications on the safety of the car-trailer system

We propose now to characterize different follow sets to illustrate the approach. We consider the trailer is following the Van der Pol vector field with the controller designed in Section 3.3.

We need to define constraints relevant to the safety of the system. Probably, the most critical state constraints are collisions. Figure 3.16 illustrates two types of collisions: external and internal.

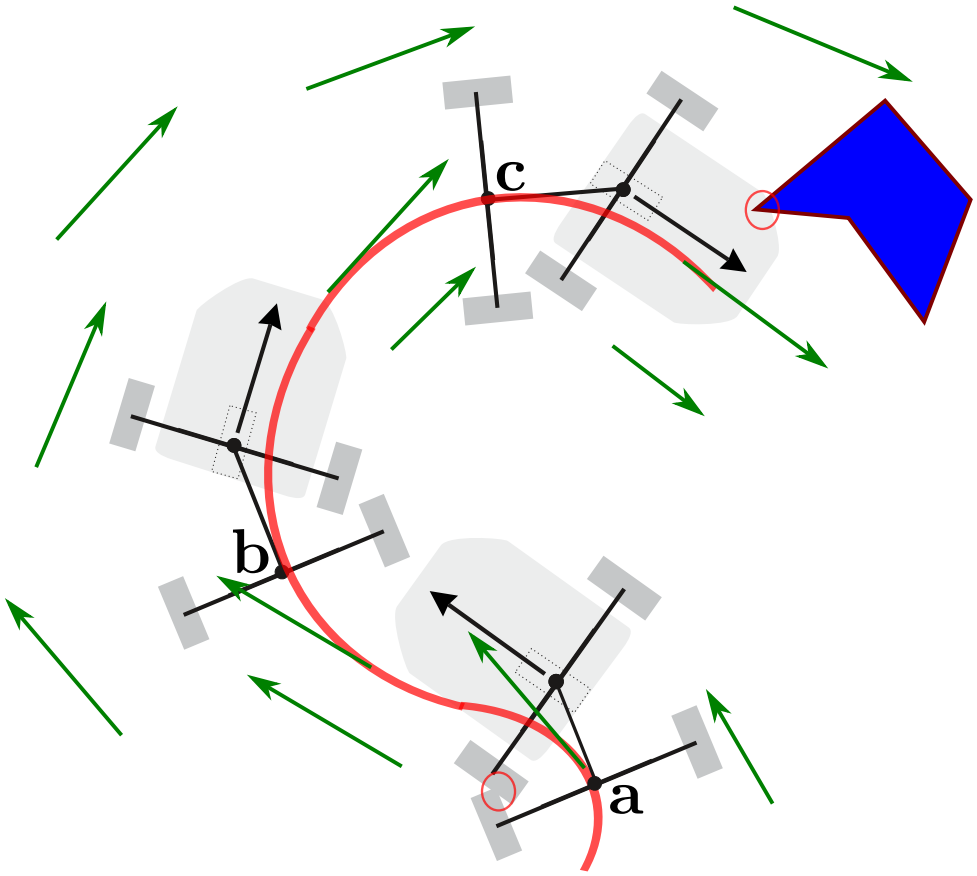


Figure 3.16: Since $\mathbf{a}$ implies an internal collision and $\mathbf{c}$ corresponds to a collision with the polygon, $\mathbf{a}$ and $\mathbf{c}$ are not in the follow set $\mathbb{Y}$.

External collision Assume that there exists a polygonal obstacle to be avoided (purple in Figure 3.17). The orange area is the place where the trailer center cannot go safely with the required dynamics. In the green area, the output $\mathbf{y}$ can safely follow the cycle with the guarantee that the robot never collides with the obstacle. The follow set has been computed with the

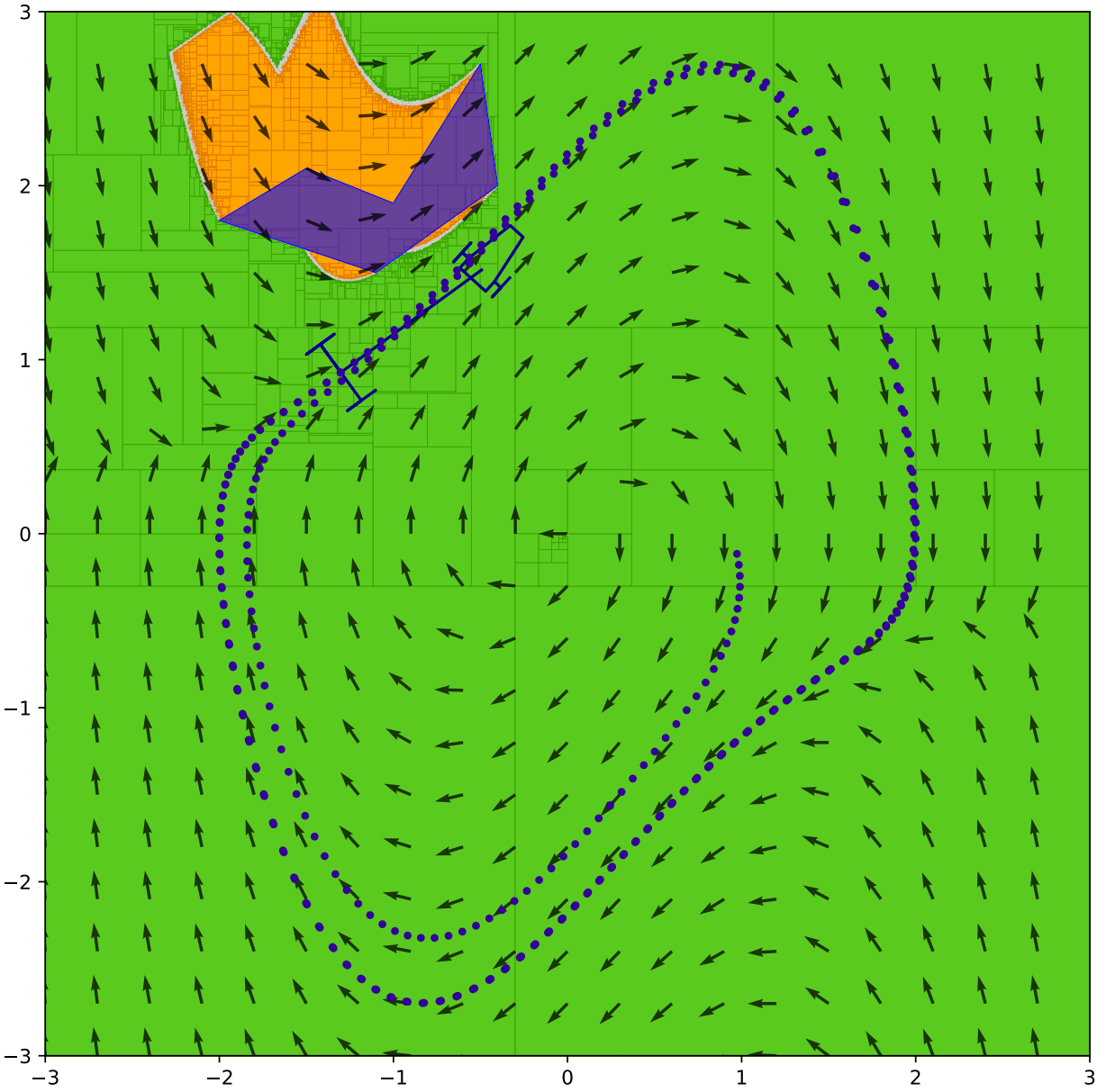


Figure 3.17: The purple polygon is the obstacle the car must not collide with. The follow set is painted green. This Figure has been obtained in less than 90 seconds on a basic laptop (with a processor of 1.7GHz).

algorithm SIVIA based on interval analysis. The polygonal assumption for the obstacle is not a limitation of the method. Any obstacle with a known shape could have been considered as well.

Internal collision Assume that if a maximum angle of 70° exists between the trailer and the tank, an internal collision occurs. The set of all proper trailer positions is painted green in Figure 3.18, assuming that the robot follows the required dynamics. Again, we have validated the painted trajectory and the limit cycle since it remains in the follow set.

An illustrating video is given at the following link: https://youtu.be/892_by8LVEw.

The computed follow set is an inner approximation (and the orange area the complementary of the outer approximation), so it is a little smaller than the exact one to guarantee safe behavior.

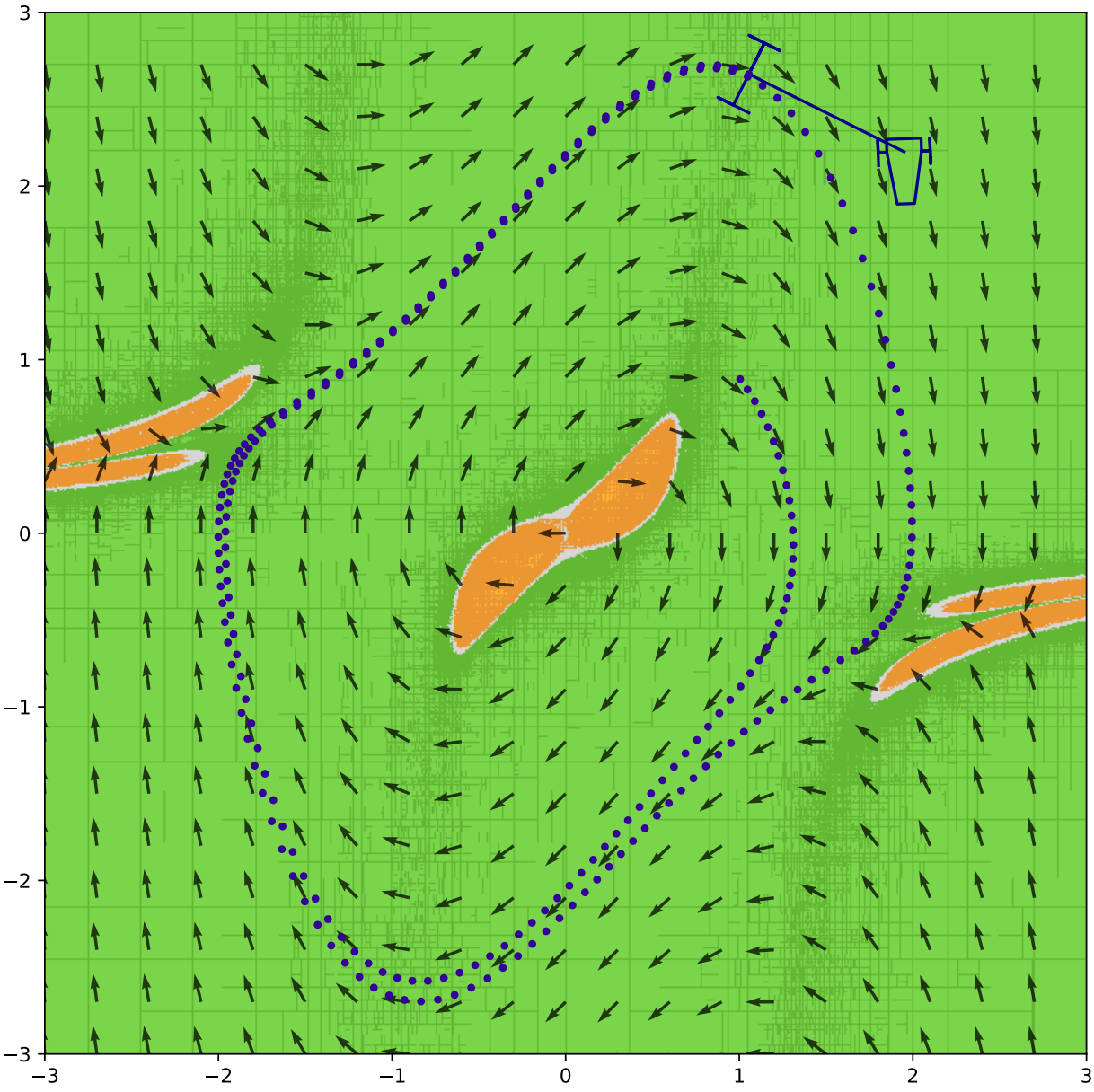


Figure 3.18: The green area corresponds to the follow set associated with an internal collision. In the orange area, the controller will lead to a state with an angle $x_4 - x_3$ too strong and the non-collision constraint will be violated. This Figure has been obtained in 2 minutes.

We do not want to include a position corresponding to a risky state in our follow set.

3.5 Conclusion

In this chapter, we have seen how the feedback linearization method can be used to propose the control of a non-linear system. Notions of controllability and flatness have been presented, and a robotic application is finally studied, dealing with the car-trailer system. The latter system has been chosen because its modelization can be used to handle the deported magnetometer in the context of searching wrecks.

Intervals are used to validate the trajectory of a robot when state constraints exist. The follow set, which is a subset of the output space, is defined and computed using an inner and outer approximation. The application with the car-trailer system has been shown as an example, where internal and external collisions should be avoided.

This problem can really show interest in the case of cumbersome articulated robots such as marine robots where sensors have to be towed in an environment with many obstacles, including rocks, islands, or other boats. It can also easily be extended to ground robots towing trailers for delivery purposes, for example.

Now that we have handled the first challenge, which is the safe control of the deported sensor, we will focus on the localization problem. Indeed, in this chapter, we assumed that we knew the state of the studied system. Nevertheless, access to the position of an underwater vehicle is a complex problem that constitutes the second challenge. It will be addressed in the following chapters.

CHAPTER 4

Localization using intervals

Contents

4.1 Introduction

When dealing with underwater robots, the localization problem can barely be neglected for almost any kind of mission. In our context, the objective is to find a wreck with a magnetometer. The magnetometer will provide several possible targets, but without an accurate localization of the sensor, we will not be able to associate a location to the targets. In addition, to ensure that we cannot miss the wreck, the localization must be accurate enough to allow a proper scan and guarantee that the trajectory of the sensor covers the whole area. Therefore, we have to address this localization issue.

The underwater environment is a very challenging place for localization problems. Electromagnetic waves are wholly absorbed by the water within a few meters, making the use of GNSS unsuitable. Different solutions have been developed when dealing with the absence of GNSS, especially providing localization methods for indoor environments like buildings. However, the parallel between underwater and inside environments stops quickly. The turbidity of water and the lack of light in deep water often highly complexifies or makes pointless the usage of vision sensors for long and mid-range information intake. On top of that, oceans are vast and unknown, and the seascape contains only sparse elements, rarefying possible natural seamarks. Finally, the only ways to get its bearings in this environment are proprioceptive sensors and, mainly, acoustic probing.

Concerning proprioceptive sensors, they make it possible to perform *dead reckoning*, also called *inertial navigation*. It requires to have an initial known position, typically taken from a GNSS data at the surface, before the beginning of the mission. Then, the current position of the robot is computed at any time by integrating the estimations of speed and orientation. However, errors are cumulative, making this approach suitable for only short missions and when the localization does not need to be too accurate. The notions of *short* and *accurate* directly depend on the quality of the sensors used, obviously depending on the price.

This chapter firstly presents the issues linked to the process of acoustic data in the scope of underwater localization. The scanning sonar, or Mechanically Scanned Imaging Sonar (MSIS), is mainly considered to apprehend acoustic-linked problematics. Different approaches using this sensor are envisioned to reach a localization in known environments, such as pools or harbors. The main difficulty is based on the extraction of reliable information among the outliers. A method using constraints is thus proposed.

4.2 Processing acoustic data

By acquiring information on its surrounding environments, algorithms can be developed to localize a robot, and eventually, simultaneously build a local map. Therefore, different sensors have been designed to gather data on distances as lidar and sonar, using light and sound waves, respectively. Lidars are widely used on aerial and ground robots, as they return many reliable

data. Unfortunately, underwater the light is quickly absorbed, depending on the turbidity of the water. It makes lidars barely suitable. Thus, the best choice is to use sonars. We now give an overview of the main issues when handling acoustic data.

4.2.1 Context

Using sound

Basically, a lidar and a sonar work the same way. They send a pulse in a given direction and measure the time elapsed before the reception of the same but reflected pulse. Then, from this elapsed time t and the celerity c, which is the speed of propagation of the wave, the range to the obstacle d can be computed using the simple formula:

$$d = \frac{ct}{2}. \tag{4.1}$$

With the speed of light being much higher than the sound one, much more data can be collected. On top of that, light beams are considered to propagate in straight lines without scattering, making returned data quite reliable. However, the behavior of sound waves is far from being straightforward, leading to noisy and unclear data. Firstly, acoustic data are very noisy due to ambient noise, embedded electronics, propagation. Propagation noises are linked to the variations of the local celerity, which depends on pressure, temperature, and salinity. So the local celerity must be estimated, causing a slight error and reducing the obtained resolution.

An acoustic beam does not just propagate in a straight line. It is characterized by its *beamwidth*: it propagates more like the shape of a cone. After emitting a pulse, the received acoustic level is recorded by the sonar with respect to the range, computed with formula (4.1). Figure 4.1 gives an example of how a pulse propagates and the shape of the associated collected data in an ideal scenario.

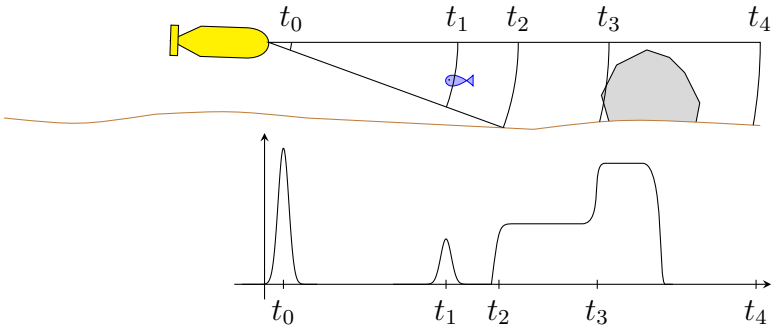


Figure 4.1: Received data from sonar after a ping emission in an ideal scenario.

The peaks in the graph correspond to targets in reality. After the emission, at t_0, a signal is received: it corresponds to the noise linked to the emission itself. This phenomenon is called *cross-talk*. Then at t_1, a tiny target (a fish) creates a small peak. At t_2, the seabed reflects some sound. The rock is detected at t_3, causing a significant peak. This graph is very approximately the one expected. It misses plenty of noises that largely complexify the received signal.

An example of a real received signal is given in Figure 4.2. So the detection of the targets within such a signal is not evident at all. The presence of outliers, *i.e.* peaks in the signal that do not correspond to actual targets makes the interpretation of gathered data intricate. Indeed, the returned data of an acoustic beam can be a complete outlier due to other acoustic sources or unexpected obstacles like other robots or fishes.

Another phenomenon that creates outliers is *multipath*. Natural surfaces never being perfectly smooth, a beam encountering an obstacle is reflected in different directions. Some of these directions make parts of the beam return to the sonar, either by the most direct way or through multipath. The latter means that the beam is reflected by several surfaces before being received by the sonar. Thus, the used path is longer than the direct one, and the associated peak is still received but too late. An illustration of this phenomenon is proposed in Figure 4.3, where acoustic beams are reflected by the seabed (magenta ray) and by the surface (red ray). The effects on the received signal are depicted in the graph using the same colors. Thus, several peaks correspond to the same target: the rock.

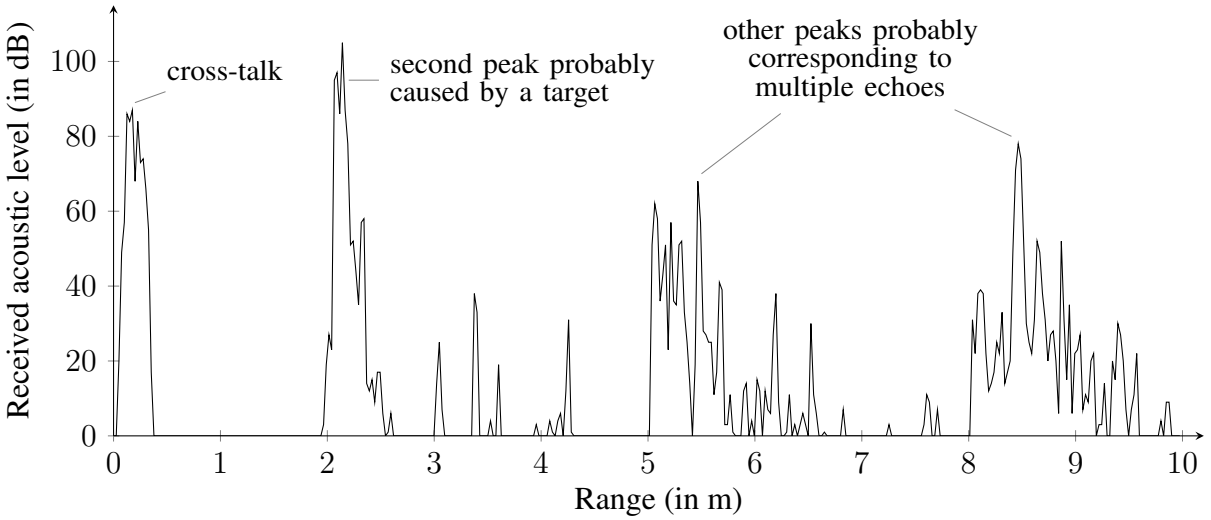


Figure 4.2: Example of the raw data returned by a sonar. The received acoustic level is represented according to the computed range using formula (4.1).

In addition, the drawn beam only corresponds to the *main lobe*. However, the beam pattern also has undesired side lobes and even back radiations [28, Chap. 14], causing more potential multipath.

The last phenomenon that mainly occurs in pools and harbors due to the walls is multiple echoes. The principle is simple: the beam travels from the sonar to the target then is reflected back. The sonar detects it, but it is not all. The beam is once again reflected by the sonar and can make the same path a new time. Thus, if a target is detected at t_1 and multiple echoes happen, then peaks appear in the signal at t_1, $2t_1$, $3t_1$, and so on until the energy is totally dissipated. It especially happens when dealing with walls or rigid and flat bodies, as most of the energy is

well-reflected, but this phenomenon can also be observed in shallow water between the surface and the seabed.

Thus, instead of directly returning a distance associated with measurement, a sonar generally provides the received acoustic level according to time. These data consist of a list of the received levels over time. On the example of real data given in Figure 4.2, several *peaks* can be spotted out. It leaves ambiguity in target detection and prevents from deducting a distance to an obstacle.

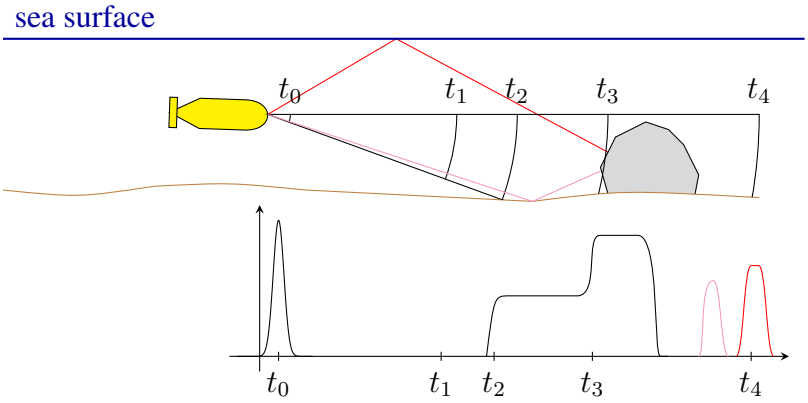


Figure 4.3: The beam is reflected through multipath, causing an outliers in the received signal.

By using such acoustic sensors, it is possible to obtain an overview of the environment through distances and bearing measurements to the seabed, obstacles, or anything that reflects acoustic waves. Many types of sonars exist, like single-beam directional sonar, scanning sonar, side-scan sonar, multibeam sonar, to name the principal. More details on these sensors can be found in [27, Sec. 25.4.3]. All of them are naturally concerned by the issues linked to acoustic signal propagation. It is all the more problematic on cheaper sonars as they gather less data, so outliers are more disturbing.

Scanning sonar in a known environment

A scanning sonar consists of a single-beam sonar mounted on a rotor, such that data are collected all around the sensor. Gathered data give information on all the obstacles in the plane orthogonal to the axis of rotation of the rotor. Thus, arrays of data such as the one of the Figure 4.2 are acquired. By representing the intensities with respect to a color bar, with lighter colors for higher intensities, we can plot together each of these arrays with respect to the bearing of measurement. For instance, Figure 4.4 represents data from a scanning sonar placed horizontally in a pool, such that it detects the walls and two gates. The map of the basin is given on the right of the Figure.

Other examples are given in Figure 4.5, with different pools of various sizes. Note that two different sonars are used; the Miniking is an older version of the Micron, but the acquired data are close.

Every image of the Figure 4.5 has been generated with a Mechanically Scanned Imaging Sonar (MSIS) (or scanning sonar) which was immobile during the scan. For each bearing, the

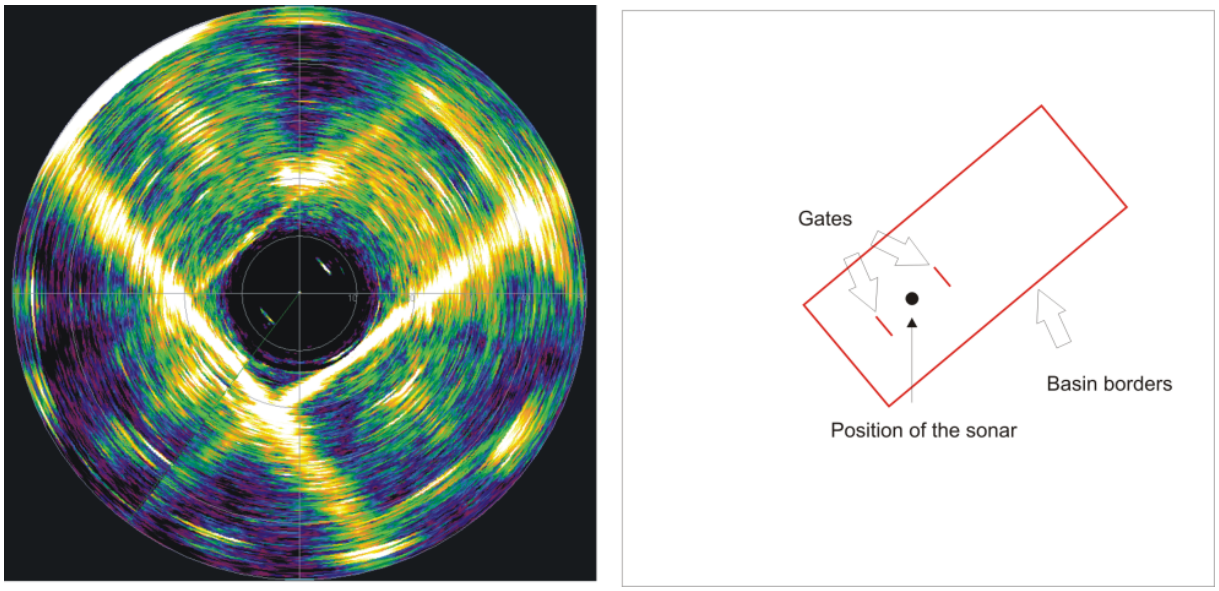


Figure 4.4: On the left, sonar data are represented with light color for high level of received signal. On the right are depicted the positions of the sonar, the walls, and two gates.

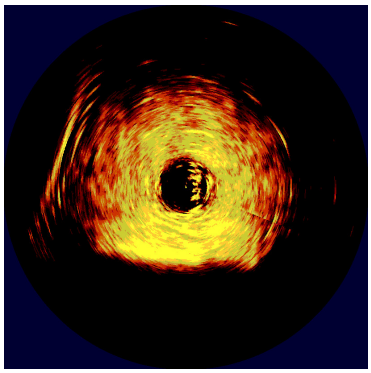
sonar emits a pulse and then listens to eventual feedback. The intensity of the received sound is collected with the associated time after emission. By knowing the speed of the sound in the water (about 1,500 m.s^{-1}), the distance is straightforwardly computed using the formula (4.1). For each bearing, each of these intensities is represented by a colored point, lighter for higher intensity, at its estimated position relative to the sonar.

For each example, the shape of the pool was either a rectangle or a "U-shape", *i.e.* an open rectangle looking like ⊔ and communicating with the sea. Finding back the walls of the different pools is far from being immediate. Sometimes, there is much noise inside the pool, multipath occurs regularly, and the walls are barely just a straight line and can be discontinuous.

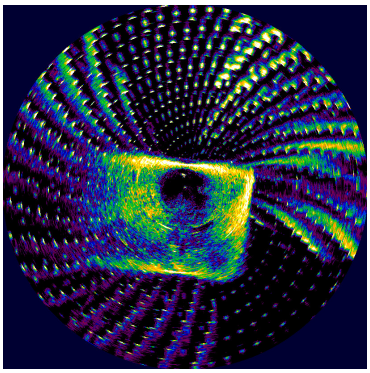
Algorithms

Underwater localization using such a MSIS has been already studied, using different approaches. We detail hereinbelow three interesting methods: peaks identification, SLAM, and image processing.

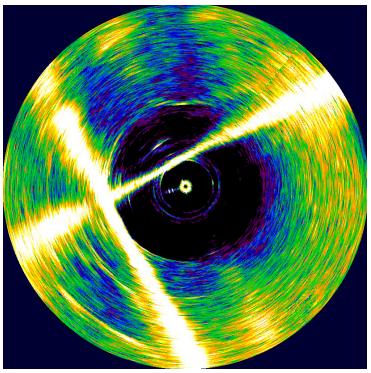
Peaks identification: The first localization approach assumes that the local map is known. Then, the idea is to identify peaks on sonar data to the obstacles of the map like walls, rocks, pillars. Thus, the suitable positions for the robot are reduced until the actual one is found. This approach is studied in Section 4.3 using interval analysis tools. The localization of a robot in a pool with a MSIS has already been studied in [72], also using interval analysis tools. A study, [134], merits to be mentioned even if it is a side-scan sonar that is used. The map contains information on indistinguishable landmarks (rocks on the seabed). The identification is based on a set-membership approach too.



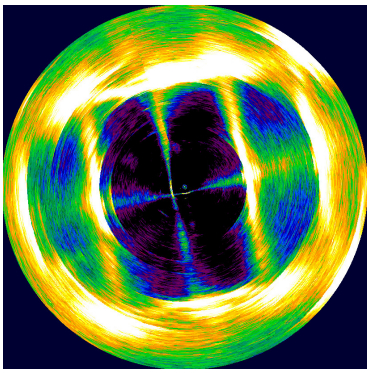
(a) Data taken in La Spezia, Italia, with a Tritech Miniking.



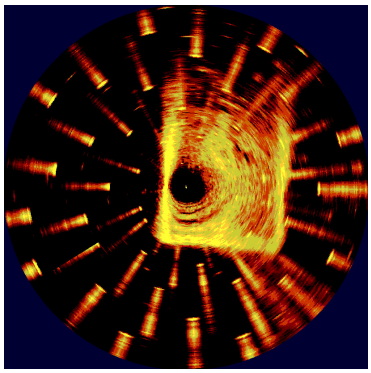
(b) Data taken in La Spezia, Italia, with a Tritech Miniking.



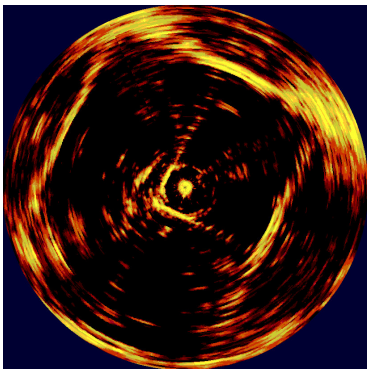
(c) Data taken in CINVESTAV, Mexico, with a Tritech Micron.



(d) Data taken in CINVESTAV, Mexico, with a Tritech Micron.



(e) Data taken in La Spezia, Italia, with a Tritech Miniking.



(f) Data taken in ENSTA Bretagne, Brest, France, with a Tritech Miniking.

Figure 4.5: Examples of data collected by a scanning sonar in different pools. The position of the sonar is always at the center of the picture. The objective is to find back the orientation and the relative position of the pool.

SLAM: With no assumption on the prior knowledge of the local map, SLAM algorithms are often very convincing. They are mostly based on probabilistic methods [151]. Thus, probabilistic filters such as EKFs or particle filters are used to match the different landmarks, or features, between the different views, as in [100]. However, it is often computationally not efficient, and the initial position is required, which is a problem for deep-water missions. Examples of different approaches for an archaeological application are given in [164]. A ROV is used to explore cisterns. A dynamic model of the ROV is used to predict the current state from a previous one and the control input. This method cannot provide precise results when external perturbations occur. Here, there are no currents, and the influence of the tether is taken into account, but it would not be suitable for an open sea environment.

With a scanning sonar, the main used method seems to consist of a probabilistic scan matching and an EKF to perform dead reckoning [108, 132, 18]. However, it requires a DVL to estimate the speed with enough accuracy for the dead reckoning part. In [163], no inertial or odometry is assumed. Only the data from the sonar are used to perform a SLAM based on features registration. This approach requires enough data to delimit clusters and can fail whether

clusters are too close. Finally, by using two scanning sonars, one horizontally and one vertically, a 3D map can be obtained as in [109] with cave exploration.

Note that the SLAM problem can also be solved with a set-membership approach. It is the case in [71], where an underwater robot is localized using a side-scan sonar.

Nevertheless, in the scope of this thesis, the SLAM problem is not handled. The map is assumed to be known, and the focus is on the interpretation of the acquired data.

Images processing approach: Another possible approach is to consider the images resulting from a whole turn of the sonar as in Figure 4.5 instead of each data separately. The advantages are that we can use well-established algorithms from image processing and that the 2D representation helps to spot out the walls more easily than by processing 1D bytes of data one by one. The method consists of correlating an image of the map (named the *template*) with the image obtained from the sonar. The template, which is smaller than the sonar image, is moved on each possible position on the sonar image, and a correlation score is computed each time. Then the position giving the highest score should correspond to the translation between the origin of the map and the estimated position of the robot. Naturally, this method requires having rotated the sonar image with respect to the heading of the robot in order to have the same orientation between the template and the sonar image.

There exist different scan matching algorithms. Two are compared in [68] with an application to image localization with MSIS data. This method has the advantage of taking an image as a map, so any structured or even unstructured map can be used. In addition, image algorithms have been highly optimized, so they are very fast. However, they require to wait for a 360° scan of the sonar to complete the image, which can be long, especially if the robot itself is rotating. If the sonar turns too fast, the obtained image becomes illegible. Enough data must be collected to obtain enough information in the image. Finally, multipath can mislead these algorithms because they can make appear several times the map in the sonar image, as in Figure 4.8d. An idea to improve this approach is to use interval analysis. Indeed, the score of each correlation between the template and the sonar image can be plotted in a new image with lighter points for higher scores. The position of these points corresponds to the possible relative localization of the robot with respect to the map. Instead of taking only the lighter point (the highest score), boxes wrapping every enough light points can be defined. These boxes are more susceptible to contain the actual position of the robot.

4.2.2 Constraints approach

Interpreting data

Acoustic data require to be interpreted with particular care. Otherwise, it can quickly lead to a wrong way. When dealing with underwater vehicles, one first priority is hardly always being

sure not to lose the expensive robot. So some guarantees on safety issues are very expected, and these guarantees directly depend on the correct interpretation of the data.

For instance, if we consider once again the Figure 4.2, the second peak probably corresponds to an obstacle. However, it would be daring to consider only this assumption which might be wrong. There is other information in this data that can be taken into account to increase the reliability of the process. The approach amounts to spot out and exploit the inliers while eliminating the outliers. So the goal is to extract as much as possible useful information from the data. Thus, inliers are most likely not to be forgotten. For example, each detected peak in the data could be considered with the assumption that at least one of them corresponds to an obstacle.

The relaxed intersection presented in Section 2.4.2 can be considered first. Then the assumption of one inlier among all the peaks can be implemented by taking the number of potential outliers q equal to the number of peaks detected minus one.

This reasoning that consists of proposing assumptions from the interpretation of the data leads to a constraints programming approach. The difficulties remain in formalizing correct constraints and handle outliers. Indeed, the chosen constraints must always be valid to guarantee the correctness of the result. But they should also be restrictive enough to converge in time to a satisfying result. Finally, a constraint that satisfies these two properties is called a *relevant constraint*.

Finding relevant constraints

It is intricate to find a relevant constraint. A single counterexample is enough to discredit a constraint candidate. For instance, the constraint "at least one of the peak corresponds to an inlier" may be misleading when dealing with an open pool (U-shape pool).

Finally, what emerged out of the thought is that the most reliable pieces of information lie in the *silence*, *i.e.* where there is no peak. Indeed, a peak can either be an inlier or an outlier, whereas silence almost always corresponds to the absence of obstacles. There is still a little doubt, hence the presence of the *almost* word in the previous sentence. There are some cases where a wall can be completely missed by the sonar. It happens when the direction of the emitted pulse is close to being parallel to the wall. In that case, the acoustic beam is nearly not reflected back by the wall, and a multipath phenomenon occurs. For instance, let consider the Figure 4.6 which is the Figure 4.5f in big, with the actual position of the pool drawn on it. Then, Figure 4.7 shows an example of a beam that is not reflected by the closest wall, but by another wall through multipath.

So this last example shows that a silence does not always mean that there is no obstacle. We cannot choose this constraint. However, it is relatively close to an actual reliable constraint. Indeed, after analyzing different data from sonar, we came to the following conclusion: when a ping is emitted perpendicularly to a wall, a silence always corresponds to the absence of obstacles.

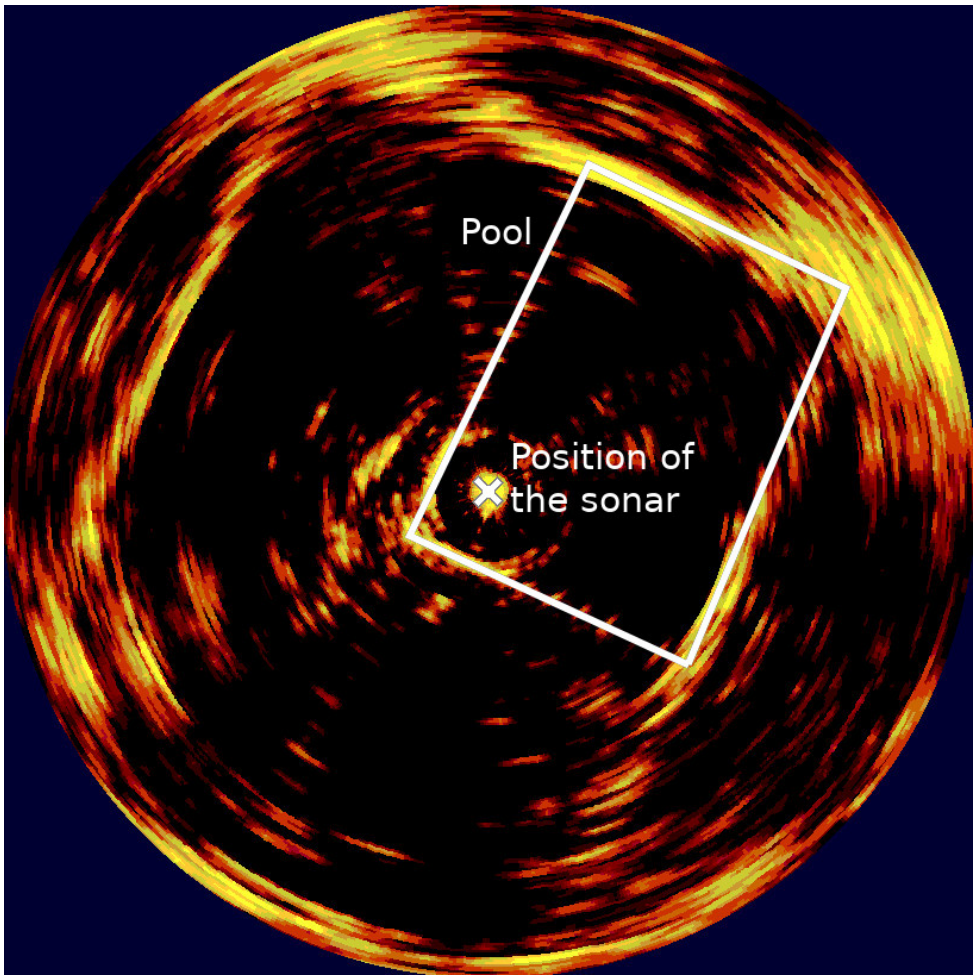


Figure 4.6: Data from a scanning sonar in a pool. The positions of the sonar and the pool are depicted in white.

The latter constraint corresponds to the reliable constraint we were searching for: it is always verified and still brings enough information to improve the localization. Figure 4.8 illustrates what is a silence in a sonar data and how it is interpreted for the localization of the robot. The three subfigures on the left represent incoming data from the sonar (the same as in Figure 4.2). A threshold (depicted in magenta on Figure 4.8) is used to separate the *peaks* from the *silences*. Note that we may have used other more complex algorithms to define these separations. However, it is the following step that matters here. The first detected silence is approximately between 0.4 m and 2 m (Figure 4.8c). In the known map of the pool, this information can be transcripted into *silence* areas (Figure 4.8d). A contractor is used to remove these silence areas from the possible positions of the robot. Indeed, if the received data were obtained from a position within a silence area, it would have meant that an echo sent perpendicularly to a wall would not have created a peak in the data, *i.e.* the constraint would have been violated. Finally, the other silences in the incoming data can also be taken into account to remove more impossible positions. It is illustrated in Figures 4.8e and 4.8f.

The design of such a contractor is done using a geometric approach. The following section

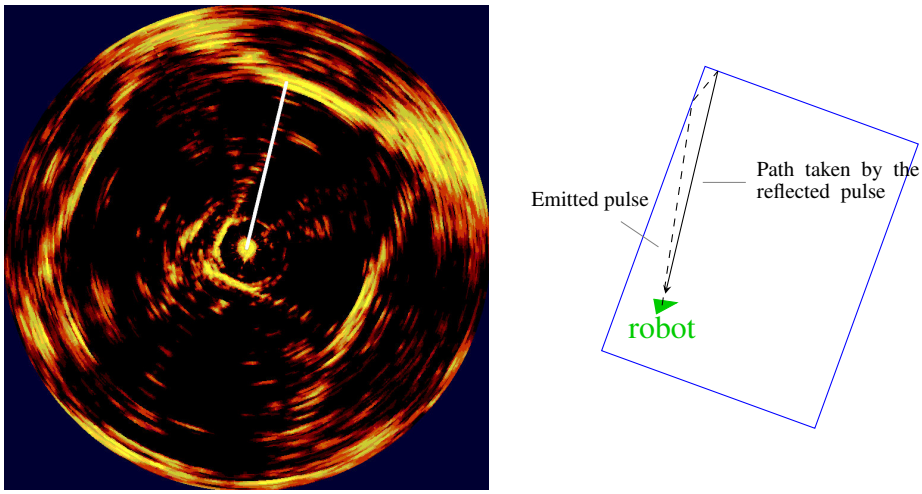
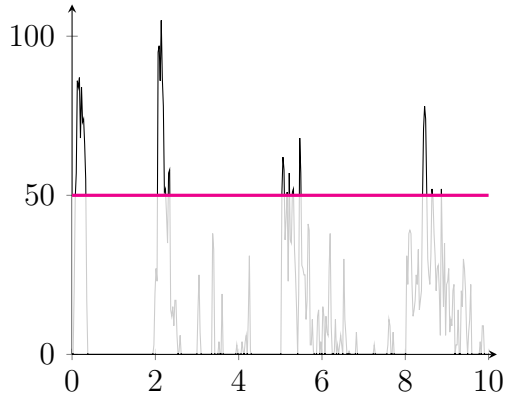
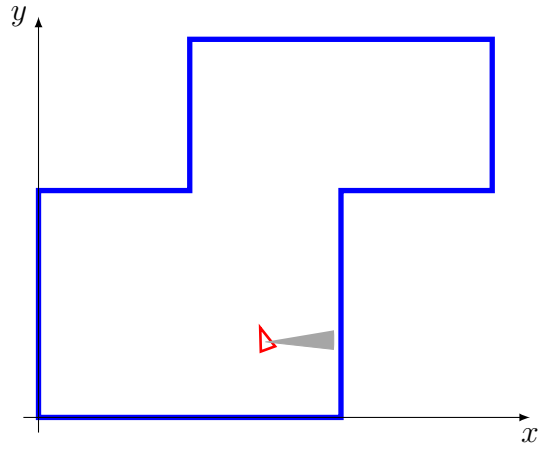


Figure 4.7: Multipath effect for a beam in a sonar image making parts of walls hardly detectable.

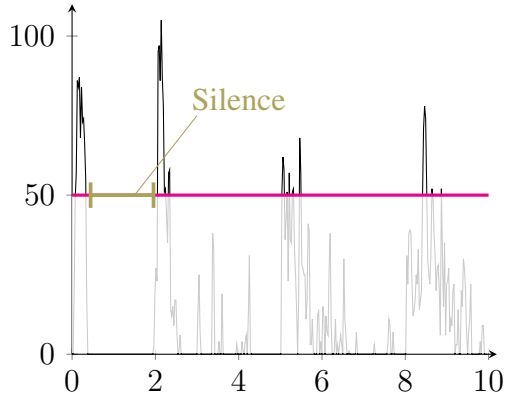
presents this step in depth.



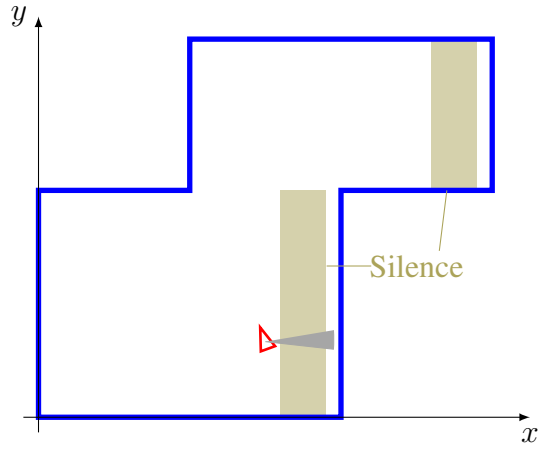
(a) Data from one ping from a sonar. A threshold is used to isolate peaks.



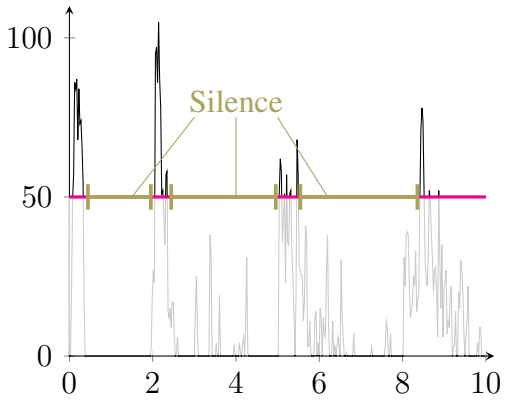
(b) A robot is in a polygonal pool. Its sonar sends a ping to the east.



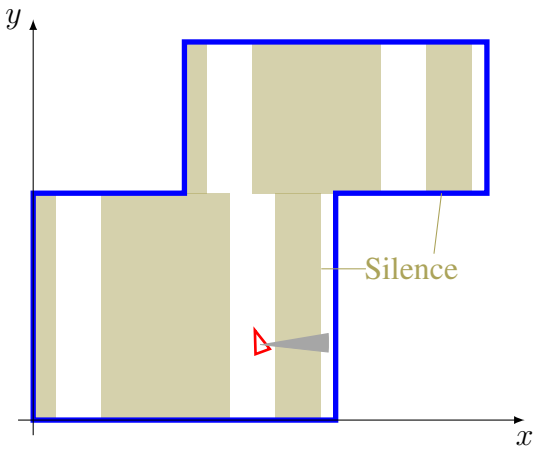
(c) The first silence in the data is extracted.



(d) The information of silence is used to eliminate some positions in the pool.



(e) Every silences in the data are identified.



(f) The information of silences are used to eliminate other positions in the pool.

Figure 4.8: Illustration of the information intake from silence.

4.3 Designing new contractors

This section studies the design of new contractors. The objective is to start from a constraint, based on a geometric approach, to define a set in the position space. This set is the searched one, and the contractor must characterize it. It amounts to solve a set inversion problem, and this is the purpose of the new contractor. Finally, the use of this contractor is shown in the context of robot localization, using some exteroceptive measurements.

4.3.1 Geometric approach

Localization with distance measurement

As presented in Section 2.3.2, a contractor is a tool used to approximate a set by outer approach. In localization context, the searched set is generally the state, or, more particularly, the position of the robot. Clues of this position can come from exteroceptive sensors, which can give absolute information. However, to achieve a successful localization, these measurements must be well interpreted. For instance, by using lidar, a robot can measure its distance to a wall. However, the distance returned by the sensor can be wrong: it can correspond to another wall, to a person, or any other obstacle. Moreover, even if the right wall is detected, the returned distance is often uncertain: a more or less significant error is added to the actual distance from the wall.

In this context, the proposed interval analysis approach consists in representing the position of the robot by a box $[\mathbf{x}]$, which is very large at the beginning as there is no information on the starting position. The idea is that this box always contains the actual position of the robot. Thus, the represented information is always reliable and guaranteed. Nevertheless, we also want this box to be as smaller as possible to obtain a more accurate localization. To reduce the size of the box, we use contractors. Each time the robot receives a new measure from its sensors, the data can be translated into a constraint, and therefore a contractor. This contractor is directly associated with the set of possible positions that match the new measurement.

Now we will detail how measurement can be translated into a constraint, with the example of the distance to a wall. We assume that the robot knows the absolute position of the wall and that the measurement returned by the sensor, although uncertain, is correct. Figure 4.9 summarizes the situation.

We also assume that the robot knows its heading and the bearing of its measurement. Finally, the absolute angle of the measurement is known and denoted by ψ. Thus, this measurement brings information about the localization: the following constraint describes the compatible positions of the robot.

This constraint amounts to find the distance from a point to a segment with a specific angle. So there are two cases: either the ray crosses the segment, then the distance can be computed, or the ray does not cross the segment, and the distance is equal to infinity. Note that both cases are a relevant information for the localization. To know in which case we are, we compute some

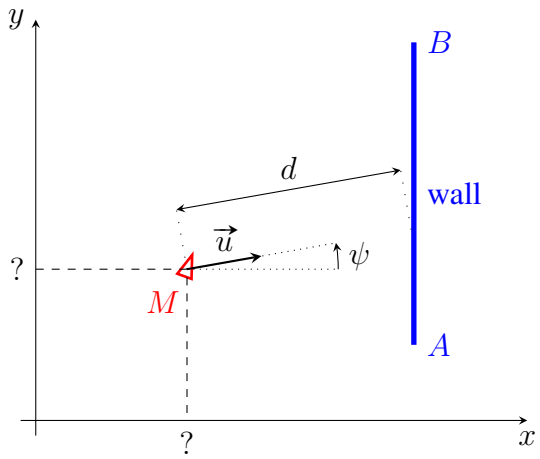


Figure 4.9: A robot (the red triangle, at M) is measuring its distance d to a wall represented by the blue segment $[A, B]$.

determinants as follow:

$$z_1 = \det(\overrightarrow{MA}, \vec{u}) \tag{4.2}$$
$$z_2 = \det(\vec{u}, \overrightarrow{MB}) \tag{4.3}$$
$$z_3 = \det(\overrightarrow{MA}, \overrightarrow{AB}) \tag{4.4}$$
$$z_4 = \det(\vec{u}, \overrightarrow{AB}). \tag{4.5}$$

Thus, we consider walls as vectors, which means that walls are oriented. If a wall is upside down, it is considered transparent. It is taken into account only if it is oriented counterclockwise. Note that it is not restrictive as it is possible to consider both cases, with the wall oriented in the two different configurations, if necessary. The three first scalar values (z_1, z_2, z_3) are all positive if and only if the ray from M along vector $\vec{u}$ crosses the wall $[A, B]$. So the condition to separate the two cases is whether the sign of the minimum of these three values is positive or negative (it is positive if and only if all the three values are positive). Finally, when this sign is positive, the ψ-oriented distance between the robot and the wall is equal to the quotient of z_3 and z_4.

Thus for one possible position M, measuring a distance $[d_{\text{mes}}]$, the constraint can be expressed as follows:

$$\mathcal{C}_{\text{dist\_wall}} : \begin{cases} \infty \in [d_{\text{mes}}] & \text{if } \min(z_1, z_2, z_3) < 0, \\ \frac{z_3}{z_4} \in [d_{\text{mes}}] & \text{otherwise.} \end{cases} \tag{4.6}$$

Associated contractor

Afterward, this constraint is translated into a contractor. The method used to realize this task is a forward/backward contractor, also called HC4-*revise* [8]. The principle is as follow: the input is the box $[\mathbf{x}]$ representing the approximation of the position. The goal is to contract this box to affine the knowledge of the position. It is impossible to straightforwardly contract the box, as the constraint coming from the measurement does not directly correspond to the position. It could be

as immediate with data from a perfect GNSS, for example. Here we can only contract an interval concerning the distance. So the forward step consists in computing this interval from the initial position box. Knowing the position of the wall (by way of the positions of the two extremity points: $[A, B]$) and the direction ψ of the measurement, we can compute the values z_1 to z_4 and thus the interval representing the distance between the current estimated position and the wall. This interval is then contracted according to the constraint. Concretely, it is the intersection of this interval and the measurement $[d_{\text{mes}}]$ which is calculated. If this intersection is empty, then an empty box is returned: the initially given box does not fit with the measurement. Otherwise, the contraction is propagated backward: this is the backward step. In other words, the new smaller (if the contraction was proper) interval is used to compute back the position box, which may be smaller. In this way, the algorithm returns the same, but contracted, box $[\mathbf{x}]$, giving a better approximation of the actual position of the robot. Figure 4.10 illustrates the different steps of this method.

Note that some pessimism may be introduced in such contractions: the magenta box of Figure 4.10 corresponding to the measurement contains some impossible values in the top left and bottom right corners. That is because boxes are aligned with plan axes. In this particular case, the resulting contraction brings no pessimism as these corners are not used at the end. However, it was only because the initial box $[\mathbf{x}]$ was already small enough. Pessimism would have also been introduced if the wall was not perfectly vertical, as the distance set associated would be tilted and therefore non-representable with axis-aligned boxes.

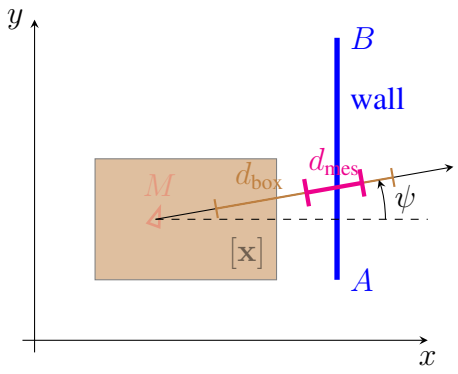
For the sake of clarity, in this example, the value of ψ was known precisely. In reality, this is not the case: this value is obtained using the robot's heading, which is never entirely sure. This uncertainty is easily taken into account by bounding it inside an interval. Some pessimism is then added again, but everything works in the same way.

4.3.2 Localization example

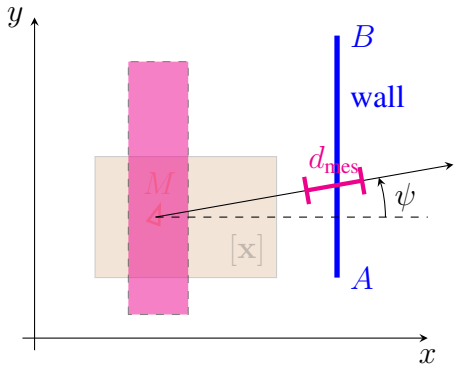
Context

In order to illustrate how all these notions can be used, an example of localization is now presented. We will see that the formalization of the problem can lead to a set inversion problem, and then how we can use the contractor $\mathcal{C}_{\text{dist\_wall}}$ to solve it. The approach can be compared with [142], where the localization problem is also translated into a CSP with a set-membership resolution.

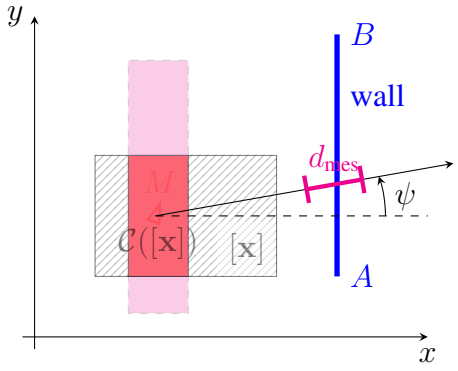
So, take again a robot measuring its distance to a wall. This time, we assume that several walls are forming a polygon. This polygon is not necessarily convex. We limit the example to polygons, as only the contractor associated with the distance to segments has been presented. However, it is entirely conceivable to extend the application to other geometric shapes using associated contractors. On the other hand, a new step has to be reached to deal with unstructured environments, but this is not the topic. This approach would also work with open shapes,



(a) The robot at M measures the distance d_{mes} depicted in magenta. Its current estimated position is the brown box $[\mathbf{x}]$. The associated distance interval is computed (forward step), and is represented in brown by d_{box}.



(b) The interval d_{box} is contracted with d_{mes}, and this new interval (equal to d_{mes} in this case) is used to compute back a position box (backward step), painted in magenta.



(c) Finally, the initial box $[\mathbf{x}]$ is contracted using the magenta box, giving the red box $\mathcal{C}([\mathbf{x}])$ as result.

Figure 4.10: The different steps of a robot contracting its position using the distance $[d_{\text{mes}}]$ to a wall represented by the segment $[A, B]$.

assuming that we can detect when the distance sensor returns the equivalent of an infinity distance (or equivalently, no obstacle). Then, another contractor may be associated with such a constraint: not detecting an obstacle can constitute a piece of information for localization.

The objective is to localize the robot in this known and structured environment. Once again, we assume the robot to measure its heading. Actually, it is possible to find back this angle

in addition to the position with this approach, as in [86]. For the moment, we only search to characterize the robot's position in the horizontal plan, constituted by two variables: x and y. Figure 4.11 summarizes the situation.

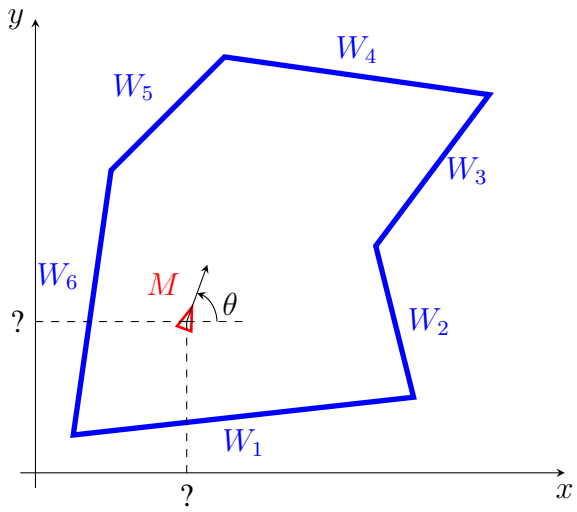


Figure 4.11: A robot in a structured environment is localizing itself.

This kind of problem has already been widely studied, and different approaches are possible, as in [44, 100]. However, here, we make one additional assumption that makes some approaches unfit: sensors return only a few measurements, four at maximum (one in each direction of the robot).

Formalization

If we now formalize the problem, we need to find a link between the measurements and the unknown variables (the position of the robot). It is much easier to express the expected measurements from the position rather than the inverse. The reason for that is the non-injectiveness of the function $\mathbf{f}$ that returns the distance to a wall from the position of the robot. Indeed, there is only one possible distance to a wall (with a given direction) from one position. The inverse is not valid: in most cases, knowing that the robot is at a distance of one meter to a wall is not enough to know where the robot is; there are several possibilities, even an infinity of possibilities. The function $\mathbf{f}$ is of the form:

$$\begin{array}{rccl} \mathbf{f}: & \mathbb{R}^2 & \to & \mathbb{R} \\ & (x,y) & \mapsto & \text{dist}((x,y), \text{wall}). \end{array}$$

Thus, the problem is a set inversion problem: from an image set obtained by measurements (distances), we want to know the associated antecedent set by the function $\mathbf{f}$. These measurements can be seen as constraints on the position of the robot. Each new measurement brings new information that allows characterizing more precisely the position set.

To complete the localization of the robot, the contractor $\mathcal{C}_{\text{dist\_wall}}$ is used. It is a little more complex than the example with one wall (as in Figure 4.10) because here, we cannot know which

wall has been detected by the sensor. For instance, if the sensor was oriented east, there are four walls likely to match (W_1 to W_4). So there is no choice to iterate over the walls and create a subcontractor for each one. These subcontractors are associated with the distances to each wall. Then, a possibility is to take the union of all these subcontractors. It consists of computing the solution set for each wall and then returning the union of all these sets. It works well, and the found solution is correct.

However, in some specific cases, it is possible to do better. Indeed, if there are several walls in front of the sensor, one behind the other, then all the walls are considered, whereas only the closer is detectable. Figure 4.12 gives an example of this particular case. The distance measured by the robot is the one to the wall W_2, but there may have ambiguity with the wall W_4. It happens because the solution set of the wall W_4 is computed independently, without considering the existence of the other walls. In order to remove these supplementary solutions, the other possibility is not to create each subcontractor, but just one which is associated with the minimum of the distances to each wall. In this way, when a wall hides another, only the closer is taken into account as the associated distance is the smallest. Specifically, still in the example of the Figure 4.12, the wrong solution which is represented transparent is removed: the minimum distance associated with this position is the one to the wall W_2 (and not W_4), which is not compatible with the measured distance d.

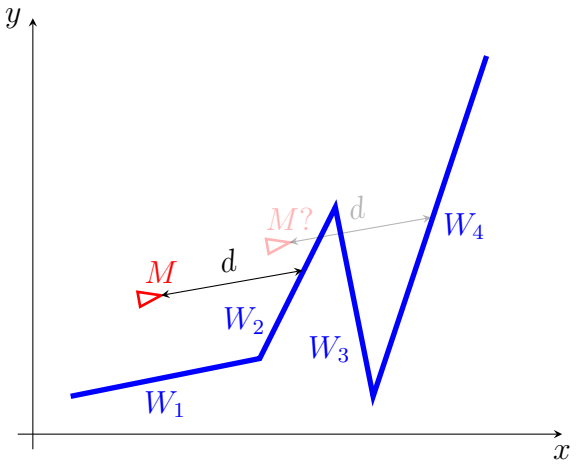


Figure 4.12: With walls in a sawtooth shape, some impossible solutions can be kept.

Examples

Finally, we can give the possible positions for the robot from a list of walls and one distance measurement. An example with a distance measurement to the east of the robot is given in Figure 4.13. Note that possible positions that are outside the room have been removed as we know the robot is in the room.

To go further, we may want to consider several measurements that ideally come from different directions to have a more refined localization. In this case, to combine the different solution

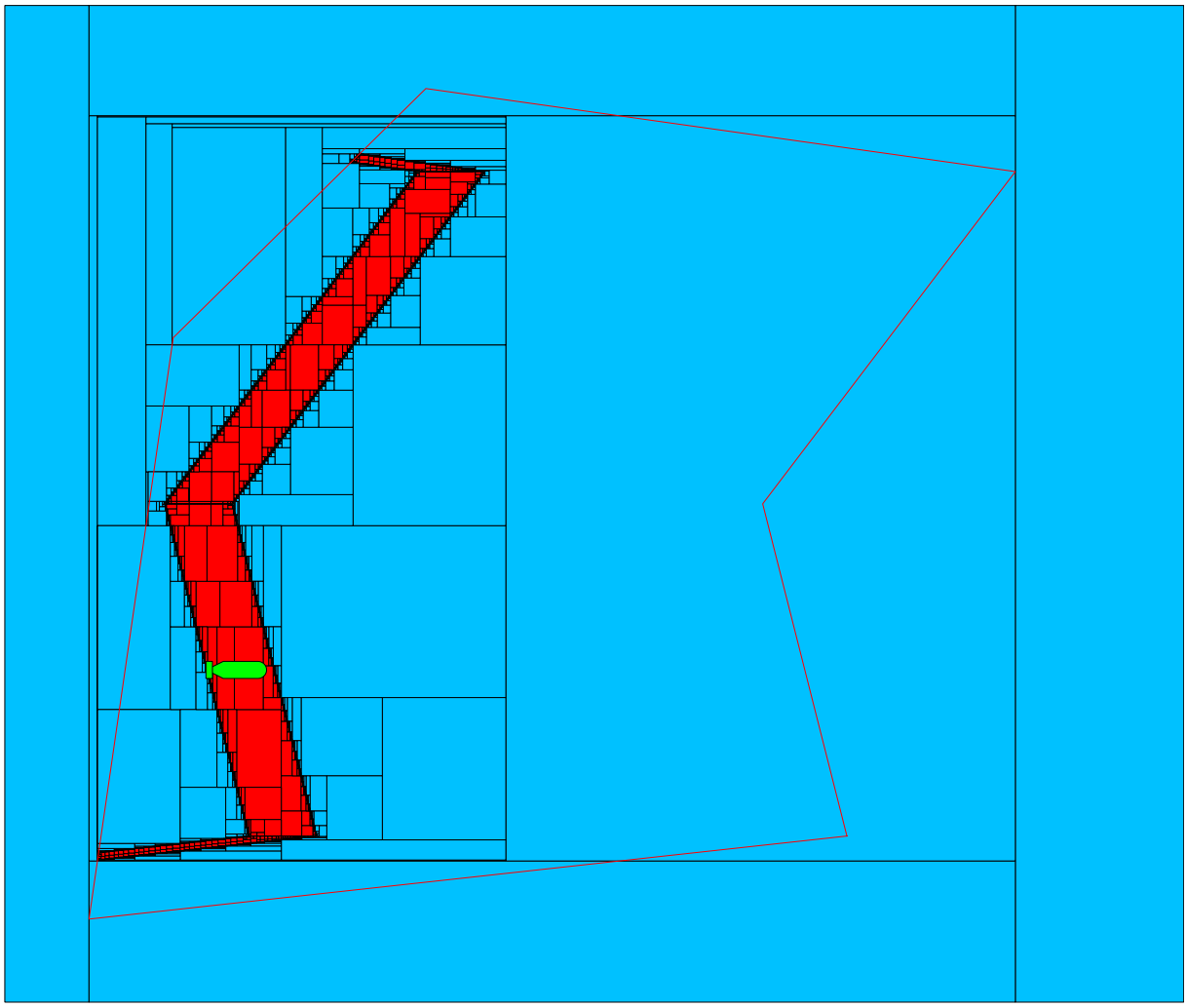


Figure 4.13: Localization using a distance measurement to the east. The compatible positions are in red.

sets, a quick way is to take the intersection of all these sets. If the robot has not moved, and if all the distances are correct, then the result will be satisfying, as in Figure 4.14 where only two range measurements have been used. However, the risk of using an intersection is that as soon as one returned distance is wrong, the localization may fail due to an empty solution set. These potential wrong measurements that can mislead a whole approach are called *outliers*. This problem has already been addressed, as in [79], especially using the relaxed intersection, which is particularly robust to outliers. The latter have been detailed in Section 2.3.3.

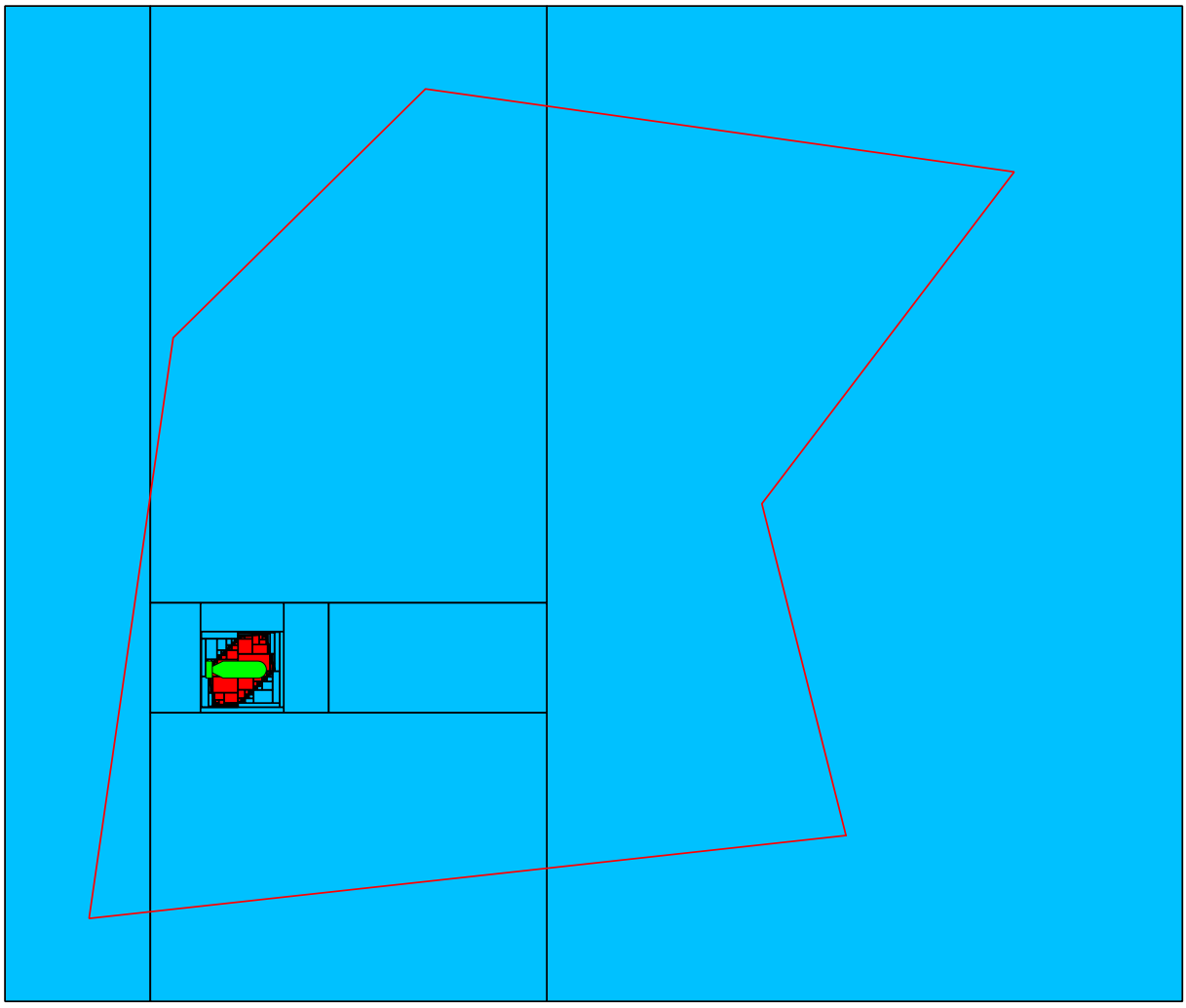


Figure 4.14: Localization using both a distance measurement to the east and to the north. The red area shows that the robot is well localized.

Silence contractor: test case

The silence contractor presented in Section 4.2.2 has been designed similarly. Then it has been tested in simulation and with an actual robot in a pool. The sonar used is a Tritech Micron (MSIS) fixed on a BlueROV2 of BlueRobotics (see Figure 4.15). This sonar is rather small: less than 8 cm in height and 7 cm in diameter. It has a vertical beamwidth of $35°$ and was set for a range of 6 meters. The IMU integrated into the flight controller (Pixhawk) returns the heading. The shape of the pool is a rectangle of approximately 5.5 x 2.75 meters. It is not a rigid pool, so the map cannot be absolutely exact. On top of that, the water level is very low (less than 1 m). The lack of depth creates more multipath coming from both the surface and the bottom of the pool. A picture of the experimentation is shown in Figure 4.16.

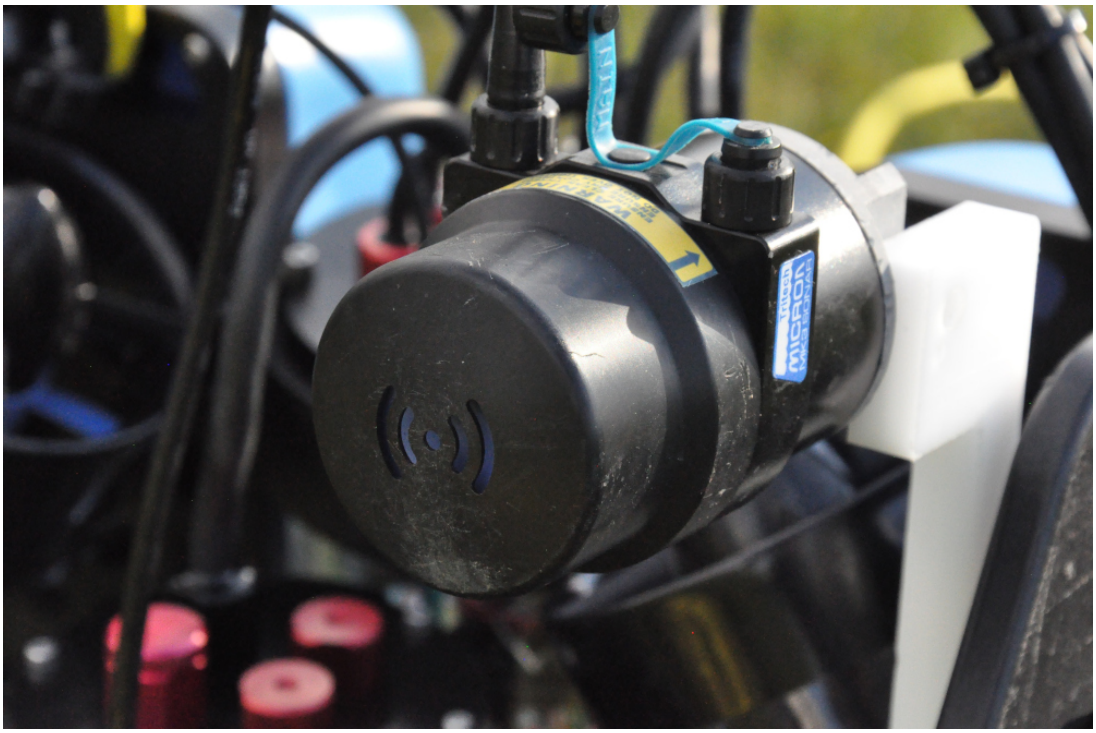


Figure 4.15: Picture of the Tritech Micron mechanical scanning sonar (MSIS) fixed on the robot.

In the beginning, the robot only knows it is in the pool (the initial box covers the whole pool). Then each information that comes from a beam sent perpendicularly to a wall is used to contract the current estimation of the robot's position. After a sonar turn, we obtain the estimation of the localization presented in Figure 4.17. The map of the pool is the green rectangle. The orange one is the contracted box corresponding to the estimation of the position. Sonar data are represented in gray level, with only the used ones, which are those perpendicular to walls. They are in the frame centered in the estimated position box. Note that this choice is only for the display, the whole box must be considered as a result, and there is no information on which position within the box is the closest to the actual one. This result is convincing concerning reliability since the actual position of the robot is never removed from possible positions. However, the precision is questionable, especially in the x axis.

Finally, even if our constraint based on the silence gives us the expected reliability, we would like to have a more precise result. Indeed, the data of Figure 4.17 seems to be sufficient for better localization. Therefore, the silence constraint is not hard enough, and we have to consider other constraints. It is possible using the fuzzy approach presented in the next chapter.

4.4 Conclusion

In this chapter, intuition on the behavior of acoustic beams has been given. An interval analysis approach has been then studied to deal with uncertainties and outliers. Thus, a reliable constraint should be found: the absence of received signal when emitting perpendicularly to a wall seems

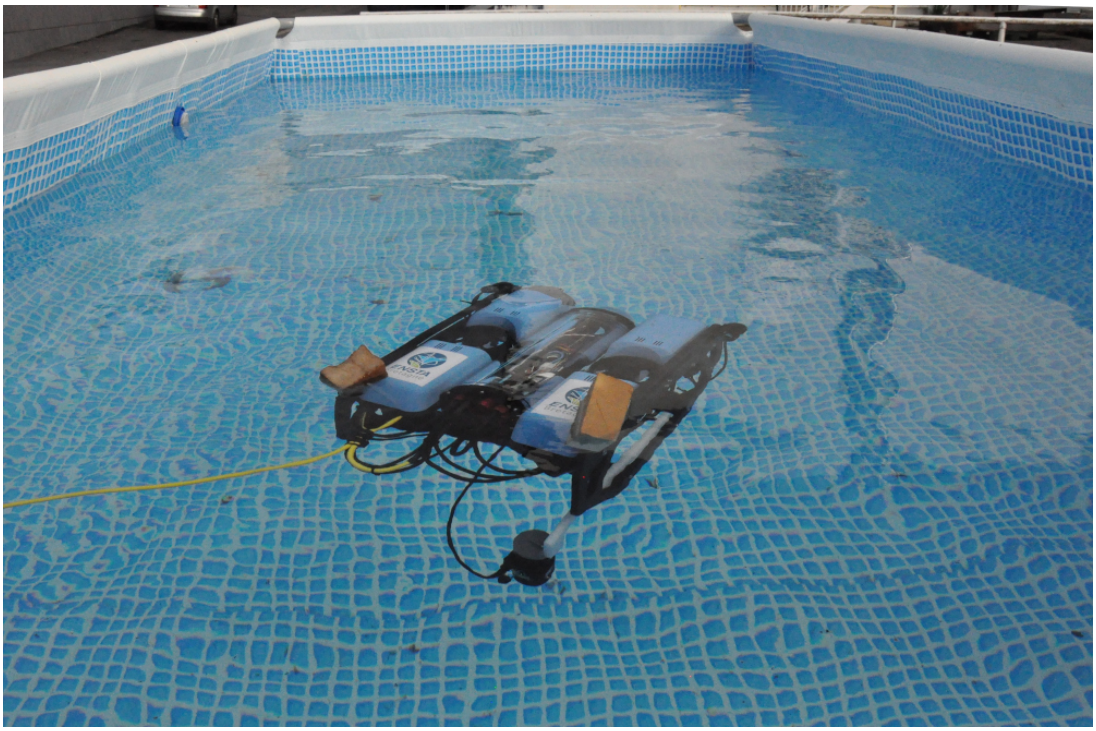


Figure 4.16: Picture of the ROV equipped with the sonar in the pool of ENSTA Bretagne.

to correspond to the absence of an obstacle. So the idea is to use the absence of signal as reliable information. A set-membership approach is perfectly adapted to consider this constraint.

The design of a contractor has been then given with examples of simulated robot localization. The contractor using the silence approach has been tested on real data, leading to a guaranteed localization. The interests of using intervals in the context of a localization problem are multiple. Indeed, intervals can efficiently deal with non-linearities and represent uncertainties. No approximations nor linearizations are necessary contrary to usual methods such as the Kalman filter. Interval analysis perfectly suits the constraint programming approach, providing a guarantee of the results. On top of that, it also provides tools to handle outliers that frequently occur with real data.

Finally, localization problems are crucial for many applications, and the issue can still be considered open as soon as no external reference like a Global Navigation Satellite System is available. Mainly, it concerns indoor and underwater environments, where hazardous and sensitive tasks can be required. The underwater environment is especially subject to uncertainties and outliers. In front of both of these sources of errors and the cost of underwater vehicles, the interest of interval analysis approaches' reliability takes on its full meaning.

Nevertheless, we can still reproach interval analysis for lacking suppleness: it is not possible to consider *soft constraints*, *i.e.* constraints that are not always true. Sometimes, we would like to add some granularity to the approach such that the result can be more accurate at the risk of violating a soft constraint. The next chapter will go further in the localization approach by considering *fuzzy sets*, which can deal with this desired granularity.

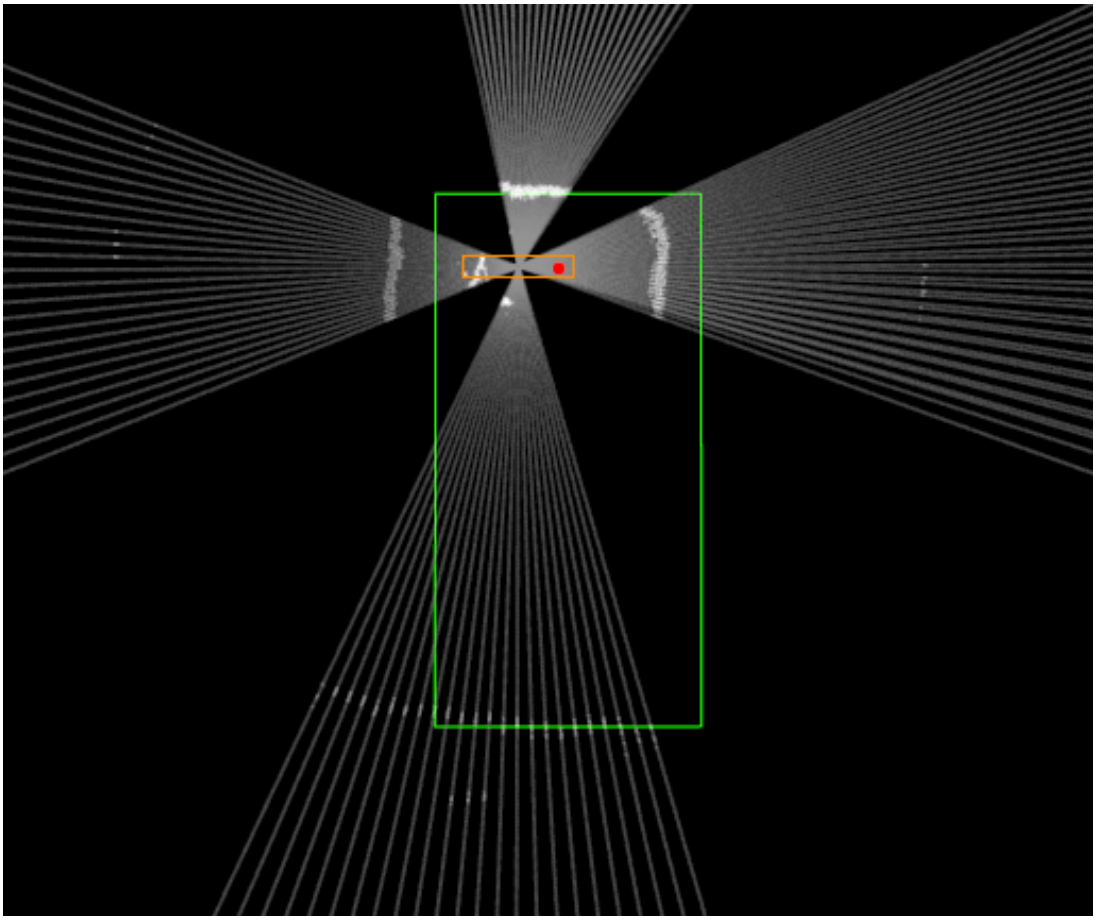


Figure 4.17: Result of the localization in the pool using silence contractor. After acquiring data from every side, the robot is localized without good precision, but the actual position (in red) is well included in the estimated position box (in orange).

CHAPTER 5

Underwater localization

Contents

5.1 Introduction

This chapter shows a new localization approach using fuzzy logic to go further than the previous chapter. Fuzzy logic, and especially fuzzy sets, bring the lacking granularity of the previous method. Indeed, fuzzy sets allow for dealing with a more complex representation and combination of the acquired granules of knowledge. Thus, soft constraints can be considered and weighted with respect to the significance we want to put on them. It allows prioritizing the constraints to maximize the number of satisfied ones.

The principle is to use the score function presented in Section 2.4.2 to combine the granules. The obtained fuzzy set is then characterized by its α-cuts, independently computed with interval analysis. These α-cuts are different levels of representation of the result.

Therefore, after presenting this new α-cut approach, this chapter addresses the localization problem to show the benefits of the new method. Simulations and experiments will illustrate the consistency of the process.

5.2 α-cut approach

The proposed approach consists in handling fuzzy sets that represent our desired combination of granules. Fuzzy sets are built with the score function. A score function provides much freedom to combine granules. It facilitates the implementation of complex combinations and allows giving different weights to each granule.

Once this score function is defined, it describes a fuzzy set representing the result of the combination of the granules. Then, when we want to access the information they contain, the objective is to characterize their α-cuts to approximate the researched set using interval analysis.

5.2.1 A new formulation of the α-cut principle

Link between score function and α-cut

Using score functions defined in Section 2.4.2, an α-cut of a fuzzy set $\mathbb{X}$ is a crisp set which is defined by

$$\begin{aligned} \mathbb{X}_\alpha &= \{\mathbf{x} \in \mathbf{\Omega} \mid \mu_{\mathbb{X}}(\mathbf{x}) \geq \alpha\} \\ &= \left\{\mathbf{x} \in \mathbf{\Omega} \mid \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha\right\}. \end{aligned} \tag{5.1}$$

Therefore, as it is a set inversion problem, an inner and outer approximation of this set can be computed using interval analysis. The following example illustrates how we can define score functions and threshold α to combine granules.

Example 5.2.1. Table 5.1 provides some examples of membership functions, thresholds α and the corresponding α-cuts.

	$\mu_{\mathbb{X}}(\mathbf{x})$	α	$\mathbb{X}_\alpha$
(i)	$\frac{\zeta^1(\mathbf{x})+\zeta^2(\mathbf{x})}{2}$	1	$\mathbb{Z}_1 \cap \mathbb{Z}_2$
(ii)	$\frac{\zeta^1(\mathbf{x})+\zeta^2(\mathbf{x})}{2}$	0.5	$\mathbb{Z}_1 \cup \mathbb{Z}_2$
(iii)	$\frac{\zeta^1(\mathbf{x})+2\zeta^2(\mathbf{x})+\zeta^3(\mathbf{x})+\zeta^4(\mathbf{x})}{5}$	0.5	$(\mathbb{Z}_1 \cap \mathbb{Z}_2) \cup (\mathbb{Z}_2 \cap \mathbb{Z}_3) \cup (\mathbb{Z}_2 \cap \mathbb{Z}_4) \cup (\mathbb{Z}_1 \cap \mathbb{Z}_3 \cap \mathbb{Z}_4)$
(iv)	$\frac{1}{m}\sum_{j=1}^{m} \zeta^j(\mathbf{x})$	$1-\frac{q}{m}$	$\bigcap_{j=1}^{m}{}^{\{q\}}\, \mathbb{Z}_j$
(v)	$\min\left(\zeta^1(\mathbf{x}), \frac{1}{m}\sum_j \zeta^j(\mathbf{x})\right)$	$1-\frac{q}{m}$	$\mathbb{Z}_1 \cap \bigcap_{j=1}^{m}{}^{\{q\}}\, \mathbb{Z}_j$

Table 5.1: Examples of membership functions with thresholds α and their corresponding α-cuts.

Line (i) should be understood as follows:

$$\begin{aligned}\frac{\zeta^1(\mathbf{x})+\zeta^2(\mathbf{x})}{2} \geq 1 &\iff \zeta^1(\mathbf{x}) = 1 \text{ and } \zeta^2(\mathbf{x}) = 1\\ &\iff \mathbf{x} \in \mathbb{Z}_1 \text{ and } \mathbf{x} \in \mathbb{Z}_2\\ &\iff \mathbf{x} \in \mathbb{Z}_1 \cap \mathbb{Z}_2.\end{aligned}$$

To understand the case (iii), the idea is to compute all grades of membership for all possible vectors $(\zeta^1(\mathbf{x}), \ldots, \zeta^4(\mathbf{x}))$. There are $2^4 = 16$ possibilities, and we represent them in Table 5.2.

		$\mathbb{Z}_1\mathbb{Z}_2$			
		00	01	11	10
$\mathbb{Z}_3\mathbb{Z}_4$	00	0	$2/5$	$3/5$	$1/5$
	01	$1/5$	$3/5$	$4/5$	$2/5$
	11	$2/5$	$4/5$	1	$3/5$
	10	$1/5$	$3/5$	$4/5$	$2/5$

$$\mu_{\mathbb{X}}(\mathbf{x}) = \frac{\zeta^1(\mathbf{x})+2\zeta^2(\mathbf{x})+\zeta^3(\mathbf{x})+\zeta^4(\mathbf{x})}{5}$$

Table 5.2: Grades of membership of all the possible vectors of granule to the fuzzy set described by the membership function of the case (iii).

In this table, each pair of binary digits indicates the considered belonging of a vector $\mathbf{x}$ to the associated sets. For instance, in the column "01" under the "$\mathbb{Z}_1\mathbb{Z}_2$" heading, we have $\zeta^1(\mathbf{x}) = 0$ and $\zeta^2(\mathbf{x}) = 1$. Thus, for each case of the table, we have the result of the different values for $\mu_{\mathbb{X}}(\mathbf{x})$. We choose this representation since when computing the associated α-cut table, it directly corresponds to a Karnaugh table.

An example is given in Table 5.3 for $\alpha \geq 0.5$. This table is the same as the previous one, but the threshold $\alpha = 0.5$ has been used to binarize the results; this is why it contains only 0 and 1.

It is called a *Karnaugh table*, which is a graphical method to find the logical function associated with a given truth table. The method is the following: the adjacent 1s are grouped by squared or rectangular boxes, such that the number of grouped values is equal to a power of 2. These groups are represented in color in the table. The goal is to minimize the number of groups to obtain the most straightforward logical expression. Finally, the logical function is the union of each group, and each group is the intersection of the shared variables. For instance, the green box groups $\mathbb{Z}_2$ and $\mathbb{Z}_4$ (in both concerned columns, we have $\zeta^2(\mathbf{x}) = 1$ and in both concerned lines, we have $\zeta^4(\mathbf{x}) = 1$). Finally, we find back the expression written in Table 5.1:

$$\mathbb{X}_\alpha = \underbrace{(\mathbb{Z}_1 \cap \mathbb{Z}_2)}_{\text{blue box}} \cup \underbrace{(\mathbb{Z}_2 \cap \mathbb{Z}_3)}_{\text{yellow box}} \cup \underbrace{(\mathbb{Z}_2 \cap \mathbb{Z}_4)}_{\text{green box}} \cup \underbrace{(\mathbb{Z}_1 \cap \mathbb{Z}_3 \cap \mathbb{Z}_4)}_{\text{red box}}. \tag{5.2}$$

$\mathbb{Z}_3$ \ $\mathbb{Z}_1\mathbb{Z}_2$	00	01	11	10
00	0	0	1	0
01	0	1	1	0
11	0	1	1	1
10	0	1	1	0

$$\mu_{\mathbb{X}}(\mathbf{x}) \geq 0.5$$

Table 5.3: Karnaugh table associated to the constraint $\mu_{\mathbb{X}}(\mathbf{x}) \geq 0.5$ of the case (iii).

The obtained expression is called the *Disjunctive Normal Form* (DNF). Note that this is not the only possibility; some factorizations could yield a shorter expression as

$$\mathbb{X}_\alpha = (\mathbb{Z}_2 \cap (\mathbb{Z}_1 \cup \mathbb{Z}_3 \cup \mathbb{Z}_4)) \cup (\mathbb{Z}_1 \cap \mathbb{Z}_3 \cap \mathbb{Z}_4)\,. \tag{5.3}$$

This method, using the Karnaugh table, may yield an expression with a length that is exponential in m even after factorization routines, depending on $\mu_{\mathbb{X}}(\mathbf{x})$. This is why we prefer here to work directly on the expression of $\mu_{\mathbb{X}}(\mathbf{x})$ instead of the set expression. Since the expression for μ admits more symbols ($+, -, \exp, \sin, \max, \min, \ldots$) than for the expression for $\mathbb{X}_\alpha$ ($\cup, \cap, \bar{}$), we limit the combinatorial length and the corresponding evaluation.

Complementary of an α-cut

The score function allows us to combine the granules as wanted and create a fuzzy set whose α-cuts represent the logical expression we want to characterize. So we want to find an approximation of these α-cuts, and tools from interval analysis are actually adapted. However, to obtain

the inner approximation in addition to the outer one, to obtain bracketing of the solution set, we need to compute the complementary of the studied α-cut. Obtaining the inner approximation is also essential to obtain an efficient interval algorithm. It allows to reduce the number of required bisections and, thus, boxes in comparison with only the outer approximation.

So the objective is to find the expression of the piecewise constant membership function such that it defines the complementary of α-cuts of an other similar membership function.

Theorem 5.2.1 (De Morgan rule)**.** *The complementary set of the α-cut*

$$\mathbb{X}_\alpha \overset{(5.1)}{=} \left\{\mathbf{x} \mid \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha\right\}$$

is the α-cut defined by

$$\bar{\mathbb{X}}_\alpha = \left\{\mathbf{x} \mid \bar{\sigma}\left(\bar{\zeta}^1(\mathbf{x}), \ldots, \bar{\zeta}^m(\mathbf{x})\right) > \bar{\alpha}\right\}, \tag{5.4}$$

where

$$\begin{aligned} \bar{\zeta}^j &= 1 - \zeta^j \\ \bar{\sigma}\left(a_1, \ldots, a_m\right) &= 1 - \sigma\left(1 - a_1, \ldots, 1 - a_m\right) \\ \bar{\alpha} &= 1 - \alpha. \end{aligned} \tag{5.5}$$

Proof. First, let us note that since the function $\bar{\sigma}$ takes only a finite number of values, the strict inequality ($>$) can always be transformed into a non-strict inequality ($\geq$). Since $\mathbb{X}_\alpha = \{\mathbf{x} \mid \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha\}$, we have

$$\begin{aligned} \mathbf{x} \in \bar{\mathbb{X}}_\alpha &\iff \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) < \alpha \\ &\iff 1 - \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) > 1 - \alpha \\ &\iff 1 - \sigma\left(1 - \bar{\zeta}^1(\mathbf{x}), \ldots, 1 - \bar{\zeta}^m(\mathbf{x})\right) > \bar{\alpha} \\ &\iff \bar{\sigma}\left(\bar{\zeta}^1(\mathbf{x}), \ldots, \bar{\zeta}^m(\mathbf{x})\right) > \bar{\alpha}. \end{aligned}$$

We now have to check that $\bar{\sigma}$ is a membership function, *i.e.* with the form of (2.48).
(i) We have $\bar{\sigma}\left(0, \ldots, 0\right) = 1 - \sigma\left(1 - 0, \ldots, 1 - 0\right) = 0$.
(ii) We have $\bar{\sigma}\left(1, \ldots, 1\right) = 1 - \sigma\left(1 - 1, \ldots, 1 - 1\right) = 1$.
(iii) We now check the monotonicity of $\bar{\sigma}$:

$$\begin{aligned} \left(a_1, \ldots, a_m\right) \leq \left(b_1, \ldots, b_m\right) &\implies \left(1 - a_1, \ldots, 1 - a_m\right) \geq \left(1 - b_1, \ldots, 1 - b_m\right) \\ &\implies \sigma\left(1 - a_1, \ldots, 1 - a_m\right) \geq \sigma\left(1 - b_1, \ldots, 1 - b_m\right) \\ &\iff 1 - \sigma\left(1 - a_1, \ldots, 1 - a_m\right) \leq 1 - \sigma\left(1 - b_1, \ldots, 1 - b_m\right) \\ &\iff \bar{\sigma}\left(a_1, \ldots, a_m\right) \leq \bar{\sigma}\left(b_1, \ldots, b_m\right). \end{aligned}$$

□

Example 5.2.2. Consider again the set $\mathbb{X}_\alpha = \mathbb{Z}_2 \cap (\mathbb{Z}_1 \cup \mathbb{Z}_3 \cup \mathbb{Z}_4) \cup (\mathbb{Z}_1 \cap \mathbb{Z}_3 \cap \mathbb{Z}_4)$ presented in (5.3), and defined by

$$\frac{\zeta^1(\mathbf{x}) + 2\zeta^2(\mathbf{x}) + \zeta^3(\mathbf{x}) + \zeta^4(\mathbf{x})}{5} \geq 0.5. \tag{5.6}$$

Note that the corresponding score function is the same as in Example 2.4.3. The complementary set $\bar{\mathbb{X}}_\alpha$ is defined by

$$\begin{aligned}
&1 - \frac{1 - \bar{\zeta}^1(\mathbf{x}) + 2\left(1 - \bar{\zeta}^2(\mathbf{x})\right) + 1 - \bar{\zeta}^3(\mathbf{x}) + 1 - \bar{\zeta}^4(\mathbf{x})}{5} > 1 - 0.5 \\
\iff &\frac{5 - 1 + \bar{\zeta}^1(\mathbf{x}) - 2 + 2\bar{\zeta}^2(\mathbf{x}) - 1 + \bar{\zeta}^3(\mathbf{x}) - 1 + \bar{\zeta}^4(\mathbf{x})}{5} > 0.5 \\
\iff &\frac{\bar{\zeta}^1(\mathbf{x}) + 2\bar{\zeta}^2(\mathbf{x}) + \bar{\zeta}^3(\mathbf{x}) + \bar{\zeta}^4(\mathbf{x})}{5} \geq 0.5.
\end{aligned}$$

The following example illustrates a situation where the relaxed intersection is involved.

Example 5.2.3. Consider again the set $\mathbb{Z}_1 \cap \bigcap_{j=1}^{m}{}^{\{q\}} \mathbb{Z}_j$ presented in Example 5.2.1 and defined by

$$\min\left(\zeta^1(\mathbf{x}), \frac{1}{m}\sum_{j=1}^{m}\zeta^j(\mathbf{x})\right) \geq 1 - \frac{q}{m}. \tag{5.7}$$

The complementary set is defined by

$$\begin{aligned}
&1 - \min\left(1 - \bar{\zeta}^1(\mathbf{x}), \frac{1}{m}\sum_{j=1}^{m}\left(1 - \bar{\zeta}^j(\mathbf{x})\right)\right) > 1 - \left(1 - \frac{q}{m}\right) \\
\iff &1 - \min\left(1 - \bar{\zeta}^1(\mathbf{x}), 1 - \frac{1}{m}\sum_{j=1}^{m}\bar{\zeta}^j(\mathbf{x})\right) > \frac{q}{m} \\
\iff &1 + \max\left(-1 + \bar{\zeta}^1(\mathbf{x}), -1 + \frac{1}{m}\sum_{j=1}^{m}\bar{\zeta}^j(\mathbf{x})\right) > \frac{q}{m} \\
\iff &\max\left(\bar{\zeta}^1(\mathbf{x}), \frac{1}{m}\sum_{j=1}^{m}\bar{\zeta}^j(\mathbf{x})\right) \geq \frac{q+1}{m}.
\end{aligned}$$

Finally, from a score function and characteristic functions of granules, we can define a specific class of fuzzy sets whose α-cuts and their complementaries are characterized by inequalities. Then, it is possible to approximate these α-cuts using interval analysis tools, with an inner and an outer set. The contractors used for this task are presented in the next section.

5.2.2 Resolution using interval contractors

We recall here the expression of an α-cut of the fuzzy set $\mathbb{X}$, as introduced in (5.1):

$$\begin{aligned}
\mathbb{X}_\alpha &= \{\mathbf{x} \in \boldsymbol{\Omega} \mid \mu_{\mathbb{X}}(\mathbf{x}) \geq \alpha\} \\
&= \left\{\mathbf{x} \in \boldsymbol{\Omega} \mid \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha\right\},
\end{aligned}$$

where σ is a score function and ζ^j is the characteristic function of the granule $\mathbb{Z}_j$. We consider the problem of finding an efficient contractor for $\mathbb{X}_\alpha$. This problem is firstly addressed in the specific simpler case where granules $\mathbb{Z}_j$ are intervals, denoted by $[z](j)$. Then, the $\mathbb{Z}_j$ are considered to be boxes, denoted by $[\mathbf{z}](j)$. Finally, the general case will be handled where the granules are any subset of $\mathbb{R}^n$, with the assumption that a contractor for each of these granules is available.

Interval-based α-cuts

So firstly, we consider only the case where granules are intervals. We want to compute the smallest interval, which encloses all $x \in [x]$ such that $x \in \mathbb{X}_\alpha$. Note that computing this smallest interval amounts to finding the optimal contractor for the set $\mathbb{X}_\alpha$. We denote this contractor by $\mathcal{C}^{\text{interval}}_{\sigma,\alpha}$ since the score function σ and the scalar α are sufficient to define the set $\mathbb{X}_\alpha$, from the knowledge of characteristic functions ζ^j. We now describe a procedure (taken from [103]) that solves the problem with a complexity of m^2. In what follows, $[z](1:m)$ denotes the list of intervals $\{[z](1), \ldots, [z](m)\}$, which are the granules. Algorithm 1 describes this procedure.

Algorithm 1: Contractor $\mathcal{C}^{\text{interval}}_{\sigma,\alpha}$

Result: Contract the interval $[x]$ for the constraint $\sigma\left(\zeta^1(x), \ldots, \zeta^m(x)\right) \geq \alpha$
Input: $[x]$, $[z](1:m)$
Output: $[x]$

1 Store all endpoints of $[z](1), \ldots, [z](m)$ and $[x]$ inside a list $\mathcal{L}$
2 Remove elements of $\mathcal{L}$ that are not inside $[x]$
3 Sort $\mathcal{L}$ in ascending order
4 Take the smallest element a of $\mathcal{L}$ such that $\sigma(\zeta^1(a), \ldots, \zeta^m(a)) \geq \alpha$
5 **if** *no a has been found* **then**
6 ⌊ **return** $\emptyset$
7 Take the greatest element b of $\mathcal{L}$ such that $\sigma(\zeta^1(b), \ldots, \zeta^m(b)) \geq \alpha$
8 **return** $[a, b]$

Here are comments on each line of this algorithm:

Step 1. The algorithm first puts all endpoints of the $[z](j)$'s into a list $\mathcal{L}$ (that is lower and upper bounds). It also stores the endpoints of $[x]$. At this step, the list $\mathcal{L}$ has $2m + 2$ elements.

Step 2. Each element that is not in $[x]$ does not influence the result and thus is removed.

Step 3. The list $\mathcal{L}$ is sorted in an ascending manner. We thus get a sorted sequence $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_\eta$ where $\ell_1 = x^-$ and $\ell_\eta = x^+$.

Step 4. The first ℓ_k such that $\sigma(\zeta^1(a), \ldots, \zeta^m(a)) \geq \alpha$ corresponds to the lower bound of $\mathbb{X}$.

Step 5. If no such an a exists, then $\mathbb{X}$ is empty.

Step 6. In that case, the algorithm returns an empty interval.

Step 7. The upper bound of $\mathbb{X}$ is computed similarly.

Step 8. The interval hull of $\mathbb{X}$ is returned.

As shown in [103], it is possible to get a complexity of $O(n\log(n))$.

Box-based α-cuts

We have now a procedure for the case where granules are intervals. We can extend it for finding a contractor for $\mathbb{X}_\alpha$ in the case where granules are boxes and are denoted by $[\mathbf{z}](j)$. Finding the optimal contractor for $\mathbb{X}_\alpha$ is known to be an NP-hard problem [103]. In order to build a good contractor denoted by $\mathcal{C}^{\text{box}}_{\sigma,\alpha}$, we project the problem with respect to each of the n axis and then call n times the contractor $\mathcal{C}^{\text{interval}}_{\sigma,\alpha}$ described in the previous subsection. The procedure is based on the following theorem.

Theorem 5.2.2. *Consider m boxes $[\mathbf{z}](1),\ldots,[\mathbf{z}](m)$ such that $[\mathbf{z}](j)=[z_1](j)\times\cdots\times[z_n](j)$. Fix $\mathbf{x}=(x_1,\ldots,x_n)$. If σ is a score function and $i\in\{1,\ldots,n\}$, we have*

$$\sigma\left(\zeta^1(\mathbf{x}),\ldots,\zeta^m(\mathbf{x})\right)\leq\sigma\left(\zeta_i^1(x_i),\ldots,\zeta_i^m(x_i)\right), \tag{5.8}$$

where ζ^j and ζ_i^j are the characteristic functions of $[\mathbf{z}](j)$ and $[z_i](j)$, respectively.

Proof. Consider a vector $\mathbf{x}=(x_1,\ldots,x_n)$ and a box $[\mathbf{z}]=[z_1]\times\cdots\times[z_n]$ with a characteristic function ζ. We have

$$\zeta(\mathbf{x})=\zeta_1(x_1)\cdot\zeta_2(x_2)\cdot\ldots\cdot\zeta_n(x_n) \tag{5.9}$$

and for all i, $\zeta_i(x_i)\in[0,1]$; thus, we have $\zeta(\mathbf{x})\leq\zeta_i(x_i)$. Since σ is monotonic, we get the inequality to be proved. □

An illustrative example is now proposed to understand this theorem better.

Example 5.2.4. Consider the situation of Figure 5.1 where the membership function is

$$\mu_{\mathbb{X}}(\mathbf{x})=\sigma\left(\zeta^1(\mathbf{x}),\zeta^2(\mathbf{x}),\zeta^3(\mathbf{x})\right)=\frac{1}{3}\sum_{j=1}^{3}\zeta^j(\mathbf{x}). \tag{5.10}$$

For the three points $\mathbf{a},\mathbf{b},\mathbf{c}$, we obtain the results given by Table 5.4.

	$\mathbf{a}$	$\mathbf{b}$	$\mathbf{c}$
$\sigma(\zeta^1,\zeta^2,\zeta^3)$	$2/3$	0	$1/3$
$\sigma(\zeta_1^1,\zeta_1^2,\zeta_1^3)$	$2/3$	$1/3$	$2/3$
$\sigma(\zeta_2^1,\zeta_2^2,\zeta_2^3)$	$2/3$	$2/3$	$2/3$

Table 5.4: Score values for the three points $\mathbf{a},\mathbf{b},\mathbf{c}$.

For instance, the last cell of the table is the result of $\sigma\left(\zeta_2^1(\mathbf{c}),\zeta_2^2(\mathbf{c}),\zeta_2^3(\mathbf{c})\right)$. The subscript on functions ζ_2 indicates that we study only the second dimension. So in the Figure 5.1 it is the

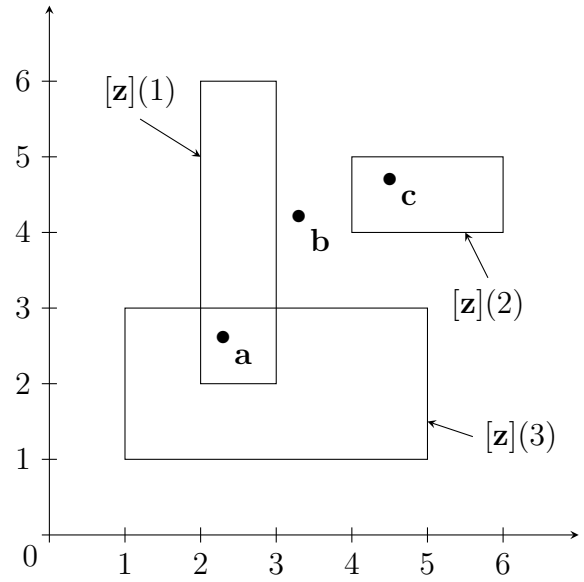


Figure 5.1: Only **a** belongs to at least two of the three boxes.

ordinate axis. Then we check if the ordinate of **c** (denoted by c_2) belongs to the ordinates of the three different boxes, that is if $c_2 \in [z_2](j)$, for $j \in \{1, 2, 3\}$. It amounts to project the point **c** and the three boxes onto the ordinate axis. Here c_2 belongs to $[z_2](1)$ and $[z_2](2)$, but not to $[z_2](3)$. We thus have $\zeta_2^1(\mathbf{c}) = \zeta_2^2(\mathbf{c}) = 1$ and $\zeta_2^3(\mathbf{c}) = 0$. As the membership function is the sum of these three terms divided by three, the result is well $2/3$.

We can observe that the inequality (5.8) is always satisfied, but equality is not.

Theorem 5.2.3. *The operator*

$$\mathcal{C}_{\sigma,\alpha}^{box}\left([\mathbf{z}](1:m),[\mathbf{x}]\right) = \mathcal{C}_{\sigma,\alpha}^{interval}\left([z_1](1:m),[x_1]\right) \times \cdots \times \mathcal{C}_{\sigma,\alpha}^{interval}\left([z_n](1:m),[x_n]\right) \quad (5.11)$$

is a contractor for the set $\mathbb{X}_\alpha$ defined by (5.1), in the case where the $\mathbb{Z}_j$ are boxes $[\mathbf{z}](j)$.

Proof. We have to check that $\mathcal{C}_{\sigma,\alpha}^{\text{box}}\left([\mathbf{x}]\right)$ satisfies the properties of (2.31). Since all components $[x_i]$ of $[\mathbf{x}]$ are contracted by the contractor $\mathcal{C}_{\sigma,\alpha}^{\text{interval}}\left([z_i]\,(1:m),[x_i]\right)$, we get the contractance property. Let us show the consistency:

$$\begin{aligned}
\mathbb{X}_\alpha &= \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\zeta^1(\mathbf{x}),\ldots,\zeta^m(\mathbf{x})\right) \geq \alpha\right\} && \text{(see (5.1))}\\
&\subset \left\{\mathbf{x} \in [\mathbf{x}] \mid \forall i, \sigma\left(\zeta_i^1(x_i),\ldots,\zeta_i^m(x_i)\right) \geq \alpha\right\} && \text{(see (5.8))}\\
&= \bigcap_i \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\zeta_i^1(x_i),\ldots,\zeta_i^m(x_i)\right) \geq \alpha\right\}\\
&= \left\{x_1 \in [x_1] \mid \sigma\left(\zeta_1^1(x_1),\ldots,\zeta_1^m(x_1)\right) \geq \alpha\right\} \times \ldots\\
&\qquad \cdots \times \left\{x_n \in [x_n] \mid \sigma\left(\zeta_n^1(x_n),\ldots,\zeta_n^m(x_n)\right) \geq \alpha\right\}.
\end{aligned}$$

Finally, the monotonicity is a consequence of the fact that the Cartesian product is a monotonic operator. □

Algorithm 2: Contractor $\mathcal{C}^{\text{box}}_{\sigma,\alpha}$

Result: Contract the box $[\mathbf{x}]$ for the constraint $\sigma\left(\zeta^1(\mathbf{x}),\ldots,\zeta^m(\mathbf{x})\right)\geq\alpha$
Input: $[x]$, $\mathbf{z}(1:m)$
Output: $[x]$

1 **for** $i=0$ **to** n **do**
2 $\quad[x_i]=\mathcal{C}^{\text{interval}}_{\sigma,\alpha}([z_i](1:m),[x_i])$
3 **return** $[x_1]\times\cdots\times[x_n]$

Algorithm 2 thus implements a contractor for $\mathbb{X}_\alpha$.

Here are comments on each line of this algorithm:

Step 1. This loop allows a decomposition with respect to each axis of $\mathbf{x}$.

Step 2. We call the contractor $\mathcal{C}^{\text{interval}}_{\sigma,\alpha}$ that has been developed in Algorithm 1 for the scalar case.

Step 3. The Cartesian product of all contracted intervals is returned.

General case

Finally, now that we have a contractor for boxes, we can address the general case. We assume that the granules $\mathbb{Z}_j$ are any subset of $\mathbb{R}^n$ and that a contractor for each of these granules is available. The following theorem gives a definition for the searched contractor $\mathcal{C}^{\text{set}}_{\sigma,\alpha}$.

Theorem 5.2.4. *Assume that we have contractors $\mathcal{C}_{\mathbb{Z}_j}$, for the $\mathbb{Z}_j$. The operator*

$$\mathcal{C}^{set}_{\sigma,\alpha}\left(\mathbb{Z}_{1:m},[\mathbf{x}]\right)=\mathcal{C}^{box}_{\sigma,\alpha}\left(\left(\mathcal{C}_{\mathbb{Z}_1}([\mathbf{x}]),\ldots,\mathcal{C}_{\mathbb{Z}_m}([\mathbf{x}])\right),[\mathbf{x}]\right)\tag{5.12}$$

is a contractor for the set $\mathbb{X}_\alpha$ defined by (5.1), in the case where the $\mathbb{Z}_j$ are any subsets of $\mathbb{R}^n$.

Proof. We have to check that $\mathcal{C}^{\text{set}}_{\sigma,\alpha}\left([\mathbf{x}]\right)$ satisfies the properties of (2.31). For the contractance property, since the $\mathcal{C}_{\mathbb{Z}_j}$ has the contractance property, and $\mathcal{C}^{\text{box}}_{\sigma,\alpha}$ has the monotonicity property, we have:

$$\mathcal{C}^{\text{set}}_{\sigma,\alpha}\left(\mathbb{Z}_{1:m},[\mathbf{x}]\right)\subset\mathcal{C}^{\text{box}}_{\sigma,\alpha}\left(\left([\mathbf{x}],\ldots,[\mathbf{x}]\right),[\mathbf{x}]\right).\tag{5.13}$$

In addition,

$$\begin{aligned}\mathcal{C}^{\text{box}}_{\sigma,\alpha}\left(\left([\mathbf{x}],\ldots,[\mathbf{x}]\right),[\mathbf{x}]\right)&=\mathcal{C}^{\text{interval}}_{\sigma,\alpha}\left(\left([x_1],\ldots,[x_1]\right),[x_1]\right)\times\cdots\times\mathcal{C}^{\text{interval}}_{\sigma,\alpha}\left(\left([x_n],\ldots,[x_n]\right),[x_n]\right)\\&=[x_1]\times\cdots\times[x_n]\\&=[\mathbf{x}],\end{aligned}$$

hence

$$\mathcal{C}^{\text{set}}_{\sigma,\alpha}\left(\mathbb{Z}_{1:m},[\mathbf{x}]\right)\subset[\mathbf{x}].\tag{5.14}$$

Let us show the consistency:

$$\begin{aligned}
\mathbb{X}_\alpha &= \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha\right\} && \text{(see (5.1))} \\
&= \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\chi_{\mathbb{Z}_1}(\mathbf{x}), \ldots, \chi_{\mathbb{Z}_m}(\mathbf{x})\right) \geq \alpha\right\} \\
&= \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\chi_{\mathbb{Z}_1 \cap [\mathbf{x}]}(\mathbf{x}), \ldots, \chi_{\mathbb{Z}_m \cap [\mathbf{x}]}(\mathbf{x})\right) \geq \alpha\right\} \\
&\subset \left\{\mathbf{x} \in [\mathbf{x}] \mid \sigma\left(\chi_{\mathcal{C}_{\mathbb{Z}_1}([\mathbf{x}])}(\mathbf{x}), \ldots, \chi_{\mathcal{C}_{\mathbb{Z}_m}([\mathbf{x}])}(\mathbf{x})\right) \geq \alpha\right\} \\
&\subset \mathcal{C}^{\text{box}}_{\sigma,\alpha}\left((\mathcal{C}_{\mathbb{Z}_1}([\mathbf{x}]), \ldots, \mathcal{C}_{\mathbb{Z}_m}([\mathbf{x}])), [\mathbf{x}]\right).
\end{aligned}$$

Finally, the monotonicity property directly comes from the monotonicity of the contractors $\mathcal{C}^{\text{set}}_{\sigma,\alpha}$ and the $\mathcal{C}_{\mathbb{Z}_j}$. □

Algorithm 3 thus implements a contractor for $\mathbb{X}_\alpha$.

Algorithm 3: Contractor $\mathcal{C}^{\text{set}}_{\sigma,\alpha}$

Result: Contract the box $[\mathbf{x}]$ for the constraint $\sigma\left(\zeta^1(\mathbf{x}), \ldots, \zeta^m(\mathbf{x})\right) \geq \alpha$
Input: $[x]$, $\{\mathcal{C}_{\mathbb{Z}_1}, \ldots, \mathcal{C}_{\mathbb{Z}_m}\}$
Output: $[x]$

1 **for** $j = 1$ **to** m **do**
2 $\quad [\mathbf{z}](j) = \mathcal{C}_{\mathbb{Z}_j}([\mathbf{x}])$
3 $[\mathbf{x}] = \mathcal{C}^{\text{box}}_{\sigma,\alpha}([\mathbf{z}](1:m), [\mathbf{x}])$
4 **return** $[\mathbf{x}]$

Here are comments on each line of this algorithm:

Step 1. This loop allows to browse the contractors $\mathcal{C}_{\mathbb{Z}_j}$.

Step 2. We create a list of boxes corresponding to the granules $\mathbb{Z}_j$.

Step 3. We call the contractor $\mathcal{C}^{\text{box}}_{\sigma,\alpha}$ that has been developed in Algorithm 2 for the box case, on the created list of boxes.

Step 4. The contracted box is returned.

5.2.3 Characterization of α-cuts

Separator for α-cuts

Now that we have described all the tools required to obtain a characterization of α-cuts using interval analysis, we summarize the assumptions and the final approach to obtain a result as the Figure 2.14 of Example 2.4.3.

Consider we have some granules of knowledge represented by the sets $\mathbb{Z}_j$. We assume that we have available contractors $\mathcal{C}_{\mathbb{Z}_j}$ for the $\mathbb{Z}_j$, and also contractors $\mathcal{C}_{\bar{\mathbb{Z}}_j}$ for the complementary sets $\bar{\mathbb{Z}}_j$. In other words, we have all the tools to fully characterize the granules, with an inner and an

outer approximation, since a contractor and its complementary allow us to build the associated separator. We want to compute an inner and an outer approximation for $\mathbb{X}_\alpha$, the α-cut of the fuzzy set described by the granules $\mathbb{Z}_j$, and a score function σ distributing the confidence we have in each granule. We have seen that a contractor for $\mathbb{X}_\alpha$, denoted by $\mathcal{C}_{\mathbb{X}_\alpha}$, can be obtained using Algorithm 3:

$$\mathcal{C}_{\mathbb{X}_\alpha}([\mathbf{x}]) = \mathcal{C}^{\text{set}}_{\sigma,\alpha}\left(\mathcal{C}_{\mathbb{Z}_1}, \ldots, \mathcal{C}_{\mathbb{Z}_m}, [\mathbf{x}]\right). \tag{5.15}$$

Similarly, as we have access to the contractors of the complementary granules, we define a contractor for $\bar{\mathbb{X}}_\alpha$, denoted by $\mathcal{C}_{\bar{\mathbb{X}}_\alpha}$:

$$\mathcal{C}_{\bar{\mathbb{X}}_\alpha}([\mathbf{x}]) = \mathcal{C}^{\text{set}}_{\bar{\sigma},\bar{\alpha}}\left(\mathcal{C}_{\bar{\mathbb{Z}}_1}, \ldots, \mathcal{C}_{\bar{\mathbb{Z}}_m}, [\mathbf{x}]\right), \tag{5.16}$$

where (see (5.5) in Theorem 5.2.1)

$$\begin{aligned}\bar{\sigma}(a_1, \ldots, a_m) &= 1 - \sigma(1 - a_1, \ldots, 1 - a_m)\\ \bar{\alpha} &= 1 - \alpha.\end{aligned}$$

The pair $\mathcal{C}_{\mathbb{X}_\alpha}, \mathcal{C}_{\bar{\mathbb{X}}_\alpha}$ forms a separator for $\mathbb{X}_\alpha$. Using a set inversion algorithm as SIVIA, inner and outer approximations can be obtained. This algorithm can be applied for the different α-cuts of $\mathbb{X}$, and in this way, we can obtain a complete representation of the fuzzy set, as in Figure 2.14.

Localization example

To illustrate once again the approach, let us take another example of localization. Consider a robot in a room with a shape as in Figure 5.2. There is a pillar constituting an obstacle and a beacon $\mathbf{A}$ (in red) whith which the robot can measure the distance $d_{\mathbf{A}}$. Finally, the robot also measures its distance d_{wall} to the closest wall in front of him. Note that the sensor that produces the latter measurement should detect the pillar. However, in this purely theoretical example, we assume that it ignores the pillar for the sake of simplicity.

In this situation, we have different information that constitutes our granules of knowledge:

1. $\mathbb{Z}_{\text{room}}$, which is the set of all positions in the room;

2. $\mathbb{Z}_{\text{pillar}}$, which is the set of all positions in the pillar;

3. $\mathbb{Z}_{d_{\mathbf{A}}}$, which is the set of all positions compatible with the measured distance between the robot and the beacon, including uncertainties;

4. $\mathbb{Z}_{d_{\text{wall}}}$, which is the set of all positions compatible with the measured distance between the robot and the closest wall in front of the robot, including uncertainties.

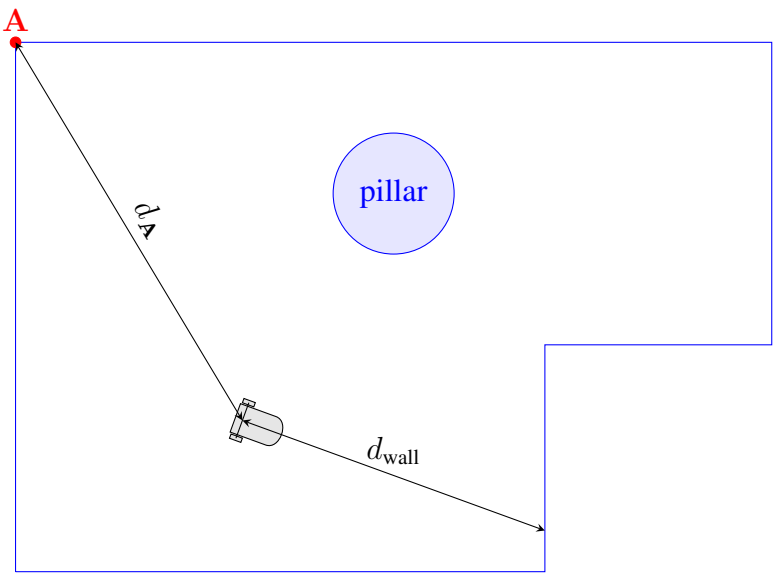


Figure 5.2: A robot is in a room.

Thus, we can define constraints on the position of the robot $\mathbf{x} \in \mathbb{R}^2$:

$$\begin{cases} \mathbf{x} \in \mathbb{Z}_{\text{room}} & \leftrightarrow & \text{The robot is in the room.} \\ \mathbf{x} \notin \mathbb{Z}_{\text{pillar}} & \leftrightarrow & \text{The robot is not in the pillar.} \\ \mathbf{x} \in \mathbb{Z}_{d_{\mathbf{A}}} & \leftrightarrow & \text{The robot is at a distance } d_{\mathbf{A}} \text{ to the beacon } \mathbf{A}. \\ \mathbf{x} \in \mathbb{Z}_{d_{\text{wall}}} & \leftrightarrow & \text{The robot is at a distance } d_{\text{wall}} \text{ to a wall.} \end{cases} \tag{5.17}$$

For each granules, we have a separator associated:

1. $\mathcal{S}_{\mathbb{Z}_{\text{room}}}$ is built using a polygon separator, which is a boundary-based separator, as described in [40, Chap. 2];
2. $\mathcal{S}_{\mathbb{Z}_{\text{pillar}}}$ is built using a forward-backward separator on the distance to the center of the pillar;
3. $\mathcal{S}_{\mathbb{Z}_{d_{\mathbf{A}}}}$ is built using a forward-backward separator on the distance to the beacon $\mathbf{A}$;
4. $\mathcal{S}_{\mathbb{Z}_{d_{\text{wall}}}}$ is built using the contractor $\mathcal{C}_{\text{dist\_wall}}$ described in Section 4.3 and its associated inner contractor $\mathcal{C}_{\overline{\text{dist\_wall}}}$ built in the same way with the opposite constraint.

Concerning the second separator, related to the pillar, it is associated with the set representing the pillar. However we want the complementary of this set: $\mathbf{x} \notin \mathbb{Z}_{\text{pillar}}$. In the separator algebra, the *complement* of a separator $\mathcal{S}$ is well defined and is denoted by $\bar{\mathcal{S}}$ [74]. Basically, it consists in switching the inner and the outer approximations to characterize the complementary set. So we will use the separator $\bar{\mathcal{S}}_{\mathbb{Z}_{\text{pillar}}}$ to characterize the second granule.

Remark. When we have access to a separator $\mathcal{S}$ that characterizes the complement of a desired granule $\bar{\mathbb{Z}}$, it is not possible to consider the complement of the associated characteristic function with our approach. Indeed, the function that returns $1 - \zeta^{\bar{\mathbb{Z}}}$ is well associated with the desired set

$\bar{\bar{\mathbb{Z}}} = \mathbb{Z}$, but it is not a score function as it is not equal to 0 in 0 and to 1 in 1. So the solution is to take the complement of the separator, $\bar{\mathcal{S}}$.

Now we have all the required assumptions to use the fuzzy approach. We want to construct the fuzzy sets $\mathbb{X}_\alpha$ for different values of α, representing the possible positions for the robot, according to the available granules. For each granule, we have its associated characteristic function: $\zeta^j(\mathbf{x}) = 1 \iff \mathbf{x} \in \mathbb{Z}_j$. We write $\zeta^{\overline{\text{pillar}}}(\mathbf{x})$ for the characteristic function associated with $\bar{\mathbb{Z}}_{\text{pillar}}$. We can now define the score function σ to combine and prioritize the granules, for instance:

$$\sigma\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\text{pillar}}(\mathbf{x}), \zeta^{d_{\mathbf{A}}}(\mathbf{x}), \zeta^{d_{\text{wall}}}(\mathbf{x})\right) = \min\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\overline{\text{pillar}}}(\mathbf{x}), \frac{\zeta^{d_{\mathbf{A}}}(\mathbf{x}) + \zeta^{d_{\text{wall}}}(\mathbf{x})}{2}\right), \tag{5.18}$$

which gives us the following correspondences:

$$\mathbb{X}_1 = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \mathbb{Z}_{d_{\mathbf{A}}} \cap \mathbb{Z}_{d_{\text{wall}}} \tag{5.19}$$

$$\mathbb{X}_{0.5} = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \left(\mathbb{Z}_{d_{\mathbf{A}}} \cup \mathbb{Z}_{d_{\text{wall}}}\right). \tag{5.20}$$

The function σ is a score function as it satisfies the three conditions defined in (2.47) (the $\min$ function is monotonic). The four granules are represented in Figure 5.3. The resulting fuzzy set defined with this score function is represented in Figure 5.4, where the orange area corresponds to the 0.5-cut and the red one to $\mathbb{X}_1$. A 3D view is also given. Naturally, each area painted in red is also painted in orange, as $\mathbb{X}_1 \subset \mathbb{X}_{0.5}$.

Assume now that the sensor measuring the distance to walls can rotate to acquire m measurements from all around it. Thus, we no longer have a unique granule $\mathbb{Z}_{d_{\text{wall}}}$, but a family of m granules $\{\mathbb{Z}_{\theta_1,d_{\text{wall}}}, \ldots, \mathbb{Z}_{\theta_m,d_{\text{wall}}}\}$. It makes us define a new score function involving the new associated characteristic function $\zeta^{\theta_k,d_{\text{wall}}}$ for k in $1, \ldots, m$. We know that among these measurements, we will encounter outliers, so we can use a relaxed intersection for these data:

$$\begin{aligned}\sigma\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\text{pillar}}(\mathbf{x}), \zeta^{d_{\mathbf{A}}}(\mathbf{x}), \left\{\zeta^{\theta_k,d_{\text{wall}}}(\mathbf{x})\right\}_{k\in[1,\ldots,m]}\right)& \\ = \min\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\overline{\text{pillar}}}(\mathbf{x}), \frac{1}{2}\left(\zeta^{d_{\mathbf{A}}}(\mathbf{x}) + \frac{1}{m}\sum_{k=1}^{m}\zeta^{\theta_k,d_{\text{wall}}}(\mathbf{x})\right)\right).&\end{aligned} \tag{5.21}$$

Once again, we check that the function σ is a score function as it satisfies the three conditions defined in (2.47). Here we have the following correspondences, where q is the number of expected outliers among m measurements:

$$\mathbb{X}_1 = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \mathbb{Z}_{d_{\mathbf{A}}} \cap \bigcap_{k=1}^{m} \mathbb{Z}_{\theta_k,d_{\text{wall}}} \tag{5.22}$$

$$\mathbb{X}_{1-\frac{q}{2m}} = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \mathbb{Z}_{d_{\mathbf{A}}} \cap \bigcap_{k=1}^{m}{}^{\{q\}} \mathbb{Z}_{\theta_k,d_{\text{wall}}} \tag{5.23}$$

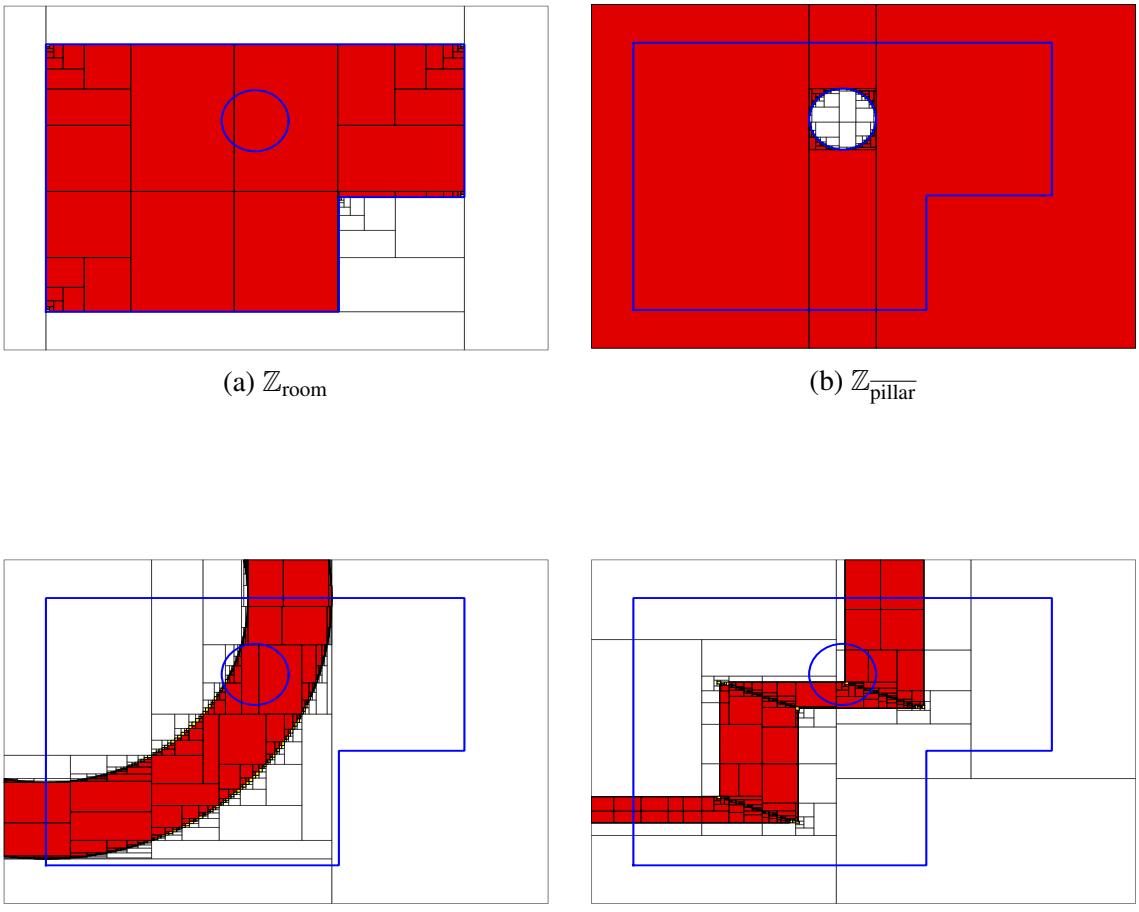
$$\mathbb{X}_{0.5} = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \left(\mathbb{Z}_{d_{\mathbf{A}}} \cup \mathbb{Z}_{d_{\text{wall}}}\right) \tag{5.24}$$

$$\mathbb{X}_{0.5-\frac{q}{2m}} = \mathbb{Z}_{\text{room}} \cap \bar{\mathbb{Z}}_{\text{pillar}} \cap \left(\mathbb{Z}_{d_{\mathbf{A}}} \cup \bigcap_{k=1}^{m}{}^{\{q\}} \mathbb{Z}_{\theta_k,d_{\text{wall}}}\right). \tag{5.25}$$

To go further, this score function can be finely adjusted depending on the confidence of each granule. For instance, if the reliability of the sensor that returns the distance to the walls depends on θ_k (it is more reliable when the wall is perpendicular to the direction of the acquisition), it is possible to implement a special relaxed intersection that takes into account this new information:

$$\sigma\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\text{pillar}}(\mathbf{x}), \zeta^{d_{\mathbf{A}}}(\mathbf{x}), \left\{\zeta^{\theta_k, d_{\text{wall}}}(\mathbf{x})\right\}_{k\in[1,\dots,m]}\right) = \min\left(\zeta^{\text{room}}(\mathbf{x}), \zeta^{\overline{\text{pillar}}}(\mathbf{x}), \frac{1}{2}\left(\zeta^{d_{\mathbf{A}}}(\mathbf{x}) + \frac{1}{m}\sum_{k=1}^{m}\varsigma(\theta_k)\zeta^{\theta_k, d_{\text{wall}}}(\mathbf{x})\right)\right), \quad (5.26)$$

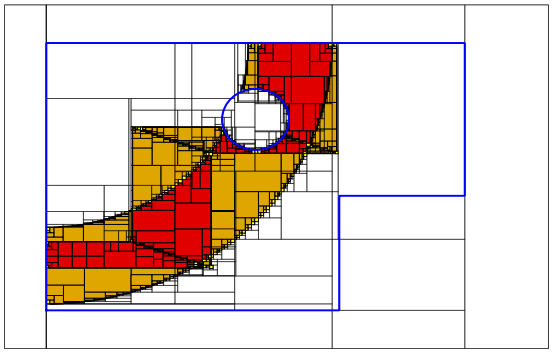
where $\varsigma(\theta_k)$ returns a scalar between 0 and 1 such that we have $\sum_{k=1}^{m}\varsigma(\theta_k) = m$ in order to ensure that the σ function remains a score function. Thus, ς allows to adjust the influence of each granule depending on the direction it is associated with.



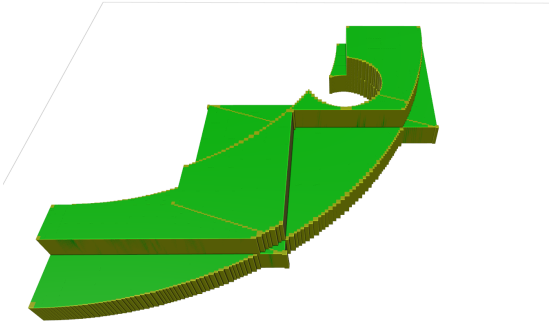
(a) $\mathbb{Z}_{\text{room}}$ (b) $\mathbb{Z}_{\overline{\text{pillar}}}$

(c) $\mathbb{Z}_{d_{\mathbf{A}}}$ (d) $\mathbb{Z}_{d_{\text{wall}}}$

Figure 5.3: Representation of the granules $\mathbb{Z}_1$ to $\mathbb{Z}_4$.



(a) Top view of the fuzzy set. Orange area corresponds to the 0.5-cut, and the red one to the 1-cut (the core of the fuzzy set).



(b) 3D-view of the fuzzy set.

Figure 5.4: Representation of the fuzzy set $\mathbb{X}$.

Dealing with fuzzy granules

Until now, the granules were always supposed to be crisp sets. They represented partial knowledge, and fuzzy logic was used to combine them into a fuzzy set that gave the different possibilities for the searched set. However, one can desire to deal with fuzzy granules. Indeed, these granules are usually defined either with the sensor data or some constraints. The former can justify fuzzy granules: perhaps a returned measurement would be better represented by a gaussian bell-shaped curve or by a trapezium-shaped curve, for instance. The latter barely requires a fuzzy representation; as an example, the constraint "the robot is in the room" is not fuzzy at all (except if the positions of the room's walls are uncertain). We could define *fuzzy constraints*, which are constraints that can be violated sometimes or that are defined on imprecise information.

For instance, we take once again the previous example with the robot in the room, but we define the granule $\mathbb{Z}_{d_{\mathbf{A}}}$ with a fuzzy set such that the distance to the beacon $\mathbf{A}$ is represented by a trapezium-shaped membership function, as in Figure 5.5.

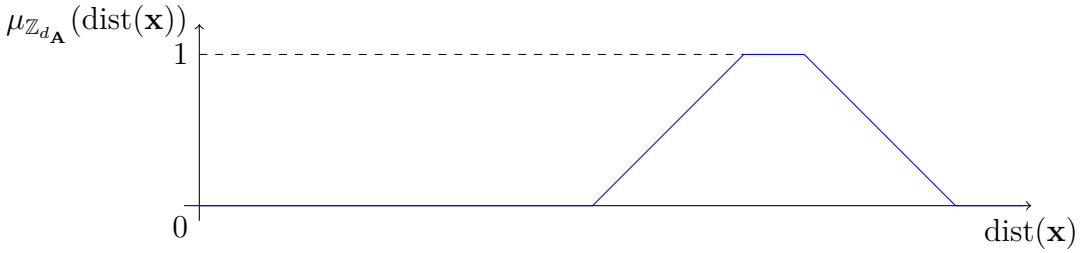


Figure 5.5: Trapezium-shaped membership function for the measured distance to the beacon $\mathbf{A}$ with respect to the distance to a position $\mathbf{x}$.

Now if we choose the same score function (5.18) as in Figures 5.4, we obtain Figure 5.6. Note that the value of α is discretized for the representation of the fuzzy set, as the method used

cannot compute the approximations of an infinite number of α-cuts. This is why the trapezium shape has some pixelation (stairs shape). Regardless, the result is still wrapping the fuzzy set as the discretization of the α keeps the guarantee that no solution is removed. It is possible because of the nested property of the α-cuts.

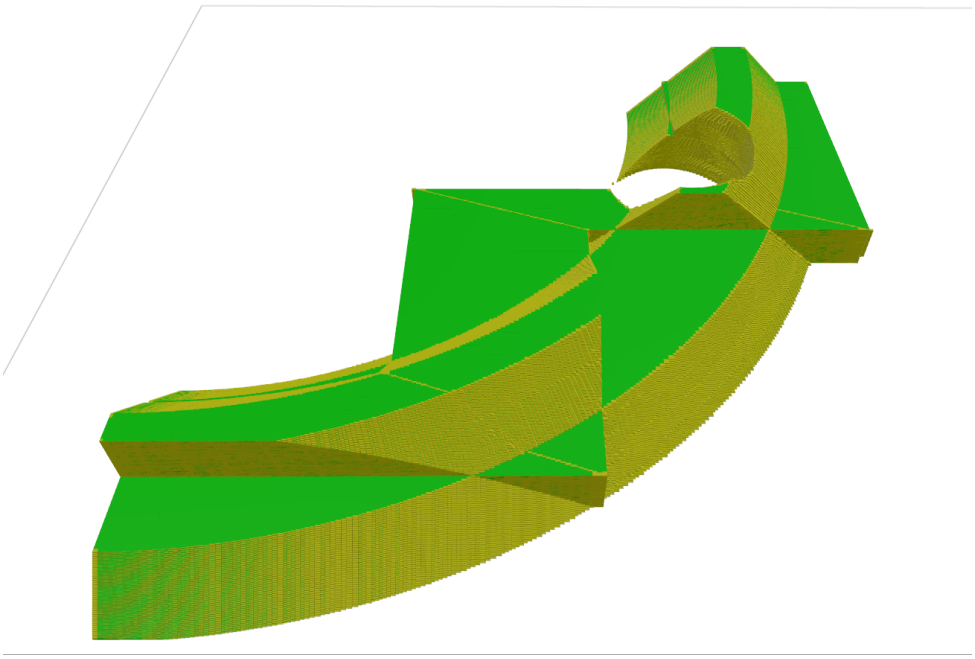


Figure 5.6: Representation of the fuzzy set $\mathbb{X}$ when the granule for the distance to the beacon **A** is a fuzzy set.

Concretely, to compute approximations of α-cuts of a fuzzy set $\mathbb{X}$ defined by the composition of granules when these granules can be fuzzy sets, we need the contractors for the α-cuts of the fuzzy granules. So, Algorithm 3 can be used to compute these α-cuts, and then the Algorithm 2 can approximate the resulting α-cut of the researched fuzzy set. Finally, the researched fuzzy set can be characterized by its different α-cuts, discretizing the possible values for α when membership functions of the fuzzy granules are continuous.

5.3 Application to localization with scanning sonar

In Chapter 4, a first approach to the localization of an underwater robot using a scanning sonar has been presented. To obtain a better precision in localization, we need to take a little more risk in the choice of constraints. Indeed, some constraints seemed relevant, so we would like to use them even if sometimes it is misleading. The fuzzy approach presented in this chapter brings the perfect tool for this kind of situation. From the sonar data, we obtain multiple information, which would be our granules. We cannot trust the same way each one of these granules. The fuzzy method allows giving more or less weight, so some risk can be taken by associating a lower weight to the hazardous granules.

For instance, a peak in sonar data does not always correspond to a wall. However, we can be closer to reality by adjusting the weights we give to each peak. We know that the first peak after the cross-talk one more likely corresponds to an obstacle. In addition, perhaps it would be relevant to give more weight to higher and more prominent peaks.

This section shows the suitability of this approach on the localization of a ROV in a pool.

5.3.1 Formalism and simulation

Formalism

Consider an underwater robot moving in a pool equipped with a directional sonar, a compass and a pressure sensor. We assume that the robot is static and that the pool is made with $\bar{w}$ vertical walls $\mathbb{W}(w)$, $w \in \{1, \ldots, \bar{w}\}$. Since the pressure sensor provides the depth and the compass returns the heading, the localization problem amounts to finding a point $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$ inside a horizontal plane from the distances returned by the sonar. The directional sonar rotates and emits $\bar{k}$ ultrasonic sounds (or *pings*) toward different directions $\theta_1, \ldots, \theta_{\bar{k}}$. For the kth ping, several echoes are returned. Each of them can be represented by a distance interval $[d_{k,\ell}]$. Only one is significant for us: the echo ℓ which corresponds to the nearest wall $\mathbb{W}(w)$ reached by the sound emitted toward to the direction θ_k. The 3-uple (k, w, ℓ) is then an inlier. For non significant echoes, (k, w, ℓ) is an outlier.

Figure 5.7 shows a typical echo signal that could have been collected just after the kth ping. The first echo $[d_{k,1}]$ may correspond to an echo from the surface of the water or any unmapped object. The second echo is the inlier (it corresponds to the echo returned by the nearest wall). The corresponding interval $[d_{k,2}]$ contains the true distance d_k. The last echo $[d_{k,3}]$ may correspond to a multipath or to a noise emitted by another robot. The first and the third echoes correspond to outliers.

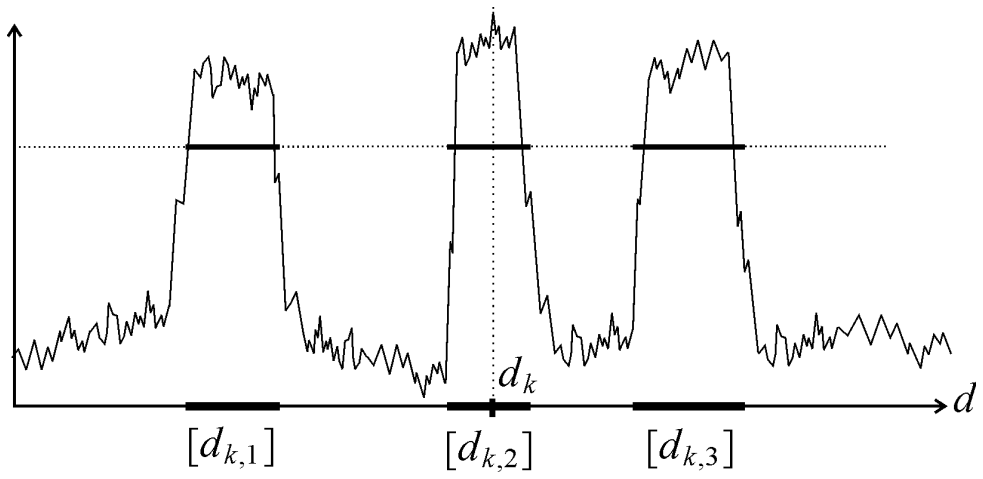


Figure 5.7: The signal collected by the sonar just after the ping emission.

Define the set

$$\begin{aligned}\mathbb{Z}_{k,w,\ell} = \{\mathbf{x} \in \mathbb{R}^2 \mid\ & \exists d \in [d_{k,\ell}], \\ & \exists w \in \{1,\ldots,\bar{w}\}, \\ & \exists \mathbf{m} = (m_1, m_2) \in \mathbb{W}(w), \\ & m_1 = x_1 + d\cos\theta_k, \\ & m_2 = x_2 + d\sin\theta_k \qquad \},\end{aligned} \tag{5.27}$$

as illustrated by Figure 5.8. We also define the set $\mathbb{Z}_0$, corresponding to all $\mathbf{x}$ that are in the pool. The set $\mathbb{Z}_{k,w,\ell}$ should contain the position $\mathbf{x}$ of the robot if (k, w, ℓ) corresponds to an inlier.

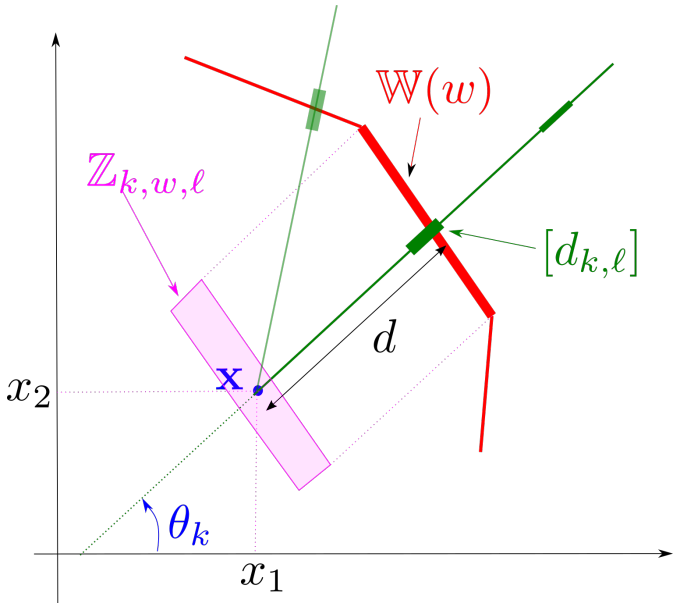


Figure 5.8: $\mathbb{Z}_{k,w,\ell}$ is the set of all positions for the robot consistent with the fact that the ℓth echo of the kth ping corresponds to the wth wall $\mathbb{W}(w)$.

An optimal contractor can easily be built for $\mathbb{Z}_{k,w,\ell}$ [38]. We introduce the fuzzy set $\mathbb{X}$ represented by the membership function

$$\mu_{\mathbb{X}}(\mathbf{x}) = \frac{1}{\bar{k}} \sum_{k \in \{1,\ldots,\bar{k}\}} \max_{\ell \in \{1,\ldots,\bar{\ell}(k)\}} \max_{w \in \{1,\ldots,\bar{w}\}} \min\left(\zeta^{k,w,\ell}(\mathbf{x}), \zeta^0(\mathbf{x})\right), \tag{5.28}$$

where $\zeta^{k,w,\ell}$ and ζ^0 are the characteristic functions for $\mathbb{Z}_{k,w,\ell}$ and $\mathbb{Z}_0$, respectively. Note that we have the following equivalence

$$\begin{aligned}&\max_{\ell \in \{1,\ldots,\bar{\ell}(k)\}} \max_{w \in \{1,\ldots,\bar{w}\}} \min\left(\zeta^{k,w,\ell}(\mathbf{x}), \zeta^0(\mathbf{x})\right) = 1 \\ &\qquad \iff \exists \ell \in \left\{1,\ldots,\bar{\ell}\,(k)\right\}, \exists w \in \{1,\ldots,\bar{w}\}, \mathbf{x} \in \mathbb{Z}_{k,w,\ell} \cap \mathbb{Z}_0,\end{aligned} \tag{5.29}$$

i.e., there exists (ℓ, w) such that the ℓ^{th} echo and the wth wall $\mathbb{W}(w)$ are consistent with the position $\mathbf{x}$ of the robot and the fact that $\mathbf{x}$ is in the pool. Note also that $\mu_{\mathbb{X}}(\mathbf{x}) \geq \alpha$ when at least $\bar{k} \cdot \alpha$ pings are consistent.

Simulation

In order to illustrate the approach, we now propose an example of a robot simulated in a pool as in Figure 5.9.

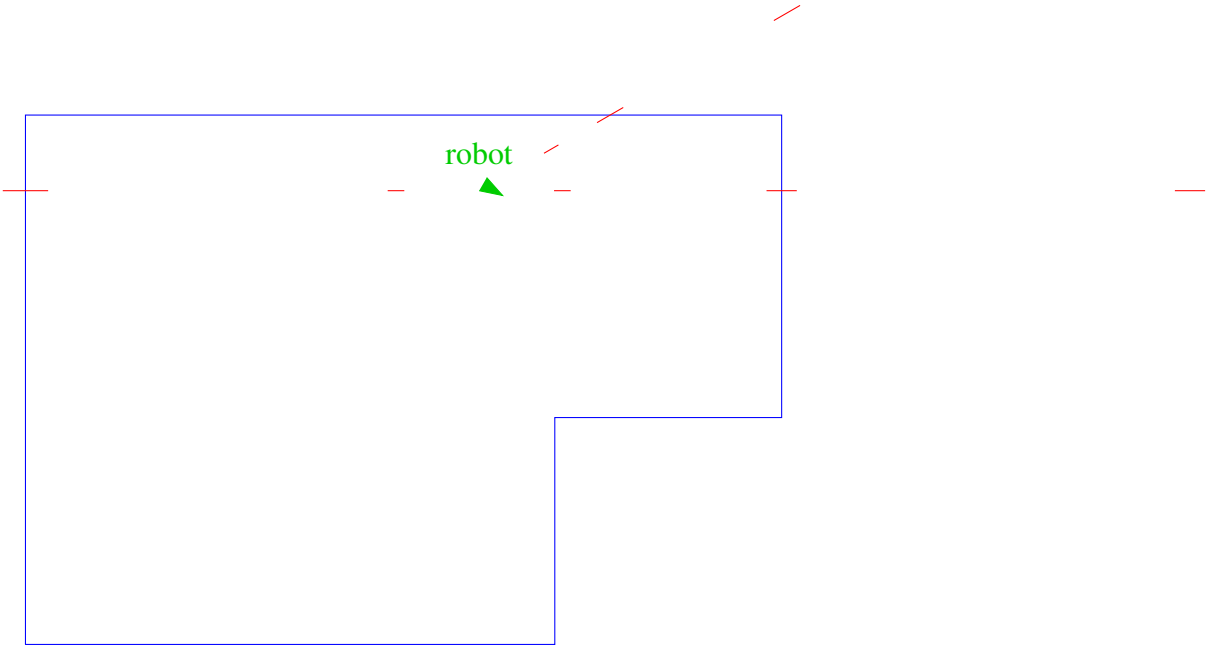


Figure 5.9: Simulation of a robot (in green) in a pool (in blue). Red lines correspond to simulated echoes received by the robot.

In this example, there are three pings with one inlier per ping. The first echo of each ping is an outlier: it comes from the surface of the water. The strategy consisting in using the relaxed intersection reaches its limits here. Indeed, there exist other possible localizations for the robot that give more inliers, as in Figure 5.10. By using more shrewdly the relaxed intersection, it is possible to define the limit of one inlier per ping and avoid misinterpretation. In this way, the two possible localizations are equivalent from the point of view of the algorithm, and it is not possible to go further with this strategy.

Now, using the fuzzy approach, we can give scores to each echo. For this application, we may want to classify the echoes by their distances: the first echo (the closer to the sonar) often corresponds to the surface but can also be a wall, the second one is probably a wall, and the other ones are less often inliers. By implementing these scores (respectively 0.8, 0.9, 0.5 in this example), the ambiguity is lifted, and the robot is well localized. Note that scores can be defined as finely as wanted. For instance, it is possible to take into account the shape of each peak, knowing that a flat wall will give a more explicit echo than the surface.

Figure 5.11 presents the result of the localization with the relaxed intersection. The solution found in the first case is misleading as several echoes of one ping are considered as inliers (5.11a). This issue is solved in the second case, and the localization of the robot is correct but imprecise (5.11b).

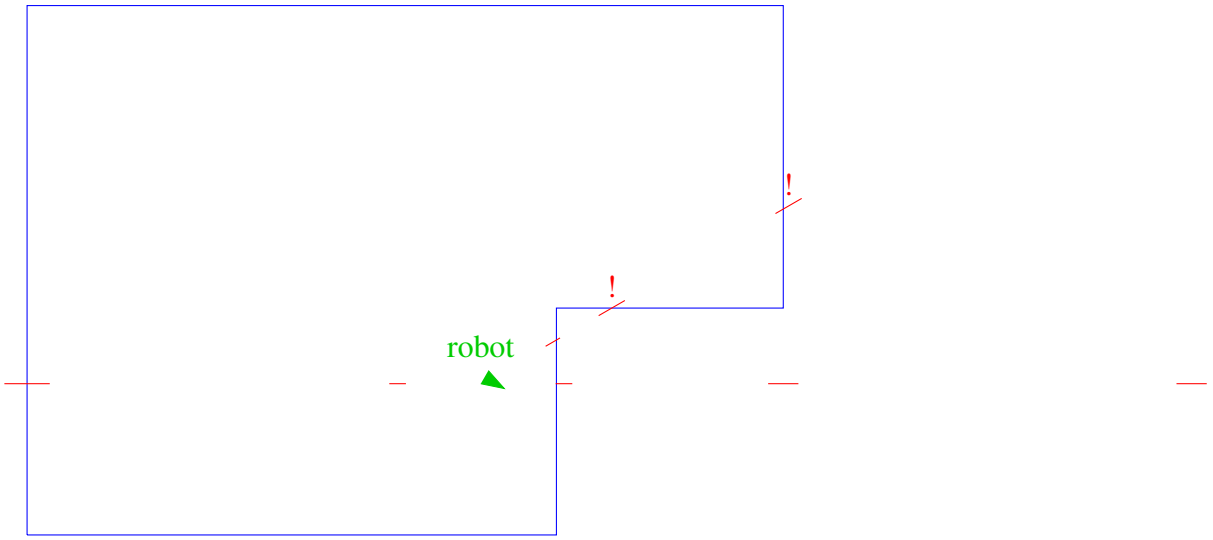


Figure 5.10: Another possible localization for the robot, where two outliers happen on the walls (represented by !).

Using a contractor associated with the μ function, we are able to go further for localizing the robot. Each echo receives a score depending on its position in the ping. By doing this weighting, it is possible to obtain a more precise localization as presented in Figure 5.12.

By increasing the α value, we have better precision, whereas by decreasing this score value, we become more robust to outliers, increasing the reliability of the result.

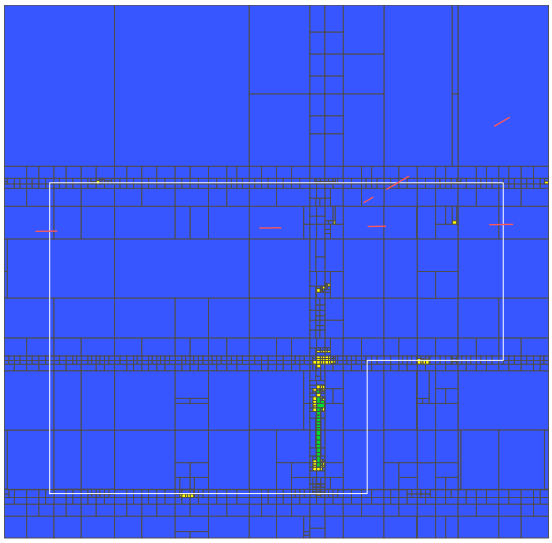
Presented figures have the following color coding: the blue is the outside of the researched set, and it corresponds to where the robot cannot be; the green is the interior of the set, where the robot can be; and finally, the yellow is the frontier, and the computation has been stopped before determining the membership of this too tiny area. Note that sometimes, as in Figure 5.12a, a green area is adjacent to a blue one, without any yellow frontier. This may happen when minimal contractors are used.

So the choice of the α value should be made wisely. Moreover, having the result for different values of α can be necessary sometimes. It may happen when we want both a reliable and precise result. That is why the computation of a 3D representation with α on the z-axis, as in Figure 5.13, can have some interest.

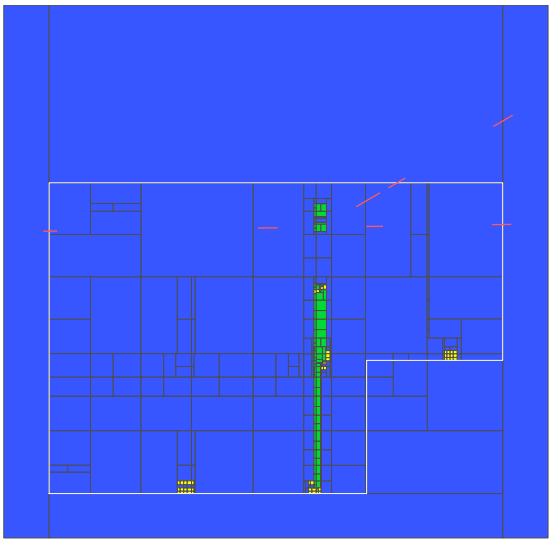
5.3.2 Experimentation

Our approach has been tested with a real robot in a pool. The sonar is still a Tritech Micron (mechanical scanning sonar) fixed on a BlueROV2 of BlueRobotics. The parameters have not been changed: the vertical beamwidth is 35° and the range 6 meters. We still have access to the heading returned by the IMU. Nevertheless, this time, the pool is rigid and has the shape of a rectangle of 3 x 4 meters. A picture of the experimentation is shown in Figure 5.14.

A picture of data received from the sonar is presented in Figure 5.15. A peak detection algorithm is used to generate the interval data. The score function has been adapted to give



(a) A naive relaxed intersection converges to a unique solution whereas this solution is not more relevant than other ones.



(b) Still using relaxed intersection, it is possible to specify that there is always a unique inlier per ping. Although correct, the result obtained can not distinguish the different possibilities of localization.

Figure 5.11: Example of localization with a relaxed intersection. The pool is drawn in white. The red lines correspond to the echoes given to the algorithm.

more importance to the first echo and less to the second one. In addition, the score is drastically reduced for echoes that are not directed perpendicularly to the walls. Indeed, it has been observed that such echoes are often less relevant (fewer chances to have an echo corresponding to a wall, as presented in Section 4.2.2).

We propose to take as a set-membership estimator the $\hat{\alpha}$-cut $\mathbb{X}_{\hat{\alpha}}$ of the fuzzy set $\mathbb{X}$ where

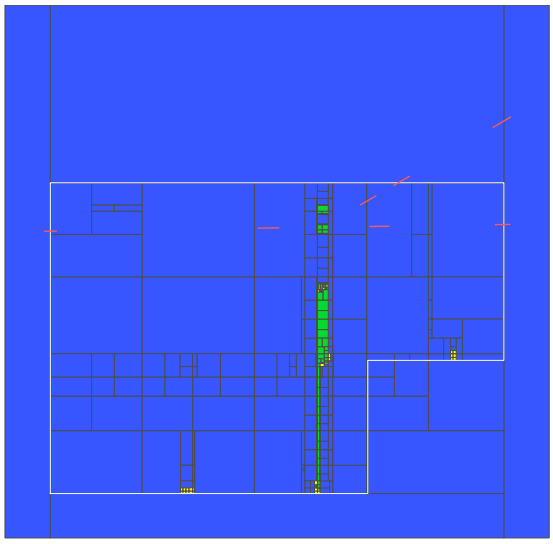
$$\hat{\alpha} = \max \{\alpha \mid \mathbb{X}_\alpha \neq \emptyset\} . \tag{5.30}$$

The corresponding estimator can be interpreted as a generalization of the Outlier Minimal Number Estimator (OMNE) presented in [91]. In OMNE, the score function σ is a simple sum.

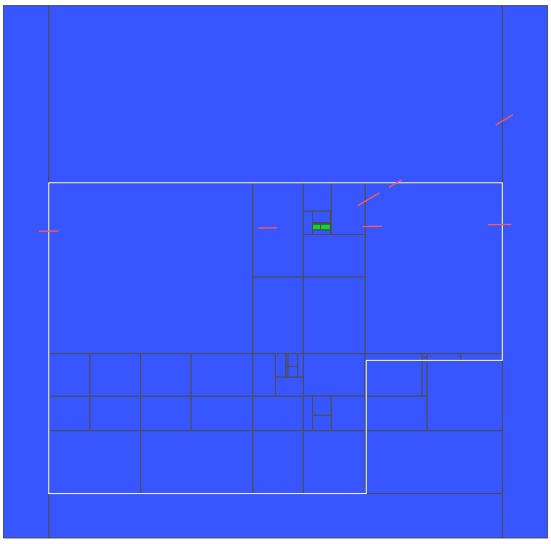
Our estimator generates a set $\mathbb{X}_{\hat{\alpha}}$ which contains the true position of the robot in the pool. As for most acoustic localization systems in a pool, many outliers exist. This is why we observe a small maximal score: $\hat{\alpha} = 0.15$. Figure 5.16 depicts the set $\mathbb{X}$. When the robot is moving, the procedure is called each time the sonar makes a turn. The localization takes less than 0.1 seconds on a standard computer and is therefore suitable for real-time application.

5.4 Conclusion

In this chapter, a fuzzy approach has been presented in order to propose a different way to represent uncertainties and deal with estimation problems. The proposed formulation is close to the priority approach to soft constraints presented in [21]. The main idea is that when we cannot



(a) Using the fuzzy approach, and by choosing $\alpha = 0.3$, the result is the same as the one obtained with the relaxed intersection.



(b) However, the fuzzy approach allows to give scores to each echo, and therefore going further in the localization. Here $\alpha = 0.4$ and the robot is localized without any ambiguity.

Figure 5.12: The same localization example with different α with the fuzzy approach. Using scores on the echoes, it is possible to get a thinner localization.

satisfy all the constraints, we should at least satisfy as many as possible, and a natural approach is to prioritize these constraints, from the absolutely required to the less required.

The principle is to combine some granules of knowledge into a fuzzy set, thus prioritizing the constraints. From the absolutely required constraint to the less important, they are all represented in the resulting fuzzy set. The score function enables defining the influence of each of these constraints. The different α-cuts allow choosing the number of constraints that should be satisfied.

The combination of these granules, which is described by the score function, allows a complex composition without combinatorial explosion. This approach is more general than the relaxed intersection, as some weights can be used depending on the confidence we have in each granule. The α-cuts characterizing the researched fuzzy set are efficiently computed using interval analysis.

The granules, coming from sensors or constraints, are crisp sets representing a partial knowledge of a given researched value. This partial knowledge can contain unknown uncertainties, and if needed, it is possible to deal with fuzzy granules to represent better the partial knowledge we want to consider.

Finally, the interests of this approach can be summarized by the following points.

- The formulation allows to combine contractors, in a more general way than what has been proposed in [155, 19, 22] since it allows the different weights in the constraints.

- Using the formulation based on the set-membership function, instead of the combination of contractors, we obtained an algorithm that is more efficient, requires fewer bisections,

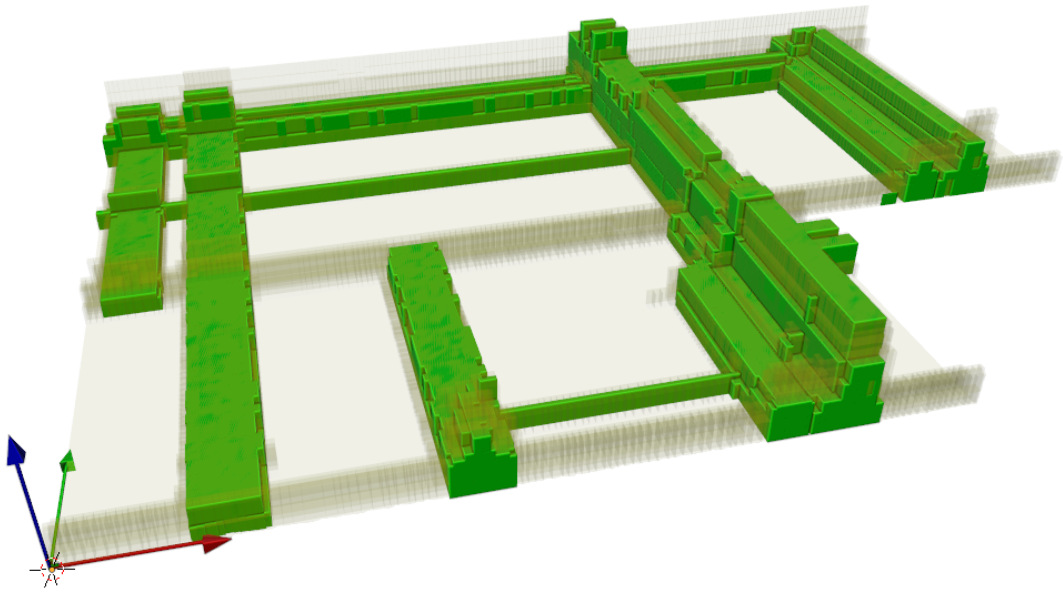


Figure 5.13: 3D view of the result with α on the z-axis. The color code is similar as previously: the interior is green, the frontier is in very transparent yellow, and the outside is not represented. The highest point corresponds to the position with the highest score, in other words, the position for the robot that fits the most the assumptions made on the sonar data.

and is simpler to implement.

- We proposed an estimator which is more general than OMNE proposed in [91] since it allows us to deal with more complex situations where outlier occurrences are interdependent.

The method has been validated on a real underwater robot for its localization using sonar data. Acoustic sensors, used in underwater environments, provide precious but hidden information. By exploiting them, it is possible to localize a robot without cumulating errors in the estimated position. Nevertheless, the data should be interpreted with particular care to avoid being misled.

The fuzzy approach we propose for the resolution of localization problems is reliable even if we have partial knowledge of the uncertainties. In this context, it is more adapted than the classical Bayesian approach due to the difficulty we have in extracting important data.

We have shown good robustness to outliers and localization with a high degree of integrity. The fuzzy approach brought us the possibility to fine-tune the influence of each sonar data with respect to the degree of confidence we have in the measurements.

The presented approach can be compared to some works made with belief functions, in evidence theory [141]. For instance, in [116], the author also deals with vehicle localization but uses belief functions to handle uncertainties. This other method allows representing both what is known and what remains unknown. Then the fusion of information is given by Dempster's rule [36], using *basic belief assignements*, or masses. Thus, it is possible to manage the confidence

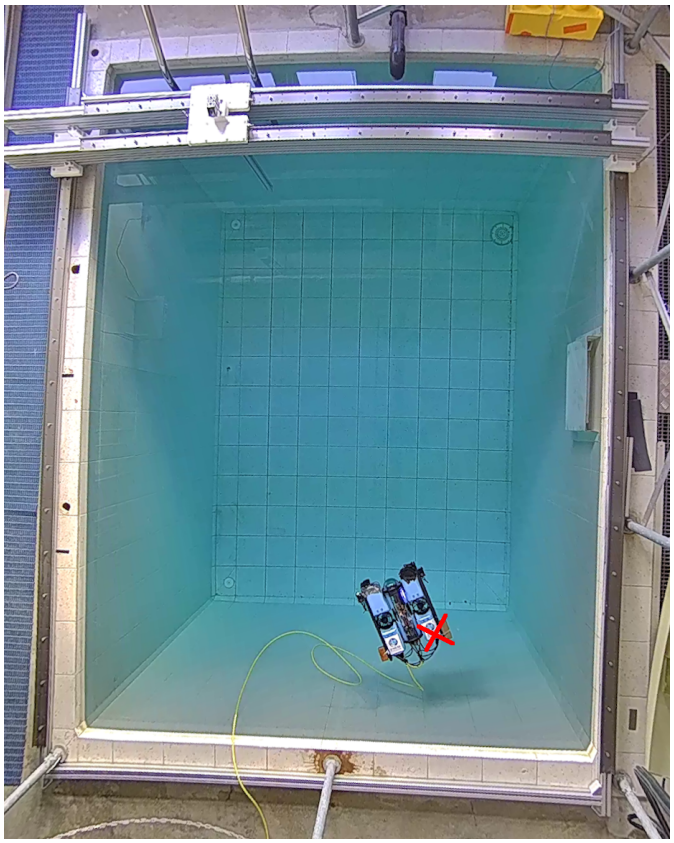


Figure 5.14: Picture of the BlueRobotics ROV in the pool localizing itself. The robot floats on the surface of the water, and the sonar is placed under the robot at the red cross position.

put in each source of information, as in [25]. The main difference relies on the combination of granules, which can be computed more efficiently using the score function. Indeed, the combination of belief functions requires considering the dependence between the data sources, leading to more complex algorithms to compute the basic belief assignments of the studied information.

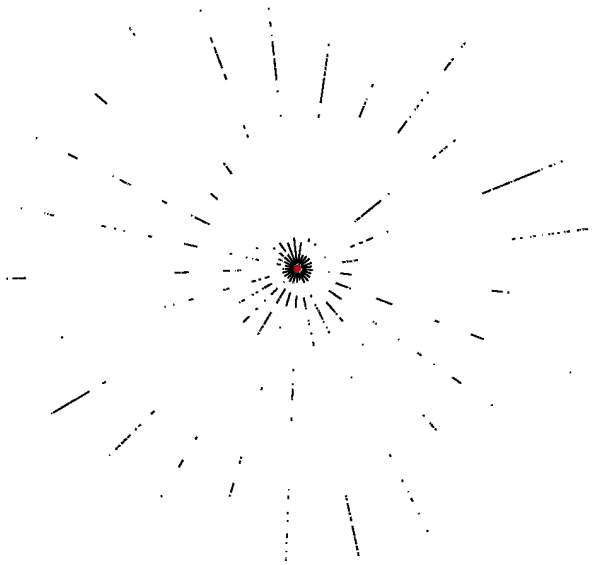
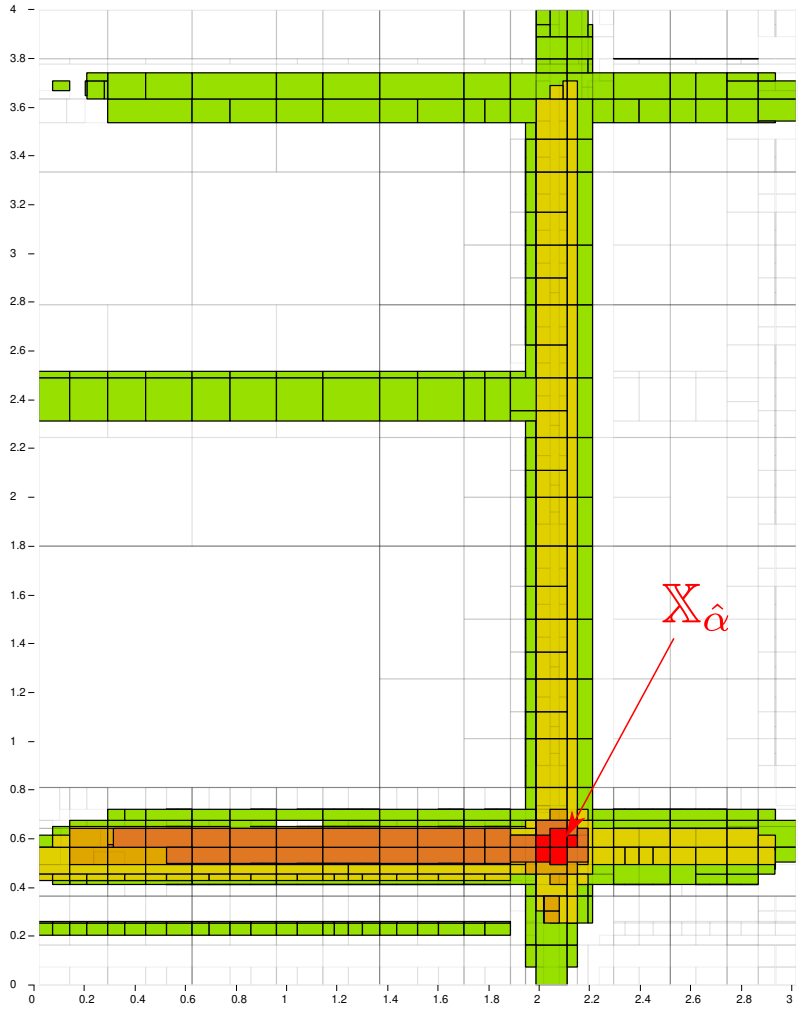
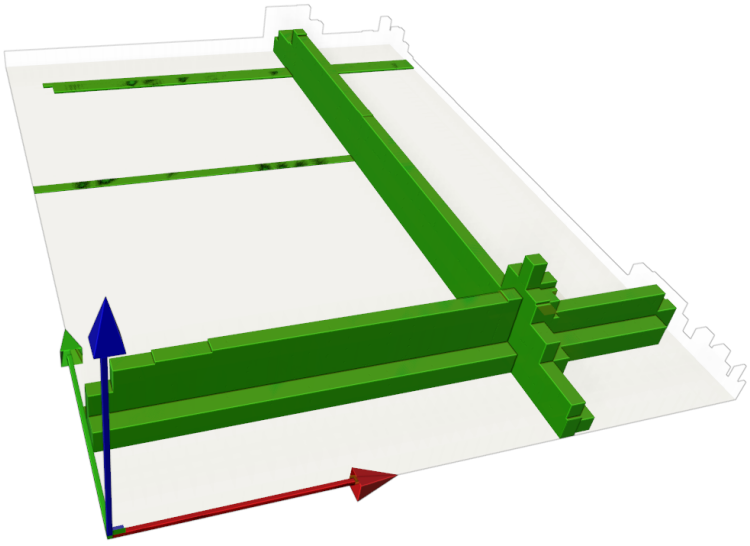


Figure 5.15: Representation of the data collected from the sonar. The red dot corresponds to the position of the robot.



(a) Top view with α going from green to red



(b) 3D view

Figure 5.16: Representation of the fuzzy set $\mathbb{X}$. (a): On the x_1, x_2-plane. The plateau corresponding to $\mathbb{X}_{\hat{\alpha}}$ is painted red. It contains the actual position of the robot. (b): 3D view of $\mathbb{X}$ in the (x_1, x_2, α)-space.

CHAPTER 6

Conclusion

Contents

6.1 Context

The two challenges presented in the introduction of this document were about the safe control and the localization of a deported sensor. The leading light application was the search of wrecks in underwater environments with a magnetometer. Searching wrecks is a vast subject with many similarities to sea exploration. It concerns various fields as history, oceanography, hydrography, acoustic, magnetism, to name a few.

We chose to study solutions for the two mentioned challenges in the scope of proposing an efficient robotic system for the search of wrecks. Robotics can bring undeniable advantages for such missions, like reducing human life hazards and human errors or increasing working time.

Nevertheless, underwater environments are challenging places for robots, and developing a robotic system to find the searched wreck is far from being straightforward. The solution that has been chosen consists of a deported magnetometer towed close to the seabed. Then, the two challenges have been studied independently in specific contexts to facilitate the approach and focus on the main difficulties. Thus, they consist in:

1. Finding a controller that makes the magnetometer follow the desired trajectory and validate this controller with respect to state constraints. This first challenge raised control issues that have been handled in Chapter 3. The problem is reduced to a 2D problem, with an application to a car towing a trailer modeling the behavior of a deported sensor. The safety issue is dealt with interval analysis by computing a *follow set* that contains every position where the wanted constraints are not violated.

2. Proposing a method to localize the robot reliably without communicating with external systems. This second challenge is complex: the environment is scanned using acoustic sensors, making us dealing with uncertain data and managing abundant outliers. Solutions have been studied firstly in Chapter 4 using tools from interval analysis. A particular effort is made to understand the basics of acoustics and find out which reliable information can be used. Then Chapter 5 has been gone further by using the fuzzy approach to manipulate more complex representations of the acquired granules of knowledge. Thus, soft constraints can be considered. By defining a score function, the constraints are prioritized to satisfy the more as possible and the more critical first.

In order to fulfill these two objectives, some required theoretical tools have been presented in Chapter 2 with, mainly, non-linear control, interval analysis, and fuzzy logic. Finally, the proposed approaches have been tested in simulation and on real robots to show the consistency of the methods.

6.2 Contributions

The main contributions in this thesis are presented in the following.

The design of a suitable controller to answer the first challenge has been studied, mainly using a feedback linearization approach. Theorem 3.3.1 describes the found modelization. An approach based on interval analysis to characterize *follow sets* has also been proposed and used to validate the designed controller. It has been the subject of a publication in a robotic conference:

> Joris Tillet, Luc Jaulin, and Fabrice Le Bars. "Non-linear control under state constraints with validated trajectories for a mobile robot towing a trailer". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020.

The presented contributions should help to address the problem of searching wrecks with a magnetometer. More generally, this work may be used to control any towed vehicle and to validate any controller with respect to state constraints.

Then the localization problem has been studied. The fuzzy logic has been used to develop a new fuzzy set estimation. Fuzzy sets are built with granules of knowledge and a score function to combine them. The approach consisting in using interval analysis to characterize α-cuts of a fuzzy set has been submitted in the *IEEE Transactions on Fuzzy Systems* journal. The application to underwater localization is proposed at the end of the paper.

The fuzzy approach seems particularly adapted to deal with underwater data. Nevertheless, it may lead to different applications and can be applied as soon as constraints should be prioritized or when incoming data should have a complex representation of their uncertainties.

6.3 Prospects

Naturally, much work can be envisioned to go into these approaches more deeply. There are several trails to improve them and find new applications.

Validation of a controller. The proposed method to characterize the follow set can be used to compute every safe point in the output space. Nevertheless, some points can belong to the follow set but lead to a hazardous state. The study of the reachability analysis could be performed to find viable domains in the output space. For instance, reachability analysis has already been studied using interval analysis in [97]. Then the *viable follow set* could be characterized: it would contain every point of the follow set such that from these points, the system stays in the viable follow set forever.

Handle the uncertainties in the design of the controller. The proposed approach to localize the robot leads to an uncertain position, represented by a fuzzy set. Then, this fuzzy set should be used as an input for the controller. Fuzzy control systems were first introduced by Lofti Zadeh [168] and have successfully been employed in robotic applications [118, 84]. In [165], different classes of fuzzy logic control are presented, with applications to marine robots.

Localize the deported sensor. Another improvement would be to estimate the position of the deported sensor without measuring its heading. It would be possible by studying the dynamics of the angle between the robot and the sensor, with the assumption of the taut rope. Then, an interval integration can wrap this angle and allows to approximate the position of the sensor.

Otherwise, an echosounder could be placed on the robot and directed toward the sensor. An echo should be detected. Then the fuzzy set estimation approach can deal with uncertainties and outliers. In addition, with the assumption of the taut rope and knowing the length of the rope, a strong constraint can be defined to help the localization of the sensor.

Fuzzy set estimation. The approach can constitute a base for any combination of granules. However, the score function should be defined by experts of the studied domain in order to define the priorities of each constraint properly. Thus, to show the relevance of the approach, different applications should be studied. Therefore the software project should be shared and makes the object of a contribution in the `Codac` library:

https://codac.io/.

This library already contains many tools for constraint programming and is mainly based on interval analysis.

Bibliography

[1] Fahed Abdallah, Amadou Gning, and Philippe Bonnifait. “Box particle filtering for nonlinear state estimation using interval analysis”. In: *Automatica* 44.3 (2008), pp. 807–815 (cit. on p. 35).

[2] Benedetto Allotta et al. “The ARROWS project: adapting and developing robotics technologies for underwater archaeology”. In: *IFAC-PapersOnLine* 48.2 (2015), pp. 194–199 (cit. on p. 3).

[3] J Barto Arnold III. “An underwater archeological magnetometer survey and site test excavation project off Padre Island, Texas”. In: (1976) (cit. on p. 4).

[4] Tim Bailey and Hugh Durrant-Whyte. “Simultaneous localization and mapping (SLAM): Part II”. In: *IEEE robotics & automation magazine* 13.3 (2006), pp. 108–117 (cit. on p. 13).

[5] Stephen Barkby et al. “An efficient approach to bathymetric SLAM”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 219–224 (cit. on p. 13).

[6] Stéphane Bazeille. “Vision sous-marine monoculaire pour la reconnaissance d’objets”. PhD thesis. Brest, 2008 (cit. on p. 7).

[7] Pierre Benet et al. “State-of-the-Art of Standalone Accurate AUV Positioning - Application to High Resolution Bathymetric Surveys”. In: *OCEANS 2019 - Marseille*. IEEE, June 2019 (cit. on p. 7).

[8] Frédéric Benhamou, Frédéric Goualard, and Laurent Granvillie. “Revising hull and box consistency”. In: *Logic Programming: Proceedings of the 1999 International Conference on Logic Programming*. MIT Press. 1999, p. 230 (cit. on p. 86).

[9] Joseph Berkson. “Estimation by least squares and by maximum likelihood”. English. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. 1956 (cit. on p. 18).

[10] Alan A Berryman. “The orgins and evolution of predator-prey theory”. In: *Ecology* 73.5 (1992), pp. 1530–1535 (cit. on p. 19).

[11] Vittorio Bichucher et al. “Bathymetric factor graph SLAM with sparse point cloud alignment”. In: *OCEANS 2015-MTS/IEEE Washington*. IEEE. 2015, pp. 1–7 (cit. on p. 13).

[12] Brian Bingham et al. “Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle”. In: *Journal of Field Robotics* 27.6 (2010), pp. 702–717 (cit. on p. 3).

[13] William Boltz. *The origin and early development of the Chinese writing system*. Eisenbrauns, 1994 (cit. on p. 2).

[14] Reda Boukezzoula, Sylvie Galichet, and Didier Coquin. “From fuzzy regression to gradual regression: Interval-based analysis and extensions”. In: *Information Sciences* 441 (May 2018), pp. 18–40 (cit. on p. 36).

[15] Reda Boukezzoula, Sylvie Galichet, and Laurent Foulloy. “MIN and MAX operators for fuzzy intervals and their potential use in aggregation operators”. In: *IEEE Transactions on Fuzzy Systems* 15.6 (2007), pp. 1135–1144 (cit. on p. 36).

[16] Reda Boukezzoula, Luc Jaulin, and Laurent Foulloy. “Thick gradual intervals: An alternative interpretation of type-2 fuzzy intervals and its potential use in type-2 fuzzy computations”. In: *Engineering Applications of Artificial Intelligence* 85 (2019), pp. 691–712 (cit. on p. 36).

[17] Antoni Burguera, Yolanda González, and Gabriel Oliver. “The UspIC: Performing Scan Matching Localization Using an Imaging Sonar”. In: *Sensors* 12.6 (June 2012), pp. 7855–7885 (cit. on p. 8).

[18] Antoni Burguera, Gabriel Oliver, and Yolanda González. “Scan-based SLAM with trajectory correction in underwater environments”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2010 (cit. on p. 79).

[19] Clement Carbonnel et al. “Q-intersection algorithms for constraint-based robust parameter estimation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014 (cit. on p. 119).

[20] Alexis Catsambis, Ben Ford, and Donny L Hamilton. *The Oxford handbook of maritime archaeology*. Oxford University Press, 2011 (cit. on pp. 2, 4).

[21] Martine Ceberio and Vladik Kreinovich. “Towards an optimal approach to soft constraint problems”. In: (2005) (cit. on p. 118).

[22] Gilles Chabert and Luc Jaulin. “A priori error analysis with intervals”. In: *SIAM Journal on Scientific Computing* 31.3 (2009), pp. 2214–2230 (cit. on p. 119).

[23] Gilles Chabert and Luc Jaulin. “Contractor programming”. In: *Artificial Intelligence* 173.11 (2009), pp. 1079–1100 (cit. on pp. 28, 35).

[24] Gilles Chabert and Luc Jaulin. “Hull consistency under monotonicity”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2009, pp. 188–195 (cit. on p. 29).

[25] Véronique Cherfaoui, Thierry Denoeux, and Zohra Leïla Cherfi. “Distributed data fusion: application to confidence management in vehicular networks”. In: *2008 11th international conference on information fusion*. IEEE. 2008, pp. 1–8 (cit. on p. 121).

[26] Mandar Chitre, John Potter, and Ong Sim Heng. “Underwater acoustic channel characterisation for medium-range shallow water communications”. In: *Oceans' 04 MTS/IEEE Techno-Ocean'04 (IEEE Cat. No. 04CH37600)*. Vol. 1. IEEE. 2004, pp. 40–45 (cit. on p. 8).

[27] Hyun-Taek Choi and Junku Yuh. “Underwater Robots”. In: *Springer Handbook of Robotics*. Springer International Publishing, 2016, pp. 595–622 (cit. on p. 77).

[28] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles*. Butterworth-Heinemann, 2013 (cit. on pp. 47, 76).

[29] Brian Claus and Ralf Bachmayer. “Terrain-aided navigation for an underwater glider”. In: *Journal of Field Robotics* 32.7 (2015), pp. 935–951 (cit. on p. 13).

[30] Arnaud Clérentin et al. “Uncertainty and imprecision modeling for the mobile robot localization problem”. In: *Autonomous Robots* 24.3 (2008), pp. 267–283 (cit. on p. 35).

[31] Etienne Colle and Simon Galerne. “Mobile robot localization by multiangulation using set inversion”. In: *Robotics and Autonomous Systems* 61.1 (2013), pp. 39–48 (cit. on p. 35).

[32] B. d'Andréa-Novel. *Commande non-linéaire des robots*. Paris, France: Hermès, 1988 (cit. on p. 17).

[33] P. Dario et al. “A microrobotic system for colonoscopy”. In: Albuquerque, NM, USA. Vol. 2. Albuquerque, NM, USA: IEEE, 1997, 1567–1572 vol.2 (cit. on p. 47).

[34] Martin Dean et al. *Archaeology underwater: The NAS guide to principles and practice*. Nautical Archaeology Society, 1992 (cit. on p. 2).

[35] James P Delgado. *Encyclopedia of underwater and maritime archaeology*. 1997 (cit. on p. 3).

[36] Arthur P Dempster. “Upper and lower probabilities induced by a multivalued mapping”. In: *Classic works of the Dempster-Shafer theory of belief functions*. Springer, 2008, pp. 57–72 (cit. on pp. 36, 120).

[37] Thierry Denœux. “Constructing belief functions from sample data using multinomial confidence regions”. In: *International Journal of Approximate Reasoning* 42.3 (2006), pp. 228–252 (cit. on p. 36).

[38] B. Desrochers and L. Jaulin. “A minimal contractor for the polar equation: Application to robot localization”. In: *Engineering Applications of Artificial Intelligence* 55 (2016), pp. 83–92 (cit. on pp. 28, 35, 115).

[39] B. Desrochers, S. Lacroix, and L. Jaulin. “Set-Membership Approach to the Kidnapped Robot Problem”. In: *IROS 2015*. 2015 (cit. on p. 25).

[40] Benoît Desrochers. “Simultaneous Localization and Mapping in Unstructured Environments”. PhD thesis. Université Bretagne Loire, 2018 (cit. on p. 109).

[41] NS Dimitrova, SM Markov, and ED Popova. “Extended interval arithmetics: new results and applications”. In: *Computer Arithmetic and Enclosure Methods* (1992), pp. 225–232 (cit. on p. 26).

[42] Sette Diop and Michel Fliess. “Nonlinear observability, identifiability, and persistent trajectories”. In: *Proceedings of the 30th IEEE Conference on Decision and Control*. IEEE. 1991, pp. 714–719 (cit. on p. 66).

[43] Vincent Drevelle and Philippe Bonnifait. “High integrity GNSS location zone characterization using interval analysis”. In: *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*. 2009, pp. 2178–2187 (cit. on p. 36).

[44] Michael Drumheller. “Mobile robot localization using sonar”. In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1987), pp. 325–332 (cit. on p. 89).

[45] L. E. Dubins. “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents”. In: *American Journal of Mathematics* 79.3 (July 1957), p. 497 (cit. on p. 48).

[46] Didier Dubois, Luc Jaulin, and Henri Prade. “Thick sets, multiple-valued mappings, and possibility theory”. In: *Statistical and Fuzzy Approaches to Data Processing, with Applications to Econometrics and Other Areas*. Springer, 2020, pp. 101–109 (cit. on p. 36).

[47] Didier Dubois and Henri Prade. *Possibility theory: an approach to computerized processing of uncertainty*. Springer Science & Business Media, 2012 (cit. on p. 36).

[48] Didier Dubois, Henri Prade, and Henri M Prade. *Fundamentals of Fuzzy Sets*. Vol. 7. Springer Science & Business Media, 2000 (cit. on p. 38).

[49] Didier J Dubois. *Fuzzy sets and systems: theory and applications*. Vol. 144. Academic press, 1980 (cit. on pp. 36, 39).

[50] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110 (cit. on p. 13).

[51] Denis Efimov et al. “Interval state observer for nonlinear time varying systems”. In: *Automatica* 49.1 (Jan. 2013), pp. 200–205 (cit. on p. 18).

[52] Brian Fagan. *The Oxford companion to archaeology*. New York: Oxford University Press, 1996 (cit. on p. 2).

[53] I. Fantoni and R. Lozano. *Non-linear control for underactuated mechanical systems*. Springer-Verlag, 2001 (cit. on p. 18).

[54] Kenneth A Farley et al. “Mars 2020 mission overview”. In: *Space Science Reviews* 216.8 (2020), pp. 1–41 (cit. on p. 47).

[55] Michel Fliess et al. “Flatness and defect of non-linear systems: introductory theory and examples”. In: *International journal of control* 61.6 (1995), pp. 1327–1361 (cit. on p. 48).

[56] Eli Fogel and Y.F. Huang. “On the value of information in system identification—Bounded noise case”. In: *Automatica* 18.2 (Mar. 1982), pp. 229–238 (cit. on p. 35).

[57] Guilherme Schvarcz Franco and Fabrice Le Bars. “Robust polygon-based localization”. In: *2018 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, Feb. 2018 (cit. on p. 7).

[58] Donald Frey. “Sub-bottom survey of Porto Longo harbour, Peleponnesus, Greece”. In: *International Journal of Nautical Archaeology* 1.1 (1972), pp. 170–175 (cit. on p. 4).

[59] G. El-Ghazaly, M. Gouttefarde, and V. Creuze. “Hybrid cable-thruster actuated underwater vehicle-manipulator systems: A study on force capabilities”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IROS, Grenoble, France, 2005 (cit. on p. 18).

[60] Amadou Gning and Ph Bonnifait. “Constraints propagation techniques on intervals for a guaranteed localization using redundant data”. In: *Automatica* 42.7 (2006), pp. 1167–1175 (cit. on p. 35).

[61] David Gouaillier et al. “Mechatronic design of NAO humanoid”. In: Kobe, Japan. Kobe, Japan: IEEE, 2009, pp. 769–774 (cit. on p. 47).

[62] Jeremy Green. *Maritime archaeology: a technical handbook*. Routledge, 2016 (cit. on p. 4).

[63] Max Guérout. “Recherche archéologique de l’épave de la Cordelière (1512)”. In: Oct. 2017 (cit. on p. 5).

[64] Max Guérout. “Recherche magnétique”. In: (Oct. 2017). Le texte analyse les conditions pratiques d’une prospection magnétique sous-marine à l’aide d’un magnétomètre remorqué. (cit. on p. 5).

[65] R. Guyonneau, S. Lagrange, and L. Hardouin. “A Visibility Information for Multi-Robot Localization”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013 (cit. on p. 25).

[66] ET Hall. “The use of the proton magnetometer in underwater archaeology”. In: *Archaeometry* 9.1 (1966), pp. 32–43 (cit. on p. 4).

[67] Michael Hanss. *Applied fuzzy arithmetic*. Springer, 2005 (cit. on p. 39).

[68] Emili Hernàndez Bes et al. “MSISpIC: A probabilistic scan matching algorithm using a mechanical scanned imaging sonar”. In: *Journal of physical agents, 2009, vol. 3, núm. 1, p. 3-11* (2009) (cit. on p. 80).

[69] A. Isidori. *Nonlinear Control Systems: An Introduction, 3rd Ed.* New-York: Springer-Verlag, 1995 (cit. on p. 17).

[70] Zool H. Ismail and Iksan Bukhori. “Efficient Detection of Robot Kidnapping in Range Finder-Based Indoor Localization Using Quasi-Standardized 2D Dynamic Time Warping”. In: *Applied Sciences* 11.4 (Feb. 2021), p. 1580 (cit. on p. 7).

[71] Luc Jaulin. “A nonlinear set membership approach for the localization and map building of underwater robots”. In: *IEEE Transactions on Robotics* 25.1 (2009), pp. 88–98 (cit. on pp. 12, 80).

[72] Luc Jaulin. “Robust set-membership state estimation; application to underwater robotics”. In: *Automatica* 45.1 (Jan. 2009), pp. 202–206 (cit. on p. 78).

[73] Luc Jaulin. “Set-membership localization with probabilistic errors”. In: *Robotics and Autonomous Systems* 59.6 (2011), pp. 489–495 (cit. on p. 36).

[74] Luc Jaulin and Benoît Desrochers. “Introduction to the algebra of separators with application to path planning”. In: *Engineering Applications of Artificial Intelligence* 33 (2014), pp. 141–147 (cit. on pp. 30, 109).

[75] Luc Jaulin and Eric Walter. “Guaranteed robust nonlinear minimax estimation”. In: *IEEE Transactions on Automatic Control* 47.11 (2002), pp. 1857–1864 (cit. on p. 32).

[76] Luc Jaulin and Eric Walter. “Set inversion via interval analysis for nonlinear bounded-error estimation”. In: *Automatica* 29.4 (1993), pp. 1053–1064 (cit. on p. 31).

[77] Luc Jaulin et al. “Guaranteed non-linear estimation using constraint propagation on sets”. In: *International Journal of Control* 74.18 (2001), pp. 1772–1782 (cit. on p. 35).

[78] Luc Jaulin et al. “Guaranteed robust nonlinear estimation with application to robot localization”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32.4 (2002), pp. 374–381 (cit. on p. 33).

[79] Luc Jaulin et al. “Guaranteed robust nonlinear estimation with application to robot localization”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32.4 (2002), pp. 374–381 (cit. on p. 91).

[80] Luc Jaulin et al. “Interval analysis”. In: *Applied interval analysis*. Springer, 2001, pp. 11–43 (cit. on p. 28).

[81] Leopoldo Jetto, Sauro Longhi, and Giuseppe Venturini. “Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots”. In: *IEEE Transactions on Robotics and Automation* 15.2 (1999), pp. 219–229 (cit. on p. 35).

[82] Rudolf Emil Kalman et al. “Contributions to the theory of optimal control”. In: *Bol. soc. mat. mexicana* 5.2 (1960), pp. 102–119 (cit. on pp. 12, 18, 35).

[83] Arnold Kaufman. *Introduction to fuzzy arithmetic*. 1991 (cit. on pp. 39, 40).

[84] Rafael Kelly et al. “Lyapunov Stable Control of Robot Manipulators: A Fuzzy Self-Tuning Procedure”. In: *Intelligent Automation & Soft Computing* 5.4 (Jan. 1999), pp. 313–326 (cit. on p. 127).

[85] Hassan K. Khalil. *Nonlinear systems. 3rd ed*. English. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002, pp. xv + 750 (cit. on pp. 17, 25).

[86] Michel Kieffer et al. “Robust autonomous robot localization using interval analysis”. In: *Reliable computing* 6.3 (2000), pp. 337–362 (cit. on p. 89).

[87] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. Prentice hall New Jersey, 1995 (cit. on p. 38).

[88] Vipin Kumar. “Algorithms for constraint-satisfaction problems: A survey”. In: *AI magazine* 13.1 (1992), pp. 32–32 (cit. on p. 12).

[89] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*. Vol. 1. Wiley-interscience New York, 1972 (cit. on p. 17).

[90] Sébastien Lagrange et al. “Nonlinear blind parameter estimation”. In: *IEEE transactions on automatic control* 53.3 (2008), pp. 834–838 (cit. on p. 35).

[91] Hélène Lahanier, Eric Walter, and Roberto Gomeni. “OMNE: a new robust membership-set estimator for the parameters of nonlinear models”. In: *Journal of Pharmacokinetics and Biopharmaceutics* 15.2 (1987), pp. 203–219 (cit. on pp. 33, 118, 120).

[92] Matheus Laranjeira, Claire Dune, and Vincent Hugel. “Catenary-based visual servoing for tether shape control between underwater vehicles”. In: *Ocean Engineering* 200 (Mar. 2020), p. 107018 (cit. on p. 10).

[93] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006 (cit. on p. 66).

[94] Fabrice Le Bars and Luc Jaulin. “An experimental validation of a robust controller with the VAIMOS autonomous sailboat”. In: *Robotic Sailing 2012*. Springer, 2013, pp. 73–84 (cit. on p. 47).

[95] Fabrice Le Bars et al. “Estimating the trajectory of low-cost autonomous robots using interval analysis: Application to the eurathlon competition”. In: *Marine Robotics and Applications*. Springer, 2018, pp. 51–68 (cit. on p. 8).

[96] Fabrice Le Bars et al. “Set-membership state estimation with fleeting data”. In: *Automatica* 48.2 (2012), pp. 381–387 (cit. on p. 35).

[97] Thomas Le Mézo, Luc Jaulin, and Benoit Zerr. “Bracketing the solutions of an ordinary differential equation with uncertain initial conditions”. In: *Applied Mathematics and Computation* 318 (2018), pp. 70–79 (cit. on pp. 67, 127).

[98] Christophe Lecoutre. *Constraint Networks*. John Wiley & Sons, Mar. 2013. 320 pp. (cit. on p. 27).

[99] John J Leonard and Alexander Bahr. “Autonomous underwater vehicle navigation”. In: *Springer handbook of ocean engineering* (2016), pp. 341–358 (cit. on p. 8).

[100] John J Leonard and Hugh F Durrant-Whyte. “Mobile robot localization by tracking geometric beacons”. In: *IEEE Transactions on robotics and Automation* 7.3 (1991), pp. 376–382 (cit. on pp. 79, 89).

[101] Margaret E Leshikar-Denton and Pilar Luna Erreguerena. *Underwater and maritime archaeology in Latin America and the Caribbean*. Vol. 56. Routledge, 2016 (cit. on p. 3).

[102] Volker Lohweg, Karl Voth, and Stefan Glock. “A Possibilistic Framework for Sensor Fusion with Monitoring of Sensor Reliability”. In: *Sensor Fusion - Foundation and Applications*. InTech, June 2011 (cit. on p. 43).

[103] Luc Longpré and Christian Servin. “Quantum computation techniques for gauging reliability of interval and fuzzy data”. In: *NAFIPS 2008-2008 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE. 2008, pp. 1–6 (cit. on pp. 103, 104).

[104] Michael Binsar Lubis, Mehrdad Kimiaei, and Mike Efthymiou. “Alternative configurations to optimize tension in the umbilical of a work class ROV performing ultra-deep-water operation”. In: *Ocean Engineering* 225 (Apr. 2021), p. 108786 (cit. on p. 10).

[105] Xavier Lurton. *An introduction to underwater acoustics : principles and applications*. London New York: Springer, 2002 (cit. on pp. 4, 9).

[106] Ronald PS Mahler. “Multitarget Bayes filtering via first-order multitarget moments”. In: *IEEE Transactions on Aerospace and Electronic systems* 39.4 (2003), pp. 1152–1178 (cit. on p. 13).

[107] D. G. Maksarov and J. P. Norton. “State bounding with ellipsoidal set description of the uncertainty”. In: *International Journal of Control* 65.5 (Nov. 1996), pp. 847–866 (cit. on p. 13).

[108] Angelos Mallios et al. “Scan matching SLAM in underwater environments”. In: *Autonomous Robots* 36.3 (June 2013), pp. 181–198 (cit. on p. 79).

[109] Angelos Mallios et al. “Toward Autonomous Exploration in Confined Underwater Environments”. In: *Journal of Field Robotics* 33.7 (Nov. 2015), pp. 994–1012 (cit. on p. 80).

[110] M. Di Marco et al. “Set membership localization and mapping for autonomous navigation”. In: *InternationalJournal of Robust and Nonlinear Control* 7.11 (2001), pp. 709–734 (cit. on p. 35).

[111] Dominique Meizel et al. “Initial localization by set inversion”. In: *IEEE transactions on robotics and Automation* 18.6 (2002), pp. 966–971 (cit. on p. 31).

[112] N. Meslem, N. Loukkas, and J.J. Martinez. “Using set invariance to design robust interval observers for discrete time linear systems”. In: *International Journal of Robust and Nonlinear Control* (2018), pp. 1–17 (cit. on p. 25).

[113] Aaron Meurer et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (Jan. 2017), e103 (cit. on p. 52).

[114] Peter Henry Milne. “Underwater acoustic positioning systems”. In: (1983) (cit. on p. 8).

[115] Ramon E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, Jan. 1979 (cit. on p. 35).

[116] Ghalia Nassreddine, Fahed Abdallah, and Thierry Denoux. “State estimation using interval analysis and belief-function theory: application to dynamic vehicle localization”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40.5 (2009), pp. 1205–1218 (cit. on pp. 36, 120).

[117] Renata Neuland et al. “Robust Hybrid Interval-Probabilistic Approach for the Kidnapped Robot Problem”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 29.2 (2021), pp. 313–331 (cit. on p. 36).

[118] A. Ollero, A. García-Cerezo, and J.L. Martínez. “Fuzzy supervisory path tracking of mobile reports”. In: *Control Engineering Practice* 2.2 (Apr. 1994), pp. 313–319 (cit. on p. 127).

[119] A.K. Al-Othman and M.R. Irving. “Robust state estimator based on maximum constraints satisfaction of uncertain measurements”. In: *Measurement* 40.3 (Apr. 2007), pp. 347–359 (cit. on p. 12).

[120] JG Paglia and WF Wyman. “DARPA’S autonomous minehunting and mapping technologies (AMMT) program an overview”. In: *OCEANS 96 MTS/IEEE Conference Proceedings. The Coastal Ocean-Prospects for the 21st Century*. Vol. 2. IEEE. 1996, pp. 794–799 (cit. on p. 7).

[121] A. Papoulis and H. Saunders. “Probability, Random Variables and Stochastic Processes (2nd Edition)”. In: 111 (1989), pp. 123–125 (cit. on p. 12).

[122] H Piet-Lahanier and E Walter. “Characterization of non-connected parameter uncertainty regions”. In: *Mathematics and computers in simulation* 32.5-6 (1990), pp. 553–560 (cit. on p. 35).

[123] J.M. Porta et al. “ A Space Decomposition Method for Path Planning of Loop Linkages”. In: *Proceedings of International Conference on Intelligent Robots and Systems, IROS*. 2007, pp. 1882–1888 (cit. on pp. 9, 25).

[124] Promotion 2021, Robotique autonome. *Projet Magmap*. Tech. rep. Available at `https://www.ensta-bretagne.fr/jaulin/magmap_rapport.pdf`. ENSTA Bretagne, 2021 (cit. on p. 63).

[125] R. E. Moore. *Interval Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1966 (cit. on p. 25).

[126] Louis B Rall. *Automatic differentiation: Techniques and applications*. Springer-Verlag, 1981 (cit. on p. 52).

[127] Andreas Rauh, Auguste Bourgois, and Luc Jaulin. “Union and Intersection Operators for Thick Ellipsoid State Enclosures: Application to Bounded-Error Discrete-Time State Observer Design”. In: *Algorithms* 14.3 (Mar. 2021), p. 88 (cit. on p. 18).

[128] Andreas Rauh, Julia Kersten, and Harald Aschemann. “An Interval Observer Approach for the Online Temperature Estimation in Solid Oxide Fuel Cell Stacks”. In: *2018 European Control Conference (ECC)*. IEEE, June 2018 (cit. on p. 18).

[129] Ilona Regulski. *The Origins and Early Development of Writing in Egypt*. Oxford University Press, May 2016 (cit. on p. 2).

[130] Eva Reitbauer and Christoph Schmied. “Bridging GNSS Outages with IMU and Odometry: A Case Study for Agricultural Vehicles”. In: *Sensors* 21.13 (June 2021), p. 4467 (cit. on p. 8).

[131] Olivier Reynet, Luc Jaulin, and Gilles Chabert. “Robust TDOA passive location using interval analysis and contractor programming”. In: *2009 International Radar Conference" Surveillance for a Safer World"(RADAR 2009)*. IEEE. 2009, pp. 1–6 (cit. on p. 35).

[132] David Ribas et al. “Underwater SLAM in man-made structured environments”. In: *Journal of Field Robotics* 25.11-12 (Nov. 2008), pp. 898–921 (cit. on p. 79).

[133] S. Rohou et al. *Reliable robot localization. A constraint-programming approach over dynamical systems*. ISTE Group, 2019 (cit. on pp. 25, 35).

[134] Simon Rohou, Benoît Desrochers, and Luc Jaulin. “Set-membership state estimation by solving data association”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4393–4399 (cit. on p. 78).

[135] Simon Rohou et al. “Reliable non-linear state estimation involving time uncertainties”. In: *Automatica* 93 (2018), pp. 379–388 (cit. on p. 35).

[136] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006 (cit. on p. 27).

[137] P. Rouchon et al. “Flatness, motion planning and trailer systems”. In: *Proceedings of 32nd IEEE Conference on Decision and Control*. Vol. 3. Dec. 1993, pp. 2700–2705 (cit. on p. 57).

[138] Carol V. Ruppé and Janet F. Barstad, eds. *International Handbook of Underwater Archaeology*. Springer US, 2002 (cit. on p. 3).

[139] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2010 (cit. on p. 27).

[140] Zsofi Ruttkay. “Fuzzy constraint satisfaction”. In: *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*. IEEE. 1994, pp. 1263–1268 (cit. on p. 12).

[141] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Dec. 1976 (cit. on pp. 43, 120).

[142] Jan Sliwka. “Using set membership methods for robust underwater robot localization”. PhD thesis. ENSTA Bretagne, 2011 (cit. on p. 87).

[143] Jean-Jacques E. Slotine and Weiping Li. *Applied nonlinear control*. English. Englewood Cliffs, NJ: Prentice-Hall, 1991, pp. xv + 459 (cit. on pp. 17, 24).

[144] Randall Smith, Matthew Self, and Peter Cheeseman. “Estimating uncertain spatial relationships in robotics”. In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193 (cit. on p. 9).

[145] D. Soetanto, L. Lapierre, and A. Pascoal. “Adaptive, Non-Singular Path-Following Control of Dynamic Wheeled Robots”. In: *IEEE Conference on Decision and Control*. Vol. 2. 2003, pp. 1765–1770 (cit. on p. 17).

[146] H. W. Sorenson. “Least-squares estimation: from Gauss to Kalman”. In: *IEEE Spectrum* 7 (7 1970), pp. 63–68 (cit. on p. 18).

[147] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. “Simultaneous localization and mapping”. In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1153–1176 (cit. on p. 13).

[148] Gerald Jay Sussman and Jack Wisdom. *Functional differential geometry*. The MIT Press, 2013 (cit. on p. 52).

[149] Yew Teck Tan, Mandar Chitre, and Franz S. Hover. “Cooperative bathymetry-based localization using low-cost autonomous underwater vehicles”. In: *Autonomous Robots* 40.7 (Oct. 2015), pp. 1187–1205 (cit. on p. 13).

[150] Sebastian Thrun. “Probabilistic algorithms in robotics”. In: *Ai Magazine* 21.4 (2000), pp. 93–109 (cit. on p. 25).

[151] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005 (cit. on pp. 13, 35, 79).

[152] Joris Tillet. “Wrecks research with underwater robots”. MA thesis. ENSTA Bretagne, 2018 (cit. on p. 10).

[153] Joris Tillet, Luc Jaulin, and Fabrice Le Bars. “Non-linear control under state constraints with validated trajectories for a mobile robot towing a trailer”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020 (cit. on p. 127).

[154] Ornella Tortorici et al. “Towards active self-management of umbilical linking ROV and USV for safer submarine missions”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 265–270 (cit. on p. 10).

[155] Gilles Trombettoni and Gilles Chabert. “Constructive Interval Disjunction”. In: *Principles and Practice of Constraint Programming – CP 2007*. Springer Berlin Heidelberg, 2007, pp. 635–650 (cit. on p. 119).

[156] Edward Tsang. *Foundations of Constraint Satisfaction*. Books on Demand, May 2014. 444 pp. (cit. on p. 12).

[157] Jerome Vaganay, John J Leonard, and James G Bellingham. “Outlier rejection for autonomous acoustic navigation”. In: *Proceedings of IEEE international conference on robotics and automation*. Vol. 3. IEEE. 1996, pp. 2174–2181 (cit. on p. 9).

[158] S. M. Veres and J. P. Norton. “Parameter-Bounding Algorithms for Linear Errors-in-Variables Models”. In: *Bounding Approaches to System Identification*. Ed. by Mario Milanese et al. Boston, MA: Springer US, 1996, pp. 275–288 (cit. on p. 13).

[159] Christophe Viel. “Self-management of the umbilical of a ROV for underwater exploration”. working paper or preprint. July 2021 (cit. on p. 10).

[160] B-N Vo and W-K Ma. “The Gaussian mixture probability hypothesis density filter”. In: *IEEE Transactions on signal processing* 54.11 (2006), pp. 4091–4104 (cit. on pp. 13, 18).

[161] Eric Walter and Hélène Piet-Lahanier. “Estimation of the parameter uncertainty resulting from bounded-error data”. In: *Mathematical Biosciences* 92.1 (Nov. 1988), pp. 55–74 (cit. on p. 13).

[162] Eric Walter and Luc Pronzato. *Identification of parametric models: from experimental data*. Springer Verlag, 1997 (cit. on p. 35).

[163] Jinkun Wang, Shi Bai, and Brendan Englot. “Underwater localization and 3D mapping of submerged structures with a single-beam scanning sonar”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4898–4905 (cit. on p. 79).

[164] Cory White et al. “The Malta cistern mapping project: Underwater robot mapping and localization within ancient tunnel systems”. In: *Journal of Field Robotics* 27.4 (Mar. 2010), pp. 399–411 (cit. on p. 79).

[165] Xianbo Xiang et al. “Survey on Fuzzy-Logic-Based Guidance and Control of Marine Surface Vehicles and Underwater Vehicles”. In: *International Journal of Fuzzy Systems* 20.2 (Oct. 2017), pp. 572–586 (cit. on p. 127).

[166] K. Yamafune, R. Torres, and F. Castro. “Multi-Image Photogrammetry to Record and Reconstruct Underwater Shipwreck Sites”. In: *Journal of Archaeological Method and Theory* 24.3 (Mar. 2016), pp. 703–725 (cit. on p. 3).

[167] Lotfi A Zadeh. “Fuzzy sets”. In: *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*. World Scientific, 1996, pp. 394–432 (cit. on p. 34).

[168] Lotfi A Zadeh. “Information and control”. In: *Fuzzy sets* 8.3 (1965), pp. 338–353 (cit. on pp. 34, 127).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE — Lab-STICC — AGENCE INNOVATION DÉFENSE — Région BRETAGNE

Titre : Localisation et contrôle sûrs d'un capteur tracté

Mots-clés : Localisation Sous-marine, Contrôle Non Linéaire, Analyse par Intervalles, Logique Floue, Estimation d'Ensemble Flou, Capteur Tracté

Résumé : L'exploration des océans devient de plus en plus accessible, notamment grâce aux avancées en robotique. Les applications pour les robots sous-marins sont nombreuses. Dans cette thèse, on s'intéresse particulièrement à la recherche d'épaves, telle que celle de La Cordelière, qui a coulé dans la Rade de Brest en 1512.
Le système robotique proposé consiste à tracter un magnétomètre susceptible de détecter les matériaux ferromagnétiques de l'épave. Le capteur ne peut pas être directement embarqué car il est sensible aux perturbations du robot. C'est pourquoi il est déporté.
Deux problématiques sont alors étudiées pour appréhender ce système.
La première est liée au contrôle de la position du magnétomètre alors que l'on ne peut agir que sur le robot tractant. Une méthode de linéarisation par bouclage est alors utilisée pour construire un contrôleur. Ce contrôleur est ensuite validé sous certaines contraintes d'état en utilisant des outils d'analyse par intervalles.
La seconde problématique concerne la localisation sous l'eau de manière fiable. Sont alors étudiés des moyens d'appréhender les incertitudes et les données aberrantes collectées par un capteur acoustique. L'analyse par intervalles permet d'obtenir des premiers résultats, et la logique floue vient compléter l'approche en donnant plus de souplesse dans la priorisation des contraintes. Finalement, des expérimentations sont présentées avec différents robots, et notamment la localisation d'un ROV dans une piscine.

Title: Safe localization and control of a towed sensor

Keywords: Underwater Localization, Non-linear Control, Interval Analysis, Fuzzy Logic, Fuzzy Set Estimation, Towed Sensor

Abstract: The oceans' exploration becomes more and more reachable, especially thanks to robotics progress. Applications for underwater robots are plentiful. In this thesis, we particularly focus on the search of wrecks, as the *Cordelière*, which sank in the Bay of Brest (France) in 1512.
The proposed robotic system consists of towing a magnetometer likely to detect the ferromagnetic materials of the wreck. The sensor cannot be directly embedded because it is sensitive to the perturbations from the robot. This is why it is deported.
Two issues are studied to approach this system.
The first one is linked to the control of the magnetometer's position, whereas we can only act on the towing robot. A feedback linearization method is used to design a controller. Then, this controller is validated under some state constraints by using tools from interval analysis.
The second issue relates to underwater localization in a reliable manner. Ways to approach uncertainties and outliers gathered by acoustic sensors are studied. The interval analysis allows to obtain first results, and the fuzzy logic completes the approach by giving more suppleness in the prioritization of the constraints.
Finally, some expérimentations are presented with different robots, and especially the localization of an ROV in a pool.