THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE
Mention : Robotique
École Doctorale SICMA

présentée par
Jan Śliwka
préparée à l'ENSTA Bretagne (ex ENSIETA), Equipe OSM, Pôle STIC

Using set membership methods for robust underwater robot localization

Soutenue le 6 décembre 2011
devant le jury composé de :

Luc JAULIN
Professeur des Universités, ENSTA Bretagne / *directeur de thèse*

Olivier REYNET
Maître de Conférence, ENSTA Bretagne / *co-directeur de thèse*

Laurent HARDOUIN
Professeur des Universités, Université d'Angers ISTIA / *rapporteur*

Bruno JOUVENCEL
Professeur des Universités, Université de Montpellier / *rapporteur*

Rogelio LOZANO
Directeur de recherche, CNRS / *président du Jury*

What is free today might not stay that way in the future.

Jan S.

Acknowledgement

I would like to first thank all the people who helped me during my thesis and especially Pr. Luc Jaulin for his perfect guidance, Dr. Olivier Reynet for precious advice, Dr. Fabrice Le Bars for his technical expertise in all robotics projects and all science lab people with whom I had very interesting conversations which also inspired me in all my personal projects. I also thank Pr. Laurent Hardouin, Pr. Bruno Jouvencel and Pr. Rogelio Lozano for rigorously examining my thesis. The examination allowed me to look at my subject from a broader perspective. I finally thank the BMO (Brest Metropole Oceane) for financial support as well as ENSTA-Bretagne and its administrative personnel for letting me work in the school's premises in perfect conditions.

I dedicate this thesis for my ancestors, my parents, my sister and all my future descendants.

Preface

Hello, my name is Jan SLIWKA and I am an engineer specialized in embedded systems and automatics. I got my engineer diploma at ENSTA-Bretagne (Previously ENSIETA), an engineering school in Brest, France. Since October 2008, three years now, I am pursuing the Ph.D. degree at the same school under the guidance of Pr. Luc JAULIN and Dr. Olivier REYNET. The Ph.D. is about relaxed solving of systems of equations which can be applied to many fields. I implemented this theory mainly to solve the problem of localization of underwater robots. Scientific work aside, I participated to numerous robotics projects (underwater robots, sailling robots, flying robots, ground robots) and was involved in the school robotics club activities. As an example, each year, we participate to an autonomous underwater vehicle competition called SAUC'E. I was also involved in the construction of a sailing boat for the Microtransat Challenge which purpose is to cross the Atlantic Ocean autonomously. I invite you now to read my thesis and know more about my research.

Contents

List of Figures

Notations

There are several notations used in the manuscript. Those notations are listed below

- $\mathbb{R}$: the set of real numbers
- Sets are denoted by blackboard bold characters. *ex* : $\mathbb{A}, \mathbb{B}, \mathbb{C}, ...$
- The vectors and vector functions are written in bold characters. *ex* : $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n, \mathbf{x} \in \mathbb{R}^m$
- $[x]$: the interval of possible values of the variable x.
- x^+, x^- : respectively upper and lower bound of the interval $[x]$
- $[\mathbf{x}] = ([x_1], .., [x_n])$: the box of possible values of the vector variable $\mathbf{x} \in \mathbb{R}^n$.
- $[\mathbf{f}]$: the inclusion function of the function $\mathbf{f}$
- $[\mathbb{A}]$: the box enclosing the set $\mathbb{A}$.
- $\mathbb{S}$: Solution set of a CSP
- $\mathbb{S}_q$: q-relaxed solution set of a relaxed CSP
- $\mathbb{IR}$: set of all the intervals of $\mathbb{R}$
- $\mathbb{IR}^n$: set of all the boxes of $\mathbb{R}^n$
- $\mathcal{P}$ $(\mathbb{R}^n)$: set of all subsets of $\mathbb{R}^n$
- $\mathcal{T}$: the sort transform
- $\mathcal{LP}(\mathbb{E})$: the set of lattice polynomials defined on the lattice $\mathbb{E}$.
- $\overrightarrow{\mathcal{LP}}(\mathbb{E})$: the set of nested lattice polynomials defined on the lattice $\mathbb{E}$.
- Accumulators are denoted using calligraphic characters. *ex* : $\mathcal{A}, \mathcal{B}, \mathcal{X}, \mathcal{Y}...$

Chapter 1

Introduction

1.1 Localization of underwater robots

For an intelligent robot to be able to interact properly with its environment, it has to know in one hand the environment and in the other hand its state in that environment. Using several sensors and knowing the map of the environment, localization algorithms allow to compute the position and orientation of the robot. Since the appearance of GPS, the problem of localization has been practically solved on the ground. GPS doesn't work underwater since high frequency electromagnetic waves don't propagate in that environment. However, the number of undersea operations increases significantly every year. This is due to the development of fields such as exploitation of resources (e.g. deep-water off-shore structure inspection), oceanography, biology, wreck exploration, security (e.g. port protection), military (e.g. minesweeping, mine-laying). As a consequence, there is an increasing need for underwater robots considering the dangerousness of those operations. The current type of vehicles used during those operations are remotely operated vehicles (ROVs). The ROVs are connected to the control station through an umbilical cable used for energy and communication as a consequence they require an important infrastructure (especially for deep-water models). The alternative is to use autonomous underwater vehicles also called AUVs for those tasks (See [Veres et al., 2008] for AUV designs). AUVs do not need any umbilical cable. On the other hand, since the communication with the control center is limited, AUVs has to be intelligent and rely on advanced algorithms such as localization, SLAM (simultaneous localization and mapping), image processing and path planning. The AUV also needs to be able to autonomously return home after the mission [Baccou and Jouvencel, 2002]. Many approaches have been proposed to address those problems such as probabilistic methods [Thrun et al., 2005] [Clérentin et al., 2008]. In this manuscript, probabilistic methods are not presented. This thesis focuses on the improvements made to set-membership methods

and particularly methods using interval computations. The problem of robot localization using set membership methods have been addressed by many authors [Meizel et al., 1996] [Halbwachs and Meizel, 1996] [Gning, 2006] [Kieffer, 1999] in the case where the problem is linear or not and also [Caiti et al., 2002] when the robot is underwater. In situations where strong nonlinearities are involved, interval computations has been shown to be useful (see, e.g., [Meizel et al., 2002], where the first localization of an actual robot has been solved with interval methods). The principle is that both input data and robots position are represented by their respective belonging sets. Constraints between the position of the robot and the sensor observations are used to contract the actual position set *i.e.* reduce its size thus increasing the estimates precision. Another strong point of set membership methods is the ability to deal with outliers [Jaulin and Walter, 2002] [Jaulin et al., 1996]. As an example, this property has been used to improve the localization in the city using a GPS sensor which data is usually corrupted by multipath [Drevelle and Bonnifait, 2009]. Those methods were also used for robust to outliers underwater robot localization [Jaulin, 2009].

In this PhD thesis, we focused on how to better deal with outliers when their number is unknown and may vary with time. We also present other contributions to set membership theory such as contractors theory. The usefulness of most of those contributions is shown through different localization problem examples.

1.2 Approach to solve the localization problem

Any localization problem involves four main concepts being the environment, the map of the environment, the pose (position and orientation) and the measurements of the environment. For each measurement of the environment is associated an equation (or set of equations) linking the map, the pose and the measurement. As such, the problem of localization can be formulated as a constraint satisfaction problem or CSP. A CSP can be seen as a set of equations (or constraints) involving variables to be determined (In the case of localization, the position of the robot is such a variable). Each of these variable is known to belong to a known set called domain or search space. In our case, each constraint can be considered as a representation of the information on the position of the robot compiling the data from both the measurement and the map. The more constraints there are, the more information there is about the position of the robot and the better is its estimation. The solution of a CSP is the set of points (positions) which satisfy all the constraints. To solve the CSP one have to assume that all sensor measurements are correct. The main sensor used for localization is a sector scan sonar. Very often, the sensor doesn't provide correct measurements *i.e.* the measurements of the real environment doesn't correspond to the theoretical measurement of a model sensor in a model environment represented by the map. We call this measurement an outlier. An outlier may be due to

either sensor electrical failure or a phenomenon not taken into account when modeling the environment (unaccounted objects which are detected by the sensor, physical phenomena such as multiple echoes in the case of sonar ...). A localization problem when there are outliers in the data is then cast into a set of equation where not all the equations have to be satisfied *i.e.* a CSP where not all the constraints have to be satisfied. We call this problem a *relaxed constraint satisfaction problem* or *relaxed CSP*. The number of outliers in the data used for example for localization is *a priori* unknown and may vary with time. A better characterization of the solution of a *relaxed CSP* allow thus to deal better with those outliers. Characterizing and finding the solution of a *relaxed CSP* is one of the major contributions to the PhD thesis.

One of the hurdles to overcome to solve localization problems is the representation of the map. In case of structured environments, it is possible to represent the map by a set of parametered objects such as segments, polygons, curves. In case of unstructured maps such as seashore or lake borders, the idea is to represent the map (which actually is a set) in the form of a binary image where pixels of interest (black for example) represents the set of points of the map. The point is then to be able to use the binary image representation in CSP or *relaxed CSP* computer solvers. One of the set membership approaches to solve CSPs is to use an algorithmic representation of the constraints called contractors. If the constraint defines a set of points which satisfy (or not) the constraint then the associated contractor enables to compute an approximation of that set on a computer. The contractors can be then combined to create a unique contractor allowing to compute the solution set. In case of localization problem we have a set of constraints (one for each measurement). Each one of those constraints can actually be decomposed into several constraints. One of those constraints is the *"belong to the map"* constraint. The second major contribution to the thesis was to define the contractor associated to the constraint *"belong to the map"*. That contractor is called the *image contractor*.

In a nutshell, using the newly developed tools, we propose an approach enabling robust to outliers robot localization in any unstructured environment.

1.3 Contributions

This is the list of all the contributions to set membership theory. During the PhD thesis, we worked on how to characterize the solution of a *relaxed constraint satisfaction problem i.e.* a CSP where some of the constraints can be let unsatisfied (See section 2.2 for the definition). A localization problem when there are outliers in the data can be cast into a *relaxed CSP*. The number of outliers in the data used for example for localization is *a priori* unknown and may vary with time. A better characterization of the solution of a *relaxed CSP* allow thus to deal better with those outliers. This theory is presented in

chapter 2.

The first contribution in this context is to represent the solution of a *relaxed CSP* in the form of a polynomial with set valued coefficients also called *set polynomial*. Each coefficient of this polynomial is the set of elements satisfying a specific number of constraints which actually corresponds to the degree of that coefficient in the polynomial. This idea is developed in section 2.5.

The second contribution in the same context is an alternative representation of the solution of a *relaxed CSP*. We define a function called *accumulator* which for each element returns the number of constraints it satisfies. In the manuscript, we highlight the links between *accumulators* and fuzzy logic [Klir and Yuan, 1995] [Nguyen and Walker, 2005] [Zadeh, 1965] and the generalized Hough transform [Bovik, 2000]. The accumulators are introduced in section 2.6.

Both *set polynomials* and *accumulators* have interesting properties with regards to distributed computations. The fusion of the information is performed by polynomial product in the case of *set polynomials* and sum in the case of *accumulators*. The *Set polynomial* and *accumulator* based relaxed CSP solvers are implemented using interval analysis [Kearfott and Kreinovich, 1996] [Jaulin et al., 2001] and contractor theory [Jaulin et al., 2001] [Chabert and Jaulin, 2009a] which are explained in chapters 3 and 4 respectively. The implementation of *set polynomials* and *accumulators* is explained in a separate chapter (chapter 5). The idea is to separate the mathematical theory from the implementation which might not be unique. An implementation of the *set polynomials* is explained in section 5.1. An implementation of the *accumulators* is explained in section 5.2.

The third contribution is the definition of the *image contractor*. Contractors are algorithmic entities used to compute an approximation (in the form of subpavings) of the set they represent. The *image contractor* is the contractor associated to the set defined by black (or white) pixels on a binary image. The *image contractor* allows to represent hardly parametrable sets such as maps in the context of localization. The *image contractor* can also be used to approximate complex contractors which require heavy computations. All we need is to generate the image ("take the picture") of the set associated to the constraint we need to use. The *image contractor* is introduced in section 4.8.

The fourth contribution also deals with contractors and provides theorems allowing to construct minimal contractors from other minimal contractors. The first theorem claims that the union of minimal contractors is also minimal. The second theorem claims that the transform of a minimal contractor (central symmetry, some axial symmetries, homothetie) is also minimal. As such it is possible to construct a complex minimal contractor from simple minimal contractors through transformations and unions. The different contributions to contractor theory are explained in chapter 4.

The final contribution is the application of the *image contractor* and *relaxed CSP* solving

techniques on a real case of localization of an underwater robot. A Particle Filtering method [Maurelli et al., 2008] has also been used on the same dataset. A comparison between set membership methods and Particle Filtering techniques is presented along with the application. The application is presented in chapter 6.

Being a robotician, I couldn't omit to also talk about the robots we worked on. The last chapters 7 and 8 presents respectively the autonomous underwater vehicles (AUV) developed in our school and the other robots we worked on such as autonomous surface vehicles (ASV) and autonomous ground vehicles (AGV).

Chapter 2

Solution characterization for relaxed constraint satisfaction problems

2.1 Introduction

This chapter presents several methods to represent the solution of a *relaxed constraint satisfaction problem* or *relaxed CSP* (See definition 2.2). A *relaxed CSP* can usually be represented by a set of equations (also called constraints) which involve an unknown variable to be determined. This variable belongs to a particular set called domain or search space. In this chapter we try to characterize the elements of the search space which satisfy only a part of the constraints. The first method introduced in [Jaulin, 2009] consists on searching for the set of elements satisfying at least a *specific number* of constraints. This method is explained in section 2.4.

The second method to deal with the *relaxed CSPs* can be seen as an extension of the previous method. The degree of satisfaction of an element of the search space is the number of constraints it satisfies. Instead on focusing on the set of elements having a particular degree of satisfaction, the idea is to consider all the sets for all degrees of satisfaction. The contribution here is to consider a polynomial with set valued coefficients which coefficients are those sets. The degree of each coefficient corresponds to the degree of satisfaction of the elements in that set coefficient. The choice of polynomial notation allow to take benefit from the polynomial arithmetics. The first benefit is a simple representation of the solution of the *relaxed CSP* in the form of a *product* of monomials. The second benefit is the possibility represent the solution of a *distributed relaxed CSP*. A *relaxed CSP* is distributed when the constraints are not immediately available at the same time and place. A lattice is a partially ordered set closed under least upper and greatest lower bounds (see [Davey and Priestley, 2002], for more details). The set of all subsets of $\mathbb{R}^n$ has a lattice structure. As such, the concept of *set polynomials* is generalized to *lattice polynomials*

where the coefficient belong to a lattice. The *lattice polynomials* are explained in section 2.5.

This chapter introduces another representation of the solution of a *relaxed CSP* in the form of a function called *accumulator*. For each element, the *accumulator* function returns the degree of satisfaction of that element (number of constraints it satisfies). The characteristic function of a set is the function returning 1 for elements belonging to that set and 0 otherwise. The *accumulator* provides a simple representation of the solution of the *relaxed CSP* is the form of a *sum* of the characteristic functions of the set associated to the constraints in the *relaxed CSP*. The *accumulators* can also be used to represent the solution of *distributed relaxed CSPs*. The *accumulator* theory is close to fuzzy logic [Klir and Yuan, 1995] [Nguyen and Walker, 2005] [Zadeh, 1965] and the generalized Hough transform [Bovik, 2000]. This link is also explained in this chapter. The accumulators are explained in Section 2.6.

2.2 Relaxed constraint satisfaction problem

In this section, the *relaxed constraint satisfaction problem* or *relaxed CSP* as well as the classical CSP are defined. The relaxed CSP's domains (see definitions below) are represented using $\mathbb{R}^n$ boxes. The boxes are used only for that purpose in this chapter. The boxes are redefined in chapter 3 in the context of interval analysis which is used to implement *relaxed CSP* solvers.

2.2.1 Intervals and boxes

Interval : An interval is a connected and closed subset of $\mathbb{R}$. If x is a real variable we denote by $[x]$ the interval containing this variable. $[x]$ is called the domain of x. An interval has an upper and lower bound which we will note as follows $[x] = [x^-, x^+]$. $\mathbb{IR}$ is the set of all the real intervals. $\mathbb{IN}$ is the set of natural number intervals. $w([x]) = x^+ - x^-$ is called width of $[x]$.

Example 1 $\emptyset, \{-1\}, [-1, 1], [-1, \infty], \mathbb{R}$ *are intervals.*

Box : A box of $\mathbb{R}^n$ is defined by a Cartesian product of intervals. A box can be also considered as an interval vector. If $\mathbf{x} = (x_1, .., x_n) \in \mathbb{R}^n$ is a real variable vector we denote by $[\mathbf{x}] = ([x_1], .., [x_n])$ the box containing this variable.

Example 2 $[1, 3] \times [2, 4]$ *is a box of* $\mathbb{R}^2$.

2.2.2 CSP and relaxed CSP

A constraint satisfaction problem (or CSP) is defined by a set of constraints $C_1, .., C_n$, a vector of variables $\mathbf{x} = (x_1, .., x_m)$ and the domain (or search space) $\mathbf{D} = D_1 \times .. \times D_m$ of possible values of $\mathbf{x}$. Originally the CSPs were defined on discrete domains [Clowes, 1971] [Waltz, 1975]. Later, CSPs were extended to continuous domains [Cleary, 1987] [Davis, 1987] [Hyvönen, 1992] [Sam-Haroud, 1995]. During the PhD thesis, we only considered continuous CSP where the domain $\mathbf{D}$ is a box of $\mathbb{R}^m$. Since the domain $\mathbf{D}$ is a box we changed its notation to $[\mathbf{x}] = ([x_1], .., [x_n])$.

The constraints are linear or nonlinear equations or inequalities

$$\begin{array}{l} g_i : \mathbb{R}^m \rightarrow \mathbb{R}, h_i : \mathbb{R}^m \rightarrow \mathbb{R} \\ C_i : g_i(\mathbf{x}) \leq 0, i = 1, .., k \\ C_i : h_i(\mathbf{x}) = 0, i = k+1, .., n. \\ \mathbf{x} \in [\mathbf{x}], \end{array} \tag{2.1}$$

A more general notation can be used to represent such a CSP

$$\begin{array}{l} \mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ \mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{array} \tag{2.2}$$

where $[y_i]$ are known real number boxes.

The CSP can also be denoted in an even more compact form

$$\begin{array}{l} \mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n \\ \mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}], [\mathbf{y}] \in \mathbb{IR}^n. \end{array} \tag{2.3}$$

where $[\mathbf{y}]$ is a known real number box.

All those notations are equivalent and are used in the manuscript depending on the situation. Searching for the solution of a CSP is to search for the set of elements $\mathbf{x} \in [\mathbf{x}]$ which satisfy all the constraints. In some cases there are no such elements. A CSP where not all the constraints are necessarily satisfied is called a *Relaxed Constraint Satisfaction Problem* or *Relaxed CSP*.

2.3 Extended summary

This section explains the different approaches to represent the solution of a relaxed CSP in a shortened form highlighting the most important results. **Reading this section is not necessary to understand the latter part of the manuscript**. The purpose is

for the reader to be able to see the whole theory in a condensed form without being lost in the details. Each time a result is presented, a reference to the section and subsection where the result is explained in more details is provided. Consider the following relaxed CSP

$$\begin{aligned} &\mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ &C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ &\mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{aligned} \tag{2.4}$$

In order to solve this problem, the first method introduced in [Jaulin, 2009] consists on searching for the set $\mathbb{S}_q$ of elements satisfying all of the constraints $\{C_1, .., C_n\}$ relaxing $q \in \{1, .., n\}$ of them. This means that up to q constraints can be not satisfied or inconsistent. This also means that $\mathbb{S}_q$ is also the set of points which satisfy at least $n - q$ constraints. Note that there is an equivalence between the formulations "relaxing q constraints" and "satisfying $n-q$ constraints". The first one is usually used when the number of constraints to be relaxed (or inconsistent constraints) is low. Such is the case of the localization problem, which can be set into a relaxed CSP, where the inconsistent constraints are caused by outliers in sensor data with a ratio inferior to 30%. The set $\mathbb{S}_q$ is defined by

$$\mathbb{S}_q = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}. \tag{2.5}$$

Denote by $\mathbb{X}_i$ the set of elements satisfying the constraint C_i

$$\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i]\}, [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \tag{2.6}$$

The sets $\mathbb{X}_i$ play an important role in the characterization of the different solutions of the relaxed CSP. For example, the set of point satisfying all the constraints (the solution of the CSP) is the intersection of the $\mathbb{X}_i$ sets. As for the solution set $\mathbb{S}_q$ of the relaxed CSP assuming q inconsistent constraints is defined by the q-relaxed intersection of the $\mathbb{X}_i$ sets (See subsection 2.4.3). The q-relaxed intersection of the $\mathbb{X}_i$ denoted $\bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i$ is the set of point belonging to at least $n - q$ sets among $\mathbb{X}_i$ sets. As such

$$\mathbb{S}_q = \bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{x} \in \mathbb{X}_i\}. \tag{2.7}$$

In the case of localization of a robot, the real number of inconsistent constraints q_{real} is usually unknown and may vary with time. It is sometimes possible to assume a maximum number of inconsistent constraints $q_{\max}$. We consider $\mathbb{S}_{q_{\max}}$ being the solution of the problem. The solution set $\mathbb{S}_{q_{\max}}$ is a guaranteed solution *i.e.* the real position is certainly in $\mathbb{S}_{q_{\max}}$ as long as the real number of outliers q_{real} is lower than $q_{\max}$. This approach is explained in more detail in section 2.4. Note that, since $q_{\text{real}} < q_{\max}$, we have $\mathbb{S}_{q_{\text{real}}} \subset \mathbb{S}_{q_{\max}}$ which means that the solution set $\mathbb{S}_{q_{\max}}$ is overestimated.

One of the contributions to the PhD thesis was to try to find a new representation of the solution of a relaxed CSP and avoid such overestimation. The approach consists on considering all the solution sets $\mathbb{S}_q$ for all possible values of q. As such, before coming up with the polynomial representation, the first mathematical representation of the solution of a relaxed CSP was in the form of a vector of sets $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$. To characterize this vector we considered a transform denoted $\mathcal{T}$ which we called the *sort transform* such as the vector $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ is the transform of the vector of sets $(\mathbb{X}_1, .., \mathbb{X}_n)$ defined in 2.6. We have

$$(\mathbb{S}_{n-1}, .., \mathbb{S}_0) = \mathcal{T}\ (\mathbb{X}_1, .., \mathbb{X}_n). \tag{2.8}$$

The *sort transform* $\mathcal{T}$ is actually introduced in subsection 2.5.3. The name of the *sort transform* comes from the fact that the output vector of sets is sorted with descending order relatively to the $\subset$ order *i.e.* $\mathbb{S}_0 \subset ... \subset \mathbb{S}_{n-1}$. The *sort transform* provides a formula for each solution set $\mathbb{S}_q, q \in \{1, .., n\}$ which corresponds to the q-relaxed intersection of the $\mathbb{X}_i, i \in \{1, ., n\}$ sets.

The representation of the solution of a relaxed CSP in form of a vector of sets $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ may seem lacking since each element of the vector $\mathbb{S}_q$ is computed separately using the *sort transform* formula (q-relaxed intersection). Basically the *sort transform* doesn't provide a unique formulation involving all of the $\mathbb{X}_i, i \in \{1, ., n\}$ sets which allow to obtain the $\mathbb{S}_q, q \in \{0, .., n-1\}$ sets. The solution we propose is to use polynomial representation. The idea is to consider polynomials with set valued coefficients also called *set polynomials*. We consider a polynomial called *solution set polynomial* which coefficients are the elements of the vector $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ such as the degree of the coefficient corresponds to the number of constraints which are satisfied by the elements in that set coefficient. Denote by $X^*(s)$ this polynomial

$$X^*(s) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{2.9}$$

The purpose of using polynomial representation is to take advantage of the set polynomial arithmetics (sum, product) to represent the *sort transform* with a unique formulation. The product and sum of *set polynomials* is similar to the product and sum of real polynomials just that the product " $*$ " of two coefficients is replaced by their intersection " $\cap$ " and the sum " $+$ " of two coefficients is replaced by their union " $\cup$ ". Consider the polynomial

$$Y^*(s) = \prod_{k=1}^{n} (\mathbb{X}_k s + \mathbb{R}^m) \tag{2.10}$$

By expanding the polynomial $Y^*(s)$ we find that

$$\prod_{k=1}^{n} (\mathbb{X}_k s + \mathbb{R}^m) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{2.11}$$

As such, the *sort transform* $\mathcal{T}$ has an equivalent polynomial formulation considering a product of the set monomials $(\mathbb{X}_k s + \mathbb{R}^m), k \in \{1, .., n\}$ also called the *polynomial sort transform* of the $\mathbb{X}_i, i \in \{1, ., n\}$ sets.

Remark 1 *A possible interpretation of the monomials* $\mathbb{X}_i s + \mathbb{R}^m, i \in \{1, .., n\}$ *is to represent the binary information* "{belongs to $\mathbb{X}_i$, belongs to $\mathbb{R}^m$ (no information)}". *The polynomial variable* s *is associated to the information* "belongs to $\mathbb{X}_i$". *This means that considering a product of monomials (equation (2.10)), the more sets* $\mathbb{X}_i$ *an element* $\mathbf{x}$ *belongs to, the higher is the degree of the coefficient to which* $\mathbf{x}$ *will belong to in* $X^*(s)$.

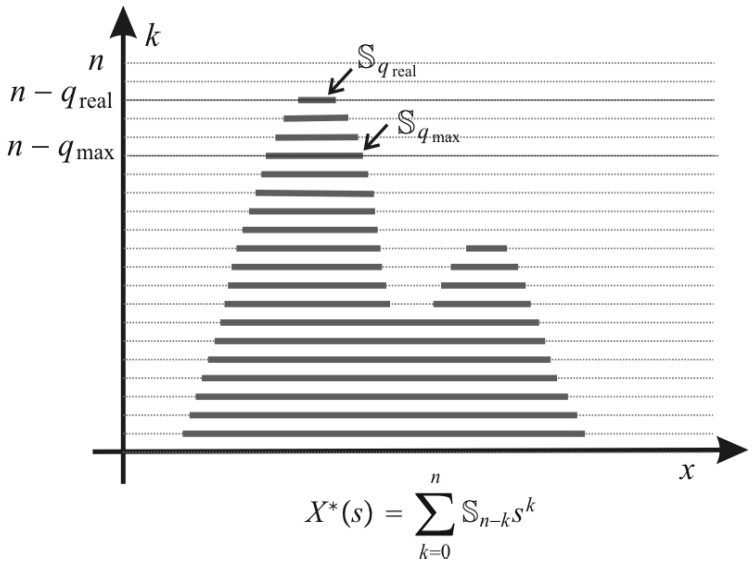

Figure 2.1: Example of solution representation of a one dimensionnal relaxed CSP

Figure 2.1, represents the solution sets $\mathbb{S}_q, q \in \{0, .., n-1\}$ for a one dimensional relaxed CSP with the unknown $x \in \mathbb{R}$. The x-axis corresponds to the unknown $x \in \mathbb{R}$ of the CSP. The solution sets $\mathbb{S}_q, q \in \{0, .., n-1\}$ are represented in bold gray in different levels represented by the k-axis. The k-axis corresponds to the number of constraints which are satisfied by elements in the set represented at that level. The set represented on level k (the set satisfying k constraints) is $\mathbb{S}_{n-k}$ (the solution set relaxing $n-k$ constraints). Denote by q_{real} the number of inconsistent constraints (which is unknown) and $q_{\max}$ the estimated maximal number of inconsistent constraints (used for the previous method). $\mathbb{S}_{q_{\text{real}}}$ and $\mathbb{S}_{q_{\max}}$ are represented on the Figure 2.1. Note that $\mathbb{S}_{q_{\text{real}}} \subset \mathbb{S}_{q_{\max}}$.

Another advantage of the polynomial notation is to represent the fusion of solutions of two independent relaxed CSPs sharing the same variable $\mathbf{x}$ using the product of their solution set polynomials. This property is useful for distributed computations. Considering two relaxed CSPs, the first one labeled $RCSP\_A$ which solution set polynomial is $A^*(s)$ and

the second one labeled $RCSP\_B$ which solution set polynomial is $B^*(s)$. Denote by $X^*(s)$ the solution of the relaxed CSP involving the constraints of both $RCSP\_A$ and $RCSP\_B$. We have

$$X^*(s) = A^*(s) * B^*(s). \tag{2.12}$$

The set of all subsets of $\mathbb{R}^n$ has a lattice structure [Davey and Priestley, 2002]. A lattice is a partially ordered set closed under least upper and greatest lower bounds. *Set polynomials* are generalized to *lattice polynomials* where the coefficient belong to a lattice. *Lattice polynomials* are introduced in subsection 2.5.4. We also generalize the set *sort transform* into the lattice *sort transform* defined in 2.5.3.

Section 2.6 introduces another possible representation of the solution of a relaxed CSP in the form of a function $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$ also called *accumulator* which for each element $\mathbf{x} \in [\mathbf{x}]$ associates the number of constraints it satisfies

$$\mathcal{A}(\mathbf{x}) = Card(\{i \in \{1, .., n\}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}). \tag{2.13}$$

The *accumulator* can also be defined using the characteristic function denoted χ of a constraint $C : \mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{f} : \mathbb{R}^m \to \mathbb{R}^p, [\mathbf{y}] \subset \mathbb{R}^p$ defined by

$$\begin{cases} \chi(C)(\mathbf{x}) = 1 \text{ if } \mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \\ \chi(C)(\mathbf{x}) = 0 \text{ otherwise.} \end{cases} \tag{2.14}$$

In fact, the *accumulator* $\mathcal{A}$ is the *sum* of the characteristic functions of all the constraints of the relaxed CSP

$$\mathcal{A}(\mathbf{x}) = \sum_{i \in \{1,..,n\}} \chi(C_i)(\mathbf{x}). \tag{2.15}$$

This last property justifies the name *"accumulator"*. Figure 2.2, represent an *accumulator* $\mathcal{A}$ as a solution representation of a one dimensional relaxed CSP with n constraints. The variable $x \in \mathbb{R}$ (one dimension) is represented on the x-axis and the k-axis corresponds to the number of constraints which are satisfied by x. Denote by q_{real} the number of constraints that are inconsistent (not satisfied) and $q_{\max}$ the estimated maximal number of inconsistent constraints. $\mathbb{S}_{q_{\text{real}}}$ and $\mathbb{S}_{q_{\max}}$ are represented on the Figure 2.2. Note that to obtain $\mathbb{S}_q, q \in \{1, .., n\}$ one have to solve the equation $\mathcal{A}(\mathbf{x}) \geq n - q$.

The advantage of this representation is an even simpler fusion of solutions of two relaxed CSPs sharing the same variable **x.** Considering two relaxed CSPs, the first one labeled $RCSP\_X$ which is associated to the accumulator $\mathcal{X}$ and the second one labeled $RCSP\_Y$ which is associated to the accumulator $\mathcal{Y}$. Denote by $\mathcal{A}$ the accumulator associated to the relaxed CSP involving the constraints of both $RCSP\_X$ and $RCSP\_Y$. We have

$$\mathcal{A} = \mathcal{X} + \mathcal{Y}. \tag{2.16}$$

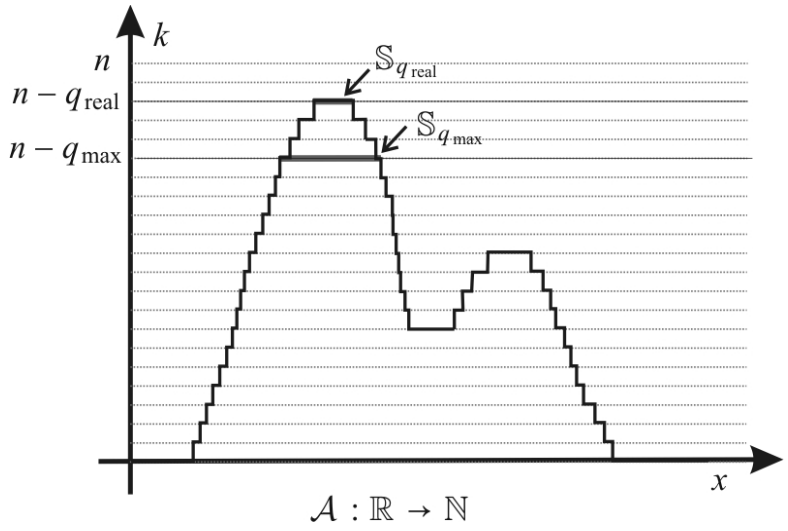

Figure 2.2: Using an accumulator to represent the solution of a one dimensionnal relaxed CSP

This property enables an even simpler distributed computations.

The *accumulator* theory is close to both fuzzy logic as well as generalized Hough Transform. This similarity is explained in section 2.6.

The computer implementation of *set polynomials* and *accumulators* based solvers are explained in a different chapter which is chapter 5. The purpose is to separate mathematical theory from the implementation which might not be unique.

2.4 Representing the solution of a relaxed CSP when the number of constraints to be relaxed is fixed

2.4.1 Introduction

The method explained in this section is used to represent a solution of a relaxed CSP defined in section 2.2. This method was actually introduced in [Jaulin, 2009]. In case of a relaxed CSP, it is supposed that not all of the constraints are necessarily satisfied. The idea is to seek the set of points $\mathbb{S}_q$ satisfying all the constraints but at most q of them. At most q constraints are thus allowed to be inconsistent with the other constraints. Before proposing a solution for the relaxed CSP, we explain in subsection 2.4.2 how to obtain the solution of a classical continuous CSP since both methods are similar. Subsection 2.4.3 defines a special set intersection called the q-relaxed intersection which is the set of points which belong to all the intersected sets but q of them. The q-relaxed intersection is then

used to define the solution of a relaxed CSP as seen in subsection 2.4.4. Subsection 2.4.5 give an application for boat localization using compass angle measurements of lighthouses to show the viability of the approach.

2.4.2 Solving a CSP

In this subsection we first try to characterize the solution of a CSP. This subsection helps understanding relaxed CSP resolution since both methods are similar. Consider the following CSP

$$\begin{aligned} &\mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ &C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ &\mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{aligned} \tag{2.17}$$

A solution of the CSP is an element $\mathbf{x}$ of the search space (domain) $[\mathbf{x}]$ which satisfies all the constraints simultaneously. Usually, we search for the set $\mathbb{S}$ of all the elements which are solution of the CSP

$$\mathbb{S} = \{\mathbf{x} \in [\mathbf{x}], \forall i \in \{1, .., n\}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}. \tag{2.18}$$

In a set membership context, the CSP can be formulated as a set inversion problem [Jaulin et al., 2001] where

$$\mathbb{S} = \mathbf{f}^{-1}([\mathbf{y}]), \tag{2.19}$$

where $[\mathbf{y}] = ([\mathbf{y}_1], .., [\mathbf{y}_n])$ and $\mathbf{f} = (\mathbf{f}_1, .., \mathbf{f}_n)$.

The solution of the CSP also can be characterized by an intersection of sets. Denote by $\mathbb{X}_i$ the set of points $\mathbf{x}$ which satisfy the constraint $C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i]$ *i.e.*

$$\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\} = \mathbf{f}_i^{-1}([\mathbf{y}_i]), i \in \{1, .., n\}. \tag{2.20}$$

The sets $\mathbb{X}_i$ play an important role in the characterization of the different solutions of the relaxed CSP. In the context of CSP resolution, we have

$$\mathbb{S} = \bigcap_{i \in \{1,..,n\}} \mathbb{X}_i. \tag{2.21}$$

Example 3 *Consider* $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ *and the following set of constraints involving* $\mathbf{x}$

$$\begin{aligned} &C_1 : x_1 - x_2 = 0 \\ &C_2 : x_1^2 + x_2^2 < 25 \\ &C_3 : x_1^2 + x_2^2 > 9 \\ &C_4 : x_1 * x_2 < 0 \end{aligned} \tag{2.22}$$

Those constraints can also be denoted

$$\begin{aligned} C_1 &: x_1 - x_2 \in \{0\} \\ C_2 &: x_1^2 + x_2^2 \in [0, 25] \\ C_3 &: x_1^2 + x_2^2 \in [9, \infty] \\ C_4 &: x_1 * x_2 \in [-\infty, 0] \end{aligned} \tag{2.23}$$

Figures 2.3.a, 2.3.b, 2.3.c, 2.3.d show the sets of the elements which satisfy the constraints C_1, C_2, C_3 and C_4 respectively. The solution set of the CSP defined by the constraints $\{C_1, C_2, C_3\}$ is shown in Figure 2.3.e. This solution set is the intersection of the solution sets represented in Figures 2.3.a, 2.3.b and 2.3.c. The CSP defined by the constraints $\{C_1, C_2, C_3, C_4\}$ doesn't admit any solution. In the latter case, it might be interesting to consider solving the relaxed CSP defined by the constraints $\{C_1, C_2, C_3, C_4\}$.

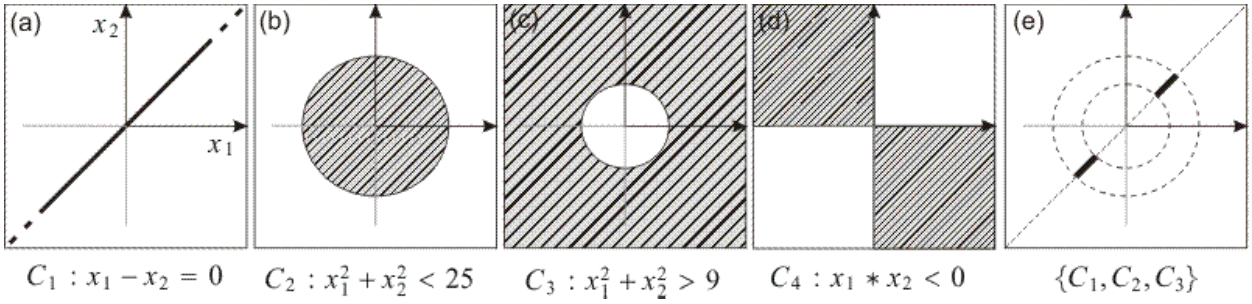

Figure 2.3: Solutions of the CSP for different sets of constraints

2.4.3 The q-relaxed intersection and union

The q-relaxed intersection

As seen in subsection 2.4.2, the solution of the CSP can be written as an intersection of the sets defined in (2.20). In case of a relaxed CSP, there is not necessarily any element in the search space satisfying all of the constraints. In the case of partial satisfaction, the intersection of the $\mathbb{X}_i$ sets defined in (2.20) yields an empty set. The idea is to define a special intersection called q-relaxed intersection which enables the characterization of the solution set of a relaxed CSP. The q-relaxed intersection is the set of points which belongs to all the intersected sets but at most q of them.

Definition 1 *The q-relaxed intersection (see [Jaulin, 2009]) of n sets $\mathbb{X}_1, .., \mathbb{X}_n$ also denoted by $\bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_n$ is the set of points $\mathbf{x} \in \mathbb{R}^m$ belonging to all the sets $\mathbb{X}_1, .., \mathbb{X}_n$ but at*

most q of them which is also defined by

$$\bigcap_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \{\mathbf{x}\in\mathbb{R}^m, \exists \mathbb{K}\subset\{1,..,n\}, card(\mathbb{K}) = n-q, \forall i\in\mathbb{K}, \mathbf{x}\in\mathbb{X}_i\}. \tag{2.24}$$

It is possible to rewrite the definition of the q-relaxed intersection using intersections and unions of the input sets $\mathbb{X}_i, i\in\{1..n\}$. This new formulation also called the disjunctive form of the relaxed intersection is used for demonstrations in section 2.5.

Definition 2 *The q-intersection of $\{\mathbb{X}_1,..,\mathbb{X}_n\}$ is the set of all elements (union) which belong to an intersection of a subset of $\{\mathbb{X}_1,..,\mathbb{X}_n\}$ of $n-q$ elements.*

$$\forall q\in\{1,..,n\}, \bigcap_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \bigcup_{\mathbb{K}\subset\{1,..,n\}, card(\mathbb{K})=n-q} \left(\bigcap_{k\in\mathbb{K}} \mathbb{X}_k\right). \tag{2.25}$$

Example 4 *Figure 2.4 illustrates the q-relaxed intersection of sets $\mathbb{X}_1,\mathbb{X}_2,\mathbb{X}_3,\mathbb{X}_4$ and $\mathbb{X}_5$. We have*

$$\begin{aligned}
&\bigcap_{i\in\{0,..,5\}}^{\{0\}} \mathbb{X}_i = \bigcap_{i\in\{1,..,5\}} \mathbb{X}_i = \emptyset\\
&\bigcap_{i\in\{0,..,5\}}^{\{2\}} \mathbb{X}_i = \mathbb{X}_3\cap\mathbb{X}_4\cap\mathbb{X}_5 \text{ (hatched set in Fig. 2.4)}\\
&\bigcap_{i\in\{0,..,5\}}^{\{5\}} \mathbb{X}_i = \bigcup_{i\in\{1,..,5\}} \mathbb{X}_i.
\end{aligned} \tag{2.26}$$

The q-relaxed union

The q relaxed intersection of n sets defines the set of points belonging to all the sets but at most q of them. At the same time, it also defines the set of points belonging to at least $n-q$ sets. In this subsubsection we define the q relaxed union of n sets, denoted by $\bigcup^{\{q\}}$, which defines the set of points belonging to at least q sets. See the mathematical definition below

$$\bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \{\mathbf{x}\in\mathbb{R}^m, \exists \mathbb{K}\subset\{1,..,n\}, card(\mathbb{K}) = q, \forall i\in\mathbb{K}, \mathbf{x}\in\mathbb{X}_i\}. \tag{2.27}$$

The relaxed union can also be rewritten using the disjunctive form

$$\forall q\in\{1,..,n\}, \bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \bigcup_{\mathbb{K}\subset\{1,..,n\}, card(\mathbb{K})=q} \left(\bigcap_{k\in\mathbb{K}} \mathbb{X}_k\right) \tag{2.28}$$

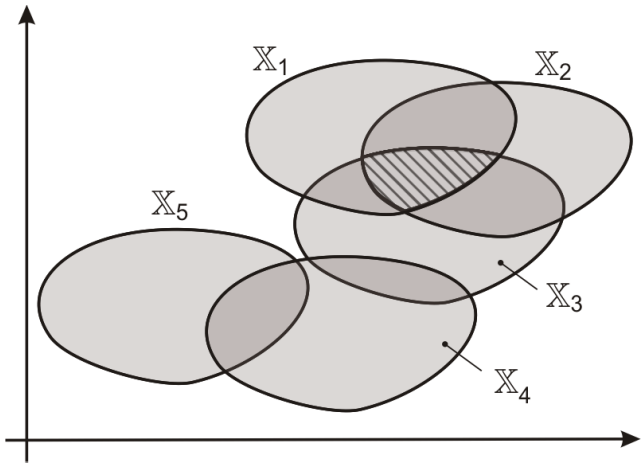

Figure 2.4: Illustration of the q-relaxed intersection of 5 sets. The hatched set corresponds to the 2-relaxed intersection.

Note that

$$\forall q \in \{1,..,n\}, \bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \bigcap_{i\in\{1,..,n\}}^{\{n-q\}} \mathbb{X}_i \tag{2.29}$$

The relaxed union and the relaxed intersection can be used in an equivalent manner. However, the concept of the relaxed intersection was invented before the concept of the relaxed union. Except the following theorem, rather than the relaxed union, we will use the relaxed intersection.

The q-relaxed union of unions of disjoint sets

Here is described a property of the q-relaxed union which is useful to handle sets composed of unions of disjoint sets. An example of such sets are periodic sets such as sets used to represent angles (see *sectors* in [Ramdani, 2005] for a possible representation of such complex sets). The set representing an angle is periodic since two real numbers with different values can represent the same angle if their value modulo 2π is the same. It is possible to prove that any interval representing an angle can be represented as a union of two disjoint intervals over the interval $[0, 2\pi]$. The following theorem is a small contribution to this thesis.

Theorem 1 *The q relaxed union of n sets composed of a finite union of disjoint subsets is the q relaxed union of all the subsets taken separately.*

proof. Consider n sets denoted $\mathbb{X}_i \subset \mathbb{R}^m$. For each set $\mathbb{X}_i$ it is possible to consider a partition denoted $\{\mathbb{X}_{i1}, .., \mathbb{X}_{ip_i}\}$. To simplify the proof consider

$$\forall i \in \{1, .., n\}, \exists m \in \mathbb{N}, p_i = m.$$

We prove that

$$\bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i = \bigcup_{i\in\{1,..,n\},j\in\{1,..,m\}}^{\{q\}} \mathbb{X}_{ij} \tag{2.30}$$

Considering the definition 2.27 of the q-relaxed union of the $\mathbb{X}_i$ sets and by replacing $\mathbb{X}_i$ by the union of the sets of its partition

$$\begin{aligned} \bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i &= \bigcup_{i\in\{1,..,n\}}^{\{q\}} \left(\bigcup_{j\in\{1,..,m\}} \mathbb{X}_{ij} \right) \\ &= \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = q, \forall i \in \mathbb{K}, \mathbf{x} \in \bigcup_{j\in\{1,..,m\}} \mathbb{X}_{ij}\}. \end{aligned} \tag{2.31}$$

Given $i \in \{1, .., n\}$, the sets $\mathbb{X}_{ij}, j \in \{1, .., m\}$ are disjoint as such

$$\begin{aligned} \bigcup_{i\in\{1,..,n\}}^{\{q\}} \mathbb{X}_i &= \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = q, \forall i \in \mathbb{K}, \exists! j \in \{1, .., m\}, \mathbf{x} \in \mathbb{X}_{ij}\} \\ &= \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\} \times \{1, .., m\}, card(\mathbb{K}) = q, \forall (i, j) \in \mathbb{K}, \mathbf{x} \in \mathbb{X}_{ij}\}, \\ &\stackrel{def}{=} \bigcup_{i\in\{1,..,n\},j\in\{1,..,m\}}^{\{q\}} \mathbb{X}_{ij} \end{aligned} \tag{2.32}$$

■

Remark 2 *The theorem is valid for sets which can be represented as a union of disjoint boxes or intervals. As a consequence, this theorem simplifies the implementation of algorithms for the relaxed union/intersection for those kind of sets.*

2.4.4 Solving a relaxed CSP using q-relaxed intersection

The example 3 in subsection 2.4.2 showed that sometimes a CSP doesn't admit solution which means there is no element in the search space satisfying all the constraints. The idea which has been introduced in [Jaulin, 2009] is to search for the set of elements satisfying all the constraints but at most q of them

$$\mathbb{S}_q = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}. \tag{2.33}$$

In the subsection 2.4.2, we saw that the solution set $\mathbb{S}$ of a CSP can be characterized by the intersection of the $\mathbb{X}_i$ sets defined in 2.20. When the CSP doesn't admit any solution the intersection of the $\mathbb{X}_i$ sets is empty. In the case of a relaxed CSP, it is also possible to express $\mathbb{S}_q$ using the $\mathbb{X}_i$ sets

$$\mathbb{S}_q = \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{x} \in \mathbb{X}_i\}, \tag{2.34}$$

which leads to the following equation using the q-relaxed intersection (see definition 1)

$$\mathbb{S}_q = \bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i.$$

The number of constraints to be relaxed q, which corresponds to the number of inconsistent or not satisfied constraints, is usually unknown but quite often q is bounded and never go beyond a maximum number $q_{\max}$. The solution of the relaxed CSP becomes

$$\mathbb{S}_{q_{\max}} = \bigcap_{i \in \{1,..,n\}}^{\{q_{\max}\}} \mathbb{X}_i. \tag{2.35}$$

Remark 3 *This solution is guaranteed as long as the assumption about the number of outliers ($q < q_{max}$) is respected.*

2.4.5 Example of localization with outliers

To show the motivation of solving the problem of relaxed CSP, this subsection presents a simple example of the localization of a boat using compass measurements of landmarks such as lighthouses. In Figure 2.5 the captain of a boat is trying to find boats position (x, y) by measuring the angle with respect to the magnetic north of four lighthouses A, B, C and D. Each lighthouse i has a known position (x_i, y_i) and the measured angle is θ_i.

Unexpectedly, the lighthouse D has been moved from its old position D^*. Since the captain is using an old map, he has mistaken the position of the lighthouse D thus creating an outlier in the measurements. The position of the boat is constrained by the following equations

$$C_i : (x_i - x) * \sin(\theta_i) - (y_i - y) * \cos(\theta_i) = 0, i \in \{A, B, C, D\} \tag{2.36}$$

For each constraint one can compute the set of compatible positions of the boat. If θ_i is measured precisely, this set of points is the line passing by the lighthouse and making an angle θ_i with the south-north axis. Those lines are represented in Figure 2.5. If all the

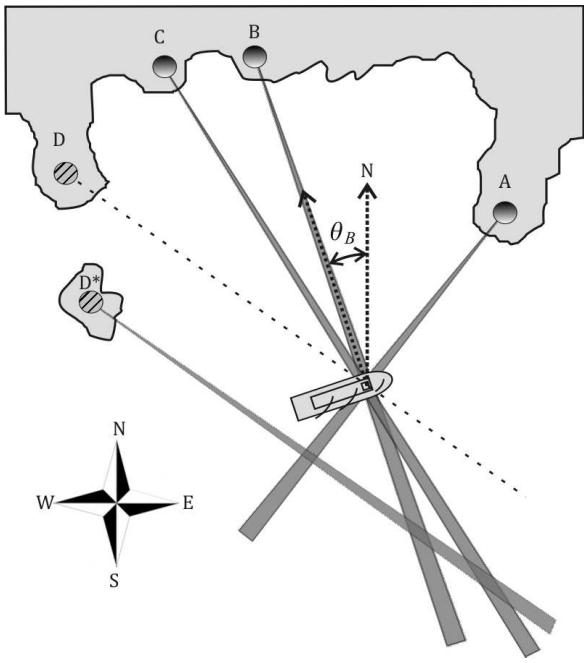

Figure 2.5: Localization of a boat using compass measurements of landmarks

measurements were correct, the position of the boat would be the intersection of those lines. In case of an outlier, it is not possible to obtain the position since those lines doesn't intersect in one point as seen in Figure 2.5. The right position is obtained by seeking the intersection of 3 lines among the 4 lines obtained by the measurements *i.e.* the 1-relaxed intersection of those lines.

2.5 Representing the solution of a relaxed CSP using set polynomials

2.5.1 Introduction

The previous section presented how to compute the solution set $\mathbb{S}_{q_{\max}}$ of a relaxed CSP assuming that the number of inconsistent constraints is always inferior to $q_{\max}$. In the context of localization, inconsistent constraints comes from outliers in sensor data. It is not easy to know exactly the number of those outliers which may also vary with time. As such, the estimated maximum of inconsistent constraints (in the localization case outliers in the data) $q_{\max}$ may be easily overestimated. In this case, the solution set $\mathbb{S}_{q_{\max}}$ is also overestimated. One of the contributions to the PhD thesis was to try to find a new representation of the solution of a relaxed CSP and avoid such overestimation. The

approach consists on considering all the solution sets $\mathbb{S}_q$ for all possible values of q *i.e.* the vector of sets $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$.

The first approach is to consider that this vector of sets is the result of a transform called the *sort transform* of the vector of sets of elements satisfying one constraint from the relaxed CSP at a time (the $\mathbb{X}_i$ sets defined in (2.20)). The *sort transform* provides a formula for each of the solution sets $\mathbb{S}_q, q \in \{0, .., n-1\}$ but doesn't provide a unique formulation of the solution as a whole.

The idea is to consider polynomials with set valued coefficients also called *set polynomials*. We consider a polynomial called *solution set polynomial* which coefficients are the elements of the vector $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ such as the degree of the coefficient corresponds to the number of constraints which are satisfied by the elements in that set coefficient. The purpose of using polynomial representation is to take advantage of the set polynomial arithmetics (sum, product) to represent the *sort transform* with a unique formulation as a product of monomials.

Because the set of all subsets of $\mathbb{R}^m$ has lattice structure, all the theory is generalized to lattices. Lattices are defined in subsection 2.5.2. The *sort transform* of vectors defined on lattices is defined in subsection 2.5.3. *Lattice polynomials i.e.* polynomials with lattice valued coefficients are defined in subsection 2.5.4. The use of *set polynomials* to represent the solution of a relaxed CSP is defined in subsection 2.5.5. Another advantage of the polynomial representation is to represent the possibility of distributed computations of a relaxed CSP. This concept is presented in subsection 2.5.6. *Set polynomials* based solvers are implemented in section 5.1 of chapter 5.

2.5.2 Lattices

A lattice $(\mathbb{E}, \leq)$ is a partially ordered set closed under least upper and greatest lower bounds (see [Davey and Priestley, 2002], for more details). The least upper bound (or supremum) of x and y is called the join and is denoted by $x \vee y$. The greatest lower bound (or infimum) is called the meet and is written as $x \wedge y$. A bounded lattice has a maximum (or top) and minimum (or bottom) element which is denoted $\top$ and $\bot$ respectively.

Example 5 *The set $\mathcal{P}(\mathbb{R}^m)$ of all subsets of $\mathbb{R}^m$ is a lattice with respect to the inclusion $\subset$.*

For two elements $A, B \in \mathcal{P}(\mathbb{R}^m)$ we have

$$\begin{aligned} A \wedge B &= A \cap B \\ A \vee B &= A \cup B \end{aligned}$$

$\mathcal{P}(\mathbb{R}^m)$ maximum and minimum are $\mathbb{R}^m$ and $\emptyset$ respectively.

Example 6 *Figure 2.6 shows an example of a Hasse diagram which is used to represent finite lattices. The elements of the lattice are the vertices of the diagram. The order of two elements is defined by their respective height in the diagram.*

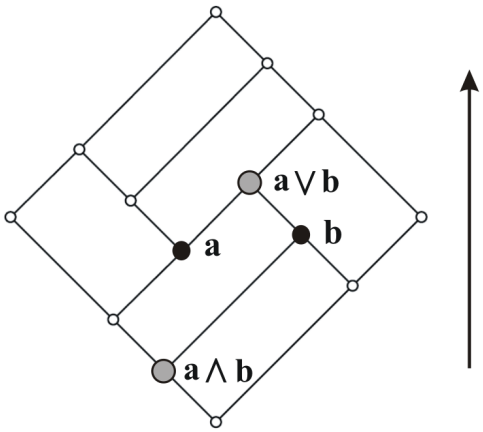

Figure 2.6: Finite lattice example

Consider a lattice $(\mathbb{E}, \leq)$.

Definition 3 *A vector* $\mathbf{y} \in \mathbb{E}^n$ *is sorted in descending order if*

$$\forall i \in \{1, .., n-1\}, y_{i+1} \leq y_i. \tag{2.37}$$

2.5.3 Sort transform

This subsection defines the *sort transform* which sorts a vector of elements in descending order (see 2.37). The *sort transform* of lattice elements is inspired from the *sort transform* of elements of a totally ordered set (such as the set of real numbers $\mathbb{R}$). Consider a totally ordered set $(\mathbb{E}, \leq)$. Denote by $\mathcal{T} : \mathbb{E}^n \to \mathbb{E}^n$ the *sort transform.*

Proposition 2 *Consider a vector* $\mathbf{x} \in \mathbb{E}^n$*. Denote by* $\mathbf{y} \in \mathbb{E}^n$ *the sorted vector in descending order of* $\mathbf{x}$ *i.e.* $\mathbf{y} = \mathcal{T}(\mathbf{x})$*. The* i^{th} *element of the sorted vector* $\mathbf{y} \in \mathbb{E}^n$ *is the smallest element among all the maxima of any* i *element sub-vector of* $\mathbf{x}$*. This can be expressed by the following formula*

$$\begin{aligned} \forall i &\in \{1, .., n\}, \\ y_i &= \bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i} \left(\bigwedge_{k \in \mathbb{K}} x_k \right). \end{aligned} \tag{2.38}$$

Example 7 *Consider the set of real numbers ordered by the usual* $\leq$ *relation. For two elements* $x, y \in \mathbb{R}$ *we have*

$$\begin{aligned} x \wedge y &= \min(x, y) \\ x \vee y &= \max(x, y) \end{aligned} \tag{2.39}$$

We have

$$\begin{aligned} \mathcal{T}\ (3,5,2) &= (3 \vee 5 \vee 2, (3 \vee 5) \wedge (3 \vee 2) \wedge (5 \vee 2), 3 \wedge 5 \wedge 2) \\ &= (5,3,2). \end{aligned} \tag{2.40}$$

The sort transform $\mathcal{T}$ can be extended to cases when order is not necessarily total. The sorted vector can be obtained using the same formula (2.38). Consider a lattice $(\mathbb{E}, \leq)$.

Proposition 3 *Consider* $\mathbf{x} \in \mathbb{E}^n$ *and* $\mathbf{y} \in \mathbb{E}^n$ *such as* $\mathbf{y} = \mathcal{T}\ (\mathbf{x})$. *The vector* $\mathbf{y}$ *is sorted in descending order (see 2.37).*

proof. For $i \in \{1, .., n-1\}$ we prove $y_{i+1} \leq y_i$

We have

$$\forall \mathbb{I}_{i+1} \subset \{1, .., n\}, Card(\mathbb{I}_{i+1}) = i+1, \exists \mathbb{I}_i \subset \{1, .., n\}, Card(\mathbb{I}_i) = i, \mathbb{I}_i \subset \mathbb{I}_{i+1}.$$

We have

$$\bigwedge_{k \in \mathbb{I}_{i+1}} x_k \leq \bigwedge_{k \in \mathbb{I}_i} x_k \tag{2.41}$$

On the other hand

$$\bigwedge_{k \in \mathbb{I}_i} x_k \leq \bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i} \left(\bigwedge_{k \in \mathbb{K}} x_k \right) \tag{2.42}$$

As a consequence

$$\forall \mathbb{I}_{i+1} \subset \{1, .., n\}, Card(\mathbb{I}_{i+1}) = i+1, \bigwedge_{k \in \mathbb{I}_{i+1}} x_k \leq \bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i} \left(\bigwedge_{k \in \mathbb{K}} x_k \right) \tag{2.43}$$

Thus

$$\bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i+1} \left(\bigwedge_{k \in \mathbb{K}} x_k \right) \leq \bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i} \left(\bigwedge_{k \in \mathbb{K}} x_k \right) \tag{2.44}$$

■

Remark 4 *In the case of elements defined on a lattice, the elements of the sorted vector* $\mathbf{y}$ *and input vector* $\mathbf{x}$ *do not necessarily remain the same*

Example 8 *Consider a finite lattice* $(\mathbb{E}, \leq)$ *which Hasse diagram is represented in Figure 2.7. The Figures 2.7.a and 2.7.b shows a vector* $(x_1, x_2, x_3, x_4) \in \mathbb{E}$ *(black dots) and the corresponding sorted vector* $(y_1, y_2, y_3, y_4) = \mathcal{T}((x_1, x_2, x_3, x_4))$*.(gray circles)*

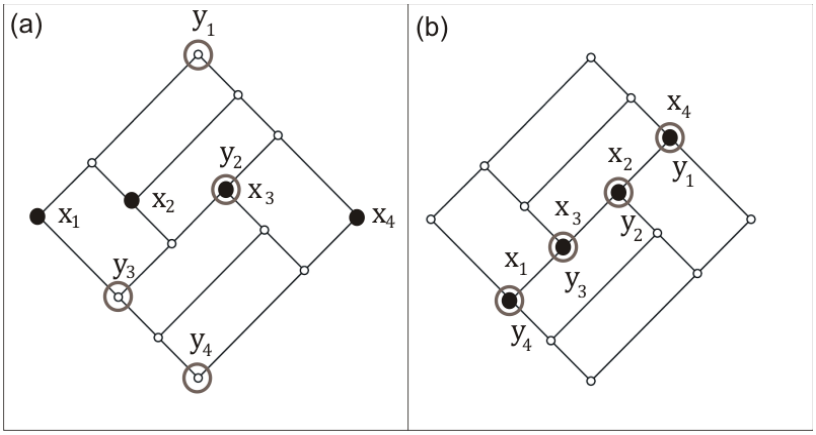

Figure 2.7: Applying the sort function to 4 points

In case of the lattice $(\mathcal{P}(\mathbb{R}^m), \subset)$ of sets ordered by the $\subset$ relation, the transform $\mathcal{T}$ is defined as follows

$$\mathcal{T} : (\mathcal{P}(\mathbb{R}^m))^n \rightarrow (\mathcal{P}(\mathbb{R}^m))^n$$
$$(\mathbb{X}_1, .., \mathbb{X}_n) \mapsto (\mathbb{Y}_1, .., \mathbb{Y}_n)$$

where

$$\forall i \in \{1, .., n\}, \tag{2.45}$$
$$\mathbb{Y}_i = \bigcup_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K})=i} \left(\bigcap_{k \in \mathbb{K}} \mathbb{X}_k \right) = \bigcap_{k \in \{1,..,n)}^{\{n-i\}} \mathbb{X}_k \text{ (see definition 2)}$$

where $\bigcap_{i \in \{1,..,n)}^{\{q\}} \mathbb{X}_i$ is the q-relaxed intersection defined in section 2.4.3.

Example 9 *Consider the lattice* $(\mathcal{P}(\mathbb{R}^2), \subset)$ *of sets ordered by the* $\subset$ *relation and a vector of 6 sets* $(\mathbb{X}_1, .., \mathbb{X}_6) \subset \mathbb{R}^2 \times ... \times \mathbb{R}^2$ *represented in Figure 2.8.a. Denote by* $(\mathbb{Y}_1, ..\mathbb{Y}_6) \subset \mathbb{R}^2 \times ... \times \mathbb{R}^2$ *the result of the sort transform of the vector* $(\mathbb{X}_1, ..\mathbb{X}_6)$*. The sorted vector is represented on Figure 2.8.b. Note that* $\mathbb{Y}_5 = \emptyset$ *and* $\mathbb{Y}_6 = \bigcap_{i \in \{1,..,6\}} \mathbb{X}_i = \emptyset$*. Later on in the manuscript the sorted vector is represented in a more compact form shown in Figure 2.9 where all the* $\mathbb{Y}_i$ *sets are superimposed.*

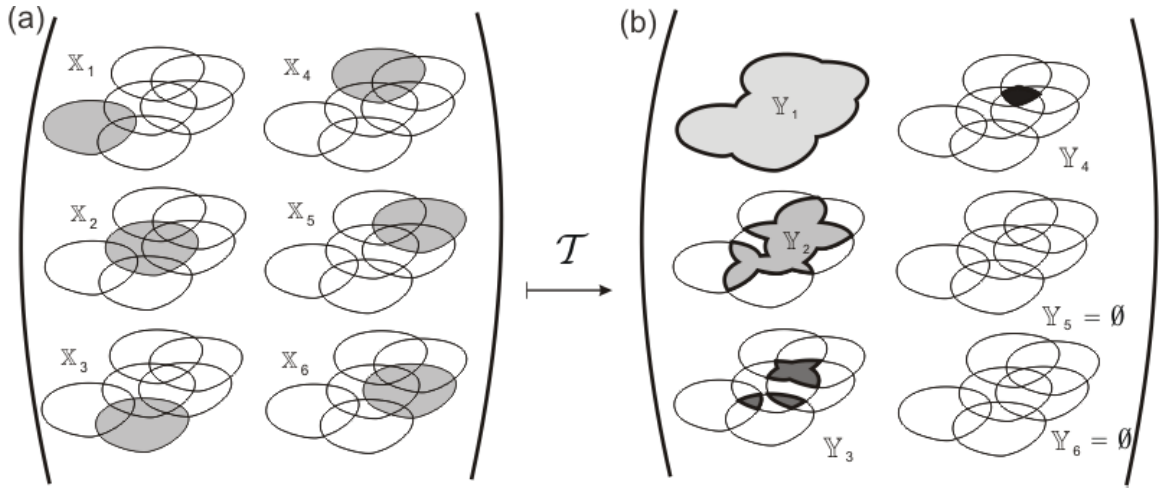

Figure 2.8: Illustration of the sort transform of six sets

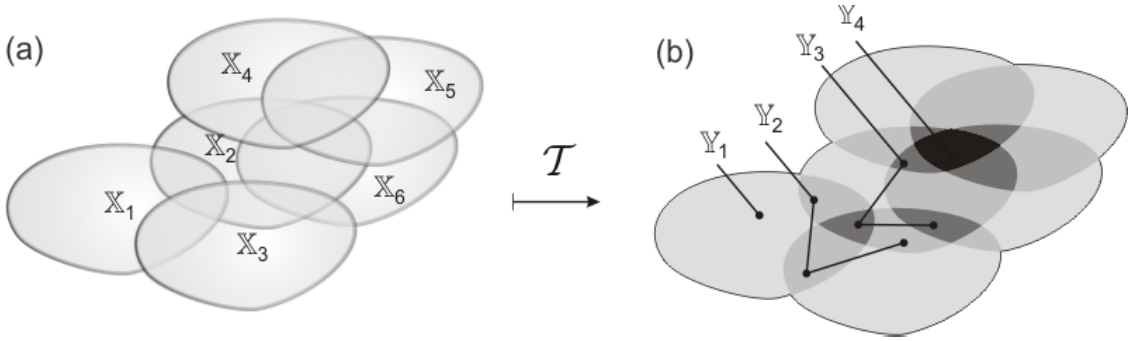

Figure 2.9: Compact illustration of the sort transform of 6 sets

Proposition 4 *The sort transform is idempotent*

$$\mathcal{T} \circ \mathcal{T}\ (\mathbf{x}) = \mathcal{T}\ (\mathbf{x}). \tag{2.46}$$

proof. Considering a vector $\mathbf{y} \in \mathbb{E}^n$ sorted in descending order, we prove that

$$\mathcal{T}\ (\mathbf{y}) = \mathbf{y}.$$

Consider $\mathbb{I} = \{1, .., i\}$. Since $\mathbf{y}$ is sorted

$$\bigwedge_{k \in \mathbb{I}} y_k = y_i. \tag{2.47}$$

We have

$$\forall \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = i, \mathbb{K} \neq \mathbb{I}, \exists j \in \mathbb{K}, j \in \{i+1, .., n\}, \tag{2.48}$$

since $\mathbf{y}$ is sorted and $i < j$

$$y_j \leq y_i. \tag{2.49}$$

On the other hand

$$\bigwedge_{k \in \mathbb{K}} y_k \leq y_j, \tag{2.50}$$

thus

$$\bigwedge_{k \in \mathbb{K}} y_k \leq y_i, \tag{2.51}$$

finally

$$y_i = \bigvee_{\mathbb{K} \subset \{1,..,n\}, card(\mathbb{K}) = i} \left(\bigwedge_{k \in \mathbb{K}} y_k \right). \tag{2.52}$$

■

2.5.4 Lattice polynomials

The previous subsection introduced the lattice *sort transform* which enables to sort elements of a lattice even tough the order is not total. The *sort transform* provides a formula (2.38) for each element of the sorted output vector function of the elements of the input vector. This subsection introduces polynomials with lattice valued coefficients also called *lattice polynomials*. In this subsection we want to show that polynomial representation is better suited to represent the *sort transform* in a more compact and unique formulation taking benefit from polynomial arithmetics. In this subsection, first, the sum and product of two polynomials are defined. Those two operations are necessary to represent the *sort transform* as using the product of specific monomials. The polynomials are used in our context to represent the solution of a relaxed CSP. In order to enable the exploitation of the solution, the image by a function of a lattice polynomial is defined. Finally, the last topic is about the algebraic structure of the set of lattice polynomials.

Definition 4 *A lattice polynomial of degree n is a polynomial with the unknown s where the coefficients* $\mathbf{x} = (x_0, .., x_n)$ *are defined on a lattice. The polynomial is denoted*

$$X(s) = \sum_{i=0}^{n} x_i s^i. \tag{2.53}$$

Denote by $\mathcal{LP}(\mathbb{E})$ *the set of lattice polynomials which coefficients are defined on the lattice* $\mathbb{E}$*.*

We decided a notation convention saying that $X(s)$ is the lattice polynomial associated to the vector of coefficients $\mathbf{x}$.

Example 10 *In the lattice of subsets of* $\mathbb{R}$, $(\mathcal{P}(\mathbb{R}), \subset)$ *consider the following polynomial*

$$X(s) = [-2, 3]s^3 + [1, \infty]s^2 + [2, 4]s + [5, 8]. \tag{2.54}$$

The product of two lattice polynomials is a lattice polynomial and follows the same rules as the product of real polynomials. The sum of coefficients " + " corresponds to " $\vee$ " operator and its product " $*$ " corresponds to the " $\wedge$ "operator.

Example 11 *Consider* $A(s)$ *and* $B(s)$ *two monomials*

$$\begin{aligned} (a_1 s + a_0) * (b_1 s + b_0) &= (a_1 * b_1)s^2 + (a_1 * b_0 + a_0 * b_1)s + a_0 * b_0 \\ &= (a_1 \wedge b_1)s^2 + (a_1 \wedge b_0) \vee (a_0 \wedge b_1)s + a_0 \wedge b_0. \end{aligned} \tag{2.55}$$

The sum of two lattice polynomials is a lattice polynomial and follows the same rules as the sum of real polynomials. The sum of coefficients " + " corresponds to " $\vee$ " operator.

Example 12 *Consider* $A(s)$ *and* $B(s)$ *two monomials*

$$\begin{aligned} (a_1 s + a_0) + (b_1 s + b_0) &= (a_1 + b_1)s + (a_0 + b_0) \\ &= (a_1 \vee b_1)s + a_0 \vee b_0. \end{aligned} \tag{2.56}$$

Definition 5 *A lattice polynomial* $X^*(s)$ *is* nested *if the coefficient vector* $\mathbf{x}^*$ *is sorted in descending order and* $x_0^* = \top$*. We also consider the polynomial* $\perp$ *as being* nested*. Denote by* $\overrightarrow{\mathcal{LP}}(\mathbb{E})$ *the set of nested polynomials which coefficients are defined on the lattice* $\mathbb{E}$*.*

Example 13 *In* $(\mathcal{P}(\mathbb{R}), \subset)$ *consider the following* nested *polynomial*

$$X^*(s) = [2, 3]s^3 + [1, 4]s^2 + [-1, \infty]s + \mathbb{R}. \tag{2.57}$$

Definition 6 *Consider* $\mathbf{x} \in \mathbb{E}^n$ *and the polynomial* $X^*(s)$ *defined by*

$$X^*(s) = \prod_{i=1}^{n}(x_i s + \top) = \sum_{i=1}^{n} x_i^* s^i + \top. \tag{2.58}$$

$X^*(s)$ *is called the* polynomial sort transform *of* $\mathbf{x}$.

Proposition 5 *The polynomial* $X^*(s)$ *is* nested.

proof. This proposition is proven using recursive approach. We have $\forall x \in \mathbb{E}, xs + \top$ is nested. For $n \in \mathbb{N}$, Suppose that $\forall \mathbf{x} \in \mathbb{E}^n, X^*(s)$ is *nested*. Consider $\mathbf{x} \in \mathbb{E}^{n+1}$. Denote by

$$Y(s) = \prod_{i=1}^{n}(x_i s + \top).$$

We have

$$X^*(s) = Y(s) * (x_{n+1} s + \top).$$

To simplify the demonstration $\forall i > n$ we assume that $y_i = \bot$ and $\forall i < 0$ we assume that $y_i = \top$. We have

$$\forall i \in \{1, ..n+1\} x_i^* = y_i \vee (y_{i-1} \wedge x_{n+1}).$$

Since $Y(s)$ is nested

$$\forall i \in \{1, ..n+1\}, y_i \leq y_{i-1},$$

thus

$$y_{i-1} \wedge x_{n+1} \leq y_{i-2} \wedge x_{n+1},$$

thus

$$y_i \vee (y_{i-1} \wedge x_{n+1}) \leq y_{i-1} \vee (y_{i-2} \wedge x_{n+1}) \text{ i.e. } x_i^* \leq x_{i-1}^*.$$

$X^*(s)$ is thus nested. ■

Proposition 6 *Denote by* $\mathbf{y} \in \mathbb{E}^n$ *the vector of coefficient of* $X^*(s)$ *which order is greater or equal than* 1 i.e. $\mathbf{y} = (x_1^*, .., x_n^*)$

We have

$$\mathbf{y} = \mathcal{T}\ (\mathbf{x}), \tag{2.59}$$

where $\mathcal{T}$ *is the* sort transform *defined in Section 2.5.3.*

proof. $X^*(s)$ is a product of monomials. Using the definition of the multiplication of real polynomials (replacing $\top$ by 1 in the case of multiplication since $\top$ is the neutral element for $\wedge$) it is possible to expand $X^*(s)$. We find that

$$x_i^* = \sum_{\mathbb{K}\subset\{1,..,n\},card(\mathbb{K})=i} \left(\prod_{k\in\mathbb{K}} x_k\right), i \in \{1,..,n\}. \tag{2.60}$$

By replacing the sum and product by $\vee$ and $\wedge$ respectively we find that

$$y_i = x_i^* = \bigvee_{\mathbb{K}\subset\{1,..,n\},card(\mathbb{K})=i} \left(\bigwedge_{k\in\mathbb{K}} x_k\right), i \in \{1,..,n\}, \tag{2.61}$$

which corresponds to the definition of the *sort transform* (see equation (2.38)). ■

This proves that the polynomial representation is well suited to represent the result as well as the nature of the *sort transform*. For each vector $\mathbf{x} \in \mathbb{E}^n$ is associated the *polynomial sort transform* $X^*(s) = \prod_{i=1}^{n}(x_i s + \top)$.

Proposition 7 *Consider* $\mathbf{a} \in \mathbb{E}^n$ *and* $\mathbf{b} \in \mathbb{E}^m$ *and* $\mathbf{c} \in \mathbb{E}^{m+n}$ *such as* $\mathbf{c} = (a_1, .., a_n, b_1, .., b_m)$.

Denote by $A^*(s), B^*(s)$ *and* $C^*(s)$ *their respective* polynomial sort transform

We have

$$C^*(s) = A^*(s) * B^*(s) \tag{2.62}$$

proof. Using the *polynomial sort transform* formula (see definition (6))

$$C^*(s) = \prod_{i=1}^{m+n}(c_i s + \top) = \prod_{i=1}^{n}(a_i s + \top) * \prod_{i=1}^{m}(b_i s + \top) = A^*(s) * B^*(s) \tag{2.63}$$

■

Remark 5 *This proposition is important computation wise. In fact, the computation of the* polynomial sort transform *can be split into multiple computations of smaller degree polynomials then merged into one polynomial through multiplication. This proposition can be useful in distributed computations of relaxed CSPs as shown in subsection 2.5.5.*

A *nested* polynomial actually represents a solution of a relaxed CSP as seen in subsection 2.5.5. It is sometimes useful to compute its image by a function to be able to exploit the solution. As an example, in the context of localization of a robot (which can be cast into a relaxed CSP), the calculated position of the robot can be represented using

a set polynomial. Suppose the robot measures a relative position of an object in the environment using a sensor such as a camera. It is possible to deduce the set polynomial corresponding to the absolute position of that object in the environment using translation function.

Definition 7 *The image by a function of a* $\mathbb{E}^n$ *lattice polynomial* $X(s)$ *by a function* $f : \mathbb{E} \to \mathbb{E}$ *is a lattice polynomial denoted* $f(X)(s)$ *defined by*

$$f(X)(s) = \sum_{i=0}^{n} f(x_i)s^i. \tag{2.64}$$

Proposition 8 *In case of the lattice of subsets of* $\mathbb{R}^m$, $(\mathcal{P}\ (\mathbb{R}^m), \subset)$, *the image by a function of a* nested *polynomial is also* nested

proof. Considering two sets $\mathbb{A}, \mathbb{B} \subset \mathbb{R}^m$ such as $\mathbb{A} \subset \mathbb{B}$ we have $f(\mathbb{A}) \subset f(\mathbb{B})$. ■

Finally, the last part of this subsection introduces the algebraic structures of the set of lattice polynomials and the set of nested lattice polynomials.

Theorem 9 *Consider a lattice* $\mathbb{E}$. *If* $(\mathbb{E}, +, *, \bot, \top)$ *is a dioïd then the set of lattice polynomials* $(\mathcal{LP}(\mathbb{E}), +, *, \bot, \top)$ *is also a dioïd.*

proof. Consider a dioïd $(\mathbb{E}, +, *, \bot, \top)$. Consider three polynomials $A, B, C \in \mathcal{LP}(\mathbb{E})$.

Part I: $(\mathcal{LP}(\mathbb{E}), +, \bot)$ is a **commutative monoïd** :

Closure: Denote by $X = A + B$

$$\forall k \in \mathbb{N}, x_k = a_k + b_k \Rightarrow \forall k \in \mathbb{N}, x_k \in \mathbb{E} \Rightarrow X \in \mathcal{LP}(\mathbb{E}). \tag{2.65}$$

Associativity: Denote by $X = (A + B) + C$ and $Y = A + (B + C)$

$$\forall k \in \mathbb{N}, x_k = (a_k + b_k) + c_k = a_k + (b_k + c_k) = y_k \Rightarrow X = Y. \tag{2.66}$$

Commutativity : Denote by $X = A + B$ and $Y = B + A$

$$\forall k \in \mathbb{N}, x_k = a_k + b_k = b_k + a_k = y_k \Rightarrow X = Y. \tag{2.67}$$

Identity element :

$$A + \bot = \bot + A = \bot + a_0 + a_1 s + ... = a_0 + a_1 s + ... = A. \tag{2.68}$$

Part II: $(\mathcal{LP}(\mathbb{E}), *, \top)$ is a **monoïd** :

Closure: Denote by $X = A * B$

$$\forall k \in \mathbb{N}, x_k = \sum_{i+j=k} a_i * b_j \Rightarrow \forall k \in \mathbb{N}, x_k \in \mathbb{E} \Rightarrow X \in \mathcal{LP}(\mathbb{E}). \tag{2.69}$$

Associativity: Denote by $X = (A * B) * C$ and $Y = A * (B * C)$

$\forall k \in \mathbb{N}$,

$$\begin{aligned} x_k &= \sum_{i+j=k} \left(\left(\sum_{p+q=i} a_p * b_q \right) * c_j \right) \\ &= \sum_{p+q+j=k} (a_p * b_q) * c_j \\ &= \sum_{i+p+q=k} a_i * (b_p * c_q) \\ &= \sum_{i+j=k} \left(a_i * \left(\sum_{p+q=j} b_p * c_q \right) \right) \\ &= y_k. \end{aligned} \tag{2.70}$$

Identity element :

$$\top * A = \top * a_0 + \top * a_1 s + ... = a_0 + a_1 s + ... = A. \tag{2.71}$$

$A * \top = A$ aswell.

Part III : **disributivity** of $*$ over $+$.

Denote by $X = A * (B + C)$ and $Y = A * B + A * C$

$$\begin{aligned} x_k &= \sum_{i+j=k} a_i * (b_j + c_j) \\ &= \sum_{i+j=k} (a_i * b_j + a_i * c_j) \\ &= \sum_{i+j=k} a_i * b_j + \sum_{i+j=k} a_i * c_j \\ &= y_k \end{aligned} \tag{2.72}$$

Similarily we prove

$$(A + B) * C = A * C + B * C$$

$\perp$ **annihilates** $\mathcal{LP}(\mathbb{E})$ with respect to multiplication

$$\perp * A = \perp * a_0 + \perp * a_1 s + ... = \perp + \perp s + ... = \perp. \tag{2.73}$$

$A * \perp = \perp$ aswell.

The **order** $\leq$ on $\mathcal{LP}(\mathbb{E})$: we choose the product order derived from the order $\leq$ of $\mathbb{E}$ ■

Example 14 $\forall m \in \mathbb{N}^*, (\mathcal{P}\ (\mathbb{R}^m), \cup, \cap, \emptyset, \mathbb{R}^m)$ *is a dioïd as such the set of polynomials with set valued coefficients is also a dioïd.*

The set of nested lattice polynomials $\overrightarrow{\mathcal{LP}}$ has an algebraic structure which we try to characterize below.

Corollary 10 *Any nested polynomial* $Y(s)$ *of the degree* n *is the polynomial sort transform of at least one vector* $\mathbf{x} \in \mathbb{E}^n$

proof. Denote by $\mathbf{x} \in \mathbb{E}^n$ the vector of coefficient of $Y(s)$ which order is greater or equal than 1. Since the transform is idempotent we have $\mathbf{x} = \mathcal{T}\ (\mathbf{x})$ i.e.

$$Y(s) = X^*(s). \tag{2.74}$$

■

Corollary 11 *The multiplication* " $*$ " *and sum* " $+$ " *are binary operations in the set of nested lattice polynomials* $\overrightarrow{\mathcal{LP}}$.

proof. Consider two nested polynomials $A(s)$ and $B(s)$.

The sum " $+$ " is a binary operation : Consider $C(s) = A(s) + B(s)$. We have $c_i = a_i \vee b_i$.

Since $a_{i+1} \leq a_i$ and $b_{i+1} \leq b_i$ then $a_{i+1} \vee b_{i+1} \leq a_i \vee b_i$ then $c_{i+1} \leq c_i$

Besides $c_0 = a_0 \vee b_0 = \top \vee \top = \top$.

As such $A(s) + B(s)$ is a nested polynomial

The product " $*$ " is a binary operation : Consider $D(s) = A(s) * B(s)$.

Based on corollary 10 we have $A(s) = \prod_{i=1}^{n}(a_i s + \top)$ and $B(s) = \prod_{i=1}^{n}(b_i s + \top)$.

Thus $D(s) = \prod_{i=1}^{n}(a_i s + \top) * \prod_{i=1}^{n}(b_i s + \top)$

The last proposition ensures that $D(s)$ is also nested. ■

Corollary 12 *Consider a lattice* $\mathbb{E}$. *If* $(\mathbb{E}, +, *, \bot, \top)$ *is a dioïd then the set of nested lattice polynomials* $(\overrightarrow{\mathcal{LP}}(\mathbb{E}), +, *, \bot, \top)$ *is a subdioïd of* $(\mathcal{LP}(\mathbb{E}), +, *, \bot, \top)$.

proof. The theorem 9 ensures that $(\mathcal{LP}(\mathbb{E}), +, *, \bot, \top)$ is a dioïd. Since $(+, *)$ are operations in $\overrightarrow{\mathcal{LP}}(\mathbb{E})$ (as shown in 11) then $(\overrightarrow{\mathcal{LP}}(\mathbb{E}), +, *, \bot, \top)$ is a subdioïd of $(\mathcal{LP}(\mathbb{E}), +, *, \bot, \top)$ (see [Hardouin, 2004]) ■

2.5.5 Using set polynomials to represent the solution of a relaxed CSP

Consider the following relaxed CSP

$$\begin{aligned} &\mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ &C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ &\mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{aligned} \tag{2.75}$$

Section 2.4 proposes a solution for the relaxed CSP when it is possible to assume the number q of inconsistent constraints (or outliers) in a relaxed CSP. The solution is the set $\mathbb{S}_q$ of elements satisfying all the constraints but at most q of them. When q is unknown and may vary with time, the idea is to consider all the possible sets $\mathbb{S}_q, q \in \{0, .., n-1\}$. Denote by $\mathbb{X}_i$ the set of points $\mathbf{x}$ which satisfy the constraint $C_i : f_i(\mathbf{x}) \in [y_i]$ i.e.

$$\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\} = \mathbf{f}_i^{-1}([\mathbf{y}_i]), i \in \{1, .., n\}. \tag{2.76}$$

The $\mathbb{X}_i$ sets are elementary in the definition of the solution of a relaxed CSP. As seen in subsection 2.4.4, $\mathbb{S}_q, q \in \{0, .., n-1\}$ can be obtained using the q-relaxed intersection of the $\mathbb{X}_i$ sets

$$\mathbb{S}_q = \bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i \tag{2.77}$$

As seen in the equation (2.45), the equation (2.77) corresponds to the expression of the *sort transform* of the $\mathbb{X}_i$ sets in the case of the lattice $(\mathcal{P}(\mathbb{R}^m), \subset)$. In other words, the solution of the relaxed CSP is the vector $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ which can be obtained by sorting the vector $(\mathbb{X}_1, .., \mathbb{X}_n)$ using the *sort transform* $\mathcal{T}$

$$(\mathbb{S}_{n-1}, .., \mathbb{S}_0) = \mathcal{T}(\mathbb{X}_1, .., \mathbb{X}_n) \tag{2.78}$$

As seen in subsection 2.5.4, the polynomial representation is better suited to represent the *sort transform*. The purpose of using polynomial representation is to take advantage of the set polynomial arithmetics (sum, product) to represent the *sort transform* with a unique formulation. The product and sum of *set polynomials* is similar to the product and sum of real polynomials just that the product " $*$ " of two coefficients is replaced by their intersection "$\cap$" and the sum "$+$" of two coefficients is replaced by their union "$\cup$". As such, the *sort transform* $\mathcal{T}$ has an equivalent polynomial formulation considering a product of the set monomials $(\mathbb{X}_k s + \mathbb{R}^m), k \in \{1, .., n\}$. This formulation (see definition 6) is called the *polynomial sort transform* of the $\mathbb{X}_i, i \in \{1, ., n\}$ sets. We obtain the following *set polynomial*

$$X^*(s) = \prod_{k=1}^{n} (\mathbb{X}_k s + \mathbb{R}^m) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{2.79}$$

The polynomial $X^*(s)$ is also called *solution set polynomial* of the relaxed CSP since its coefficients are the $\mathbb{S}_q, q \in \{0, .., n-1\}$ sets. The non empty set coefficient with the highest degree is the set of elements satisfying most of the constraints. This set, which is the smallest of all of them, can be considered as the best solution set of the relaxed CSP. On the other hand, keeping the whole polynomial can be justified in case of distributed computing as seen in the next subsection.

Remark 6 *A possible interpretation of the polynomials* $\mathbb{X}_i s + \mathbb{R}^m, i \in \{1, .., n\}$ *is to represent the binary information* "{belongs to $\mathbb{X}_i$, belongs to $\mathbb{R}^m$ (no information)}". *The polynomial variable* s *is associated to the information* "belongs to $\mathbb{X}_i$". *This means that considering a product of those polynomials (equation (2.79)), the more sets* $\mathbb{X}_i$ *an element* $\mathbf{x}$ *belongs to, the higher is the degree of the coefficient to which* $\mathbf{x}$ *belongs to in* $X^*(s)$.

2.5.6 Solving distributed relaxed CSPs

In robotics, the constraints generally come from sensor data. Sensor data are not always available at the same time or the same place (ex: in the case of robot swarm or in case of an evolving process). The purpose is to solve the relaxed CSP involving all of the constraints despite the fact that the constraints are distributed (not available immediately at the same place/time of computation). The idea is to consider several smaller relaxed CSPs formed by subsets of available constraints then merge the resulting solutions into one. The different solutions are represented using set polynomials and the merging is obtained using polynomial arithmetics. This allow distributed computations which are more adapted to distributed systems.

Consider two relaxed CSP labeled $RCSP\_A$ and $RCSP\_B$ involving the same variable $\mathbf{x} \in \mathbb{R}^m$ and constrained respectively with the set of constraints $\{C_1, .., C_r\}$ and $\{C_{r+1}, .., C_n\}$. Consider

$$\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\} = \mathbf{f}_i^{-1}([\mathbf{y}_i]), i \in \{1, .., n\}. \tag{2.80}$$

Denote by $X_A^*(s)$ and $X_B^*(s)$ the *solution set polynomials* of respectively the relaxed CSP $RCSP\_A$ and the relaxed CSP $RCSP\_B$

$$\begin{array}{l} X_A^*(s) = \prod_{i=1}^{r} (\mathbb{X}_i s + \top) \\ X_B^*(s) = \prod_{i=r+1}^{n} (\mathbb{X}_i s + \top). \end{array} \tag{2.81}$$

Denote by $X_S^*(s)$ the *solution set polynomials* of the relaxed CSP involving the variable $\mathbf{x} \in \mathbb{R}^m$ and constrained by all the constraints $C_i, i \in \{1, .., n\}$. If the constraints were

not distributed, $X_S^*(s)$ could be obtained using the *polynomial sort transform*

$$X_S^*(s) = \prod_{i=1}^{n} (\mathbb{X}_i s + \top), \tag{2.82}$$

but since the constraints are distributed, the polynomial $X_S^*(s)$ can be obtained by multiplying the polynomials $X_A^*(s)$ and $X_B^*(s)$ using proposition 7 without having to recompute the product of monomials from scratch

$$X_S^*(s) = X_A^*(s) * X_B^*(s) \tag{2.83}$$

2.5.7 Conclusion

In this chapter a new way to represent a solution of a relaxed CSP was presented. The idea is to use set polynomials which coefficients are the set of points satisfying the number of constraints corresponding to the degree of that coefficient. It is important to note that the polynomial notation is just a representation which help to understand constraint relaxation process. The data structures and algorithms used to implement and compute set polynomials are explained in section 5.1 of chapter 5.

2.6 Representing the solution of a relaxed CSP using accumulators

2.6.1 Introduction

This section presents an alternative possibility to represent the solution of a relaxed CSP using a function called *accumulator*. For each element $\mathbf{x}$ in the variable domain of the CSP, an *accumulator* function returns the number of constraints that the element $\mathbf{x}$ satisfies. The *accumulator* representation allows simpler distributed computations. Subsection 2.6.2 defines the *accumulator* and explains how it is used to represent the solution of a relaxed CSP. Subsection 2.6.4 talks about the connection between *accumulators* and fuzzy logic. Subsection 2.6.5 talks about the link between *accumulators* and generalized Hough Transform. The *accumulators* related data structures and algorithms are explained in section 5.2 of chapter 5.

2.6.2 The accumulator

An accumulator is an alternative possibility to represent the solution of a relaxed CSP. First, this subsection gives two equivalent definitions of an *accumulator*. The first defin-

ition emphasizes the link between the *accumulator* and the relaxed CSP as a whole. In this definition, the accumulator is a function which returns the number of constraints satisfied by each element of the search space. The second definition emphasizes on how to obtain the accumulator from constraints in the relaxed CSP. In the second definition the accumulator is the sum of the characteristic functions of the sets associated to the constraints in the relaxed CSP. Next, this subsection explains how can the *accumulator* be used to characterize the solution of a relaxed CSP. The *accumulator* is used in our context to represent the solution of a localization problem which can be cast into a relaxed CSP. In order to enable the exploitation of the solution, the image by a function of an *accumulator* is defined.

Definition 8 ***(Accumulator 1)*** *Consider the relaxed CSP*

$$\begin{array}{l} \mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ \mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{array} \tag{2.84}$$

An accumulator *is a function* $\mathcal{A} : \mathbb{R}^m \rightarrow \mathbb{N}$ *which for each element* $\mathbf{x} \in D$ *associates the number of constraints it satisfies*

$$\mathcal{A}(\mathbf{x}) = Card(\{i \in \{1, .., n\}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}). \tag{2.85}$$

Definition 9 *The characteristic function* χ *of the constraint* $C : \mathbf{f}(\mathbf{x}) \in [\mathbf{y}_i], \mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^p, [\mathbf{y}_i] \subset \mathbb{R}^p$ *is defined by*

$$\begin{cases} \chi(C)(\mathbf{x}) = 1 \text{ if } \mathbf{f}(\mathbf{x}) \in [\mathbf{y}_i] \\ \chi(C)(\mathbf{x}) = 0 \text{ otherwise.} \end{cases} \tag{2.86}$$

Definition 10 ***(Accumulator 2)*** *Consider the relaxed CSP defined in (2.84). We define the* accumulator *associated to this CSP as a function* $\mathcal{A} : \mathbb{R}^m \rightarrow \mathbb{N}$ *such as*

$$\mathcal{A}(\mathbf{x}) = \sum_{i \in \{1,..,n\}.} \chi(C_i)(\mathbf{x}). \tag{2.87}$$

Remark 7 *The name* accumulator *comes from the sum relation in (2.87).*

Remark 8 *In this formulation we consider only* hard constraints *which are necessarily to be either satisfied or not and what we seek is to compute the degree of satisfaction for each element of the search space. On the other hand,* soft constraints *[Benhamou and Ceberio, 2003] assume that constraints doesn't all have the same importance and have different weights associated to it. In our case, we associate a weight equal to 1 for all constraints. In the case of soft constraints a weight could be associated to the characteristic function of the constraint so that the definition of the* accumulator *would stay the same.*

The solution of the CSP can be written

$$\mathbb{S} = \{\mathbf{x} \in [\mathbf{x}], \mathcal{A}(\mathbf{x}) = n\}. \tag{2.88}$$

The solution set of a relaxed CSP assuming that at most q constraints are inconsistent can be written

$$\mathbb{S}_q = \{\mathbf{x} \in [\mathbf{x}], \mathcal{A}(\mathbf{x}) \geq n - q\}. \tag{2.89}$$

The solution set of elements satisfying the most of the constraints in the relaxed CSP also called *Max-CSP* (Solved in [Normand et al., 2010]) can be written

$$\mathbb{S}_{\max} = \{\mathbf{x} \in [\mathbf{x}], \mathcal{A}(\mathbf{x}) = \max(\mathcal{A}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^m)\}.$$

An *accumulator* actually represents a solution of a relaxed CSP. It is sometimes useful to compute the image of this accumulator by a function to be able to exploit the solution. As an example, in the context of localization of a robot (which can be cast into a relaxed CSP), the calculated position of the robot can be represented using an *accumulator*. Suppose the robot measures a relative position of an object in the environment using a sensor such as a camera. It is possible to deduce the *accumulator* corresponding to the absolute position of that object in the environment using translation function.

Definition 11 *The image by a bijective function* $\mathbf{f}$ *of an* accumulator $\mathcal{A}$ *is an* accumulator *denoted by* $\mathbf{f}(\mathcal{A})$ *such as*

$$\mathbf{f}(\mathcal{A})(\mathbf{x}) = \mathcal{A}(\mathbf{f}^{-1}(\mathbf{x})) \tag{2.90}$$

2.6.3 Solving distributed relaxed CSPs

In robotics, the constraints generally come from sensor data. Sensor data are not always available at the same time or the same place (ex: in the case of robot swarm or in case of an evolving process). The purpose is to solve the relaxed CSP involving all of the constraints despite the fact that the constraints are distributed (not available immediately at the same place/time of computation). The idea is to consider several smaller relaxed CSPs formed by subsets of available constraints then merge the resulting solutions into one. The different solutions of those relaxed CSPs are represented using *accumulators* and the merging is obtained by summing those *accumulators*. This allow distributed computations which are more adapted to distributed systems.

Definition 12 *The sum of two* accumulators $\mathcal{A}, \mathcal{B}$ *is an* accumulator *denoted* $\mathcal{A}+\mathcal{B}$ *such as*

$$\forall \mathbf{x} \in \mathbb{R}^m, (\mathcal{A} + \mathcal{B})(\mathbf{x}) = \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{x}) \tag{2.91}$$

Proposition 13 *If $\mathcal{A}, \mathcal{B}$ are the* accumulators *associated to the relaxed CSPs defined by the set of constraints $\{C_1, ..., C_r\}$ and $\{C_{r+1}, ..., C_n\}$ respectively and involving the same variable $\mathbf{x} \in \mathbb{R}^m$ then $\mathcal{A} + \mathcal{B}$ is the accumulator associated to the relaxed CSP defined by the set of constraints $\{C_1, ..., C_n\}$.*

proof. Directly inferred from definition 10. ■

Remark 9 *In practice, each* accumulator *can be stored as a grayscale image and be easily shared between distributed entities.*

2.6.4 Connection to fuzzy logic

It is possible to normalize an accumulator obtaining a function $\mu(\mathbf{x})$ ranged between 0 and 1 corresponding to the proportion of the number of satisfied constraints to the total number of constraints. $\mu(\mathbf{x})$ can be interpreted as a membership degree (in the sense of constraint satisfaction) and the function μ as a membership function is the sense off fuzzy logic [Klir and Yuan, 1995] [Nguyen and Walker, 2005] [Zadeh, 1965]. The set of all the values which satisfy at least $(1-\varepsilon)^{th}$ fraction of the constraints is the set of all values $\mathbf{x}$ for which $\mu(\mathbf{x}) \geq 1-\varepsilon$. In fuzzy terms, this set is known as an α-cut of the original fuzzy set corresponding to $\alpha = 1-\varepsilon$. Fuzzy sets are just an interpretation, however there is an analogy between accumulators and fuzzy sets which can actually help in computations by using known algorithms for processing fuzzy data. See the article [Sliwka et al., 2011a] for more details.

2.6.5 Connection to the generalized Hough transform

There is a resemblance between this formalism and the formalism of the generalized Hough transform [Bovik, 2000]. In fact, the generalized Hough Transform is used in image processing to detect shapes defined by unique function f and parameterized with a vector of parameters $\mathbf{p}$ on a binary image. The binary image can be represented by a vector $\mathbf{Y}$ corresponding to the coordinates of the pixel of interest (black or white depending on the convention). The idea is to compute an accumulator η for discrete values of the parameter $\mathbf{p}$ such as

$$\eta(\mathbf{p}) = card\{\mathbf{y} \in \mathbf{Y}, f(\mathbf{p}, \mathbf{y}) = 0\}. \tag{2.92}$$

As an example, in case of a circle detection which is parametered by its center (p_1, p_2) and the radius p_3, for each pixel $\mathbf{y} = (y_1, y_2)$ we have

$$f(\mathbf{p}, \mathbf{y}) = (y_1 - p_1)^2 + (y_2 - p_2)^2 - p_3^2. \tag{2.93}$$

In our case we are more general since we deal with a set of different constraints ($\mathbf{f}_i(\mathbf{p}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}$) instead of a unique type of parameterized constraints $f(\mathbf{p}, \mathbf{y}) = 0, \mathbf{y} \in \mathbf{Y}$. Besides, instead of using discrete values using interval values guarantees the result. Note that set membership methods have already been used in shape detection algorithms [Jaulin and Bazeille, 2009].

2.6.6 Conclusion

In this chapter an alternative representation of the solution of relaxed CSP has been presented. The idea is to use a function called an *accumulator* which for each element in the variable domain returns the number of constraints it satisfies. It is important to note that an accumulator is just a representation which help to understand constraint relaxation process. The *accumulators* related data structures and algorithms are explained in section 5.2 of chapter 5. The perspective in the research of the *accumulators* is to explore their similarity to image processing algorithms such as the generalized Hough transform and other fields such as fuzzy logic as well as p-boxes [Ferson et al., 2003].

Chapter 3

Interval analysis

3.1 Introduction

In this PhD thesis we use set membership methods to solve the localization problem. Basically, the solution of the problem (the position of the robot) is characterized by a set or a set of sets (set polynomials). Since the solution is computed using a computer it is necessary to have a representation of the sets on a computer. There are many possible approximation of a set $\mathbb{S}$ on a computer. A 2D example is shown in Figure 3.1. It is possible to use ellipsoids [Durieu et al., 1996] (Fig 3.1.(a)) or zonotopes [Combastel, 2005] (Fig 3.1.(b)). We choose to use boxes (Fig 3.1.(c)).

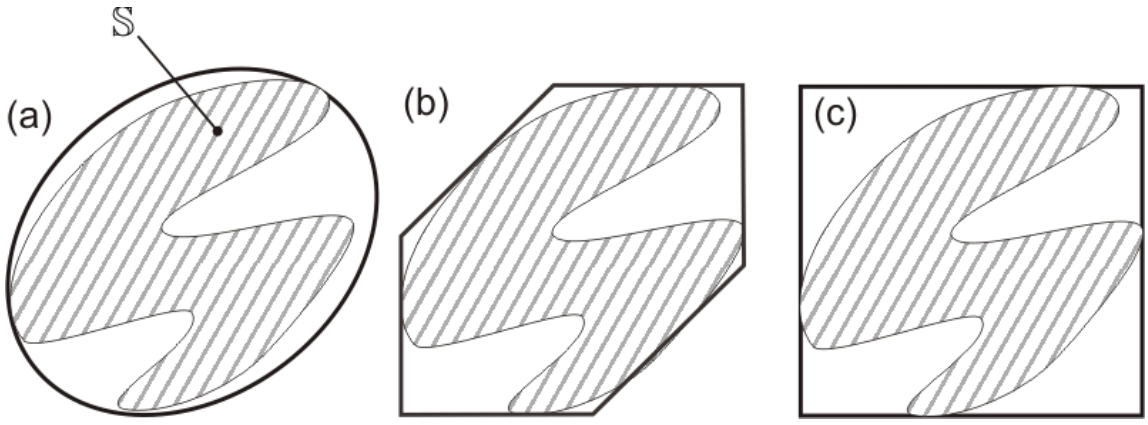

Figure 3.1: The different possible approximations of a set $\mathbb{S}$ by parametred sets (ellipsoids, zonotopes, boxes)

The use of boxes is part of a theory called *interval analysis* explained in more details in this chapter. Remark that the zonotopes and ellipsoids are a better approximation

of the set $\mathbb{S}$ than a box. On the other hand, the use of boxes simplify basic operations on sets such as intersection,union or image by a function hence our choice. *Interval analysis* was introduced when the first electronic computers were developed in the mid-20th century. Digital computations started to be widely used in science and engineering to solve mathematical problems. It was probably at that time that scientists realized that it is often impossible to obtain exact results and that it would be practical to bound the errors. Thus, scientists like Moore [Moore, 1959] and Warmus [Warmus, 1956] created solid bases for *interval analysis*. Today, *interval analysis* is not only used to handle errors but to solve many problems in which sets are involved. *Interval analysis* is very similar to the real analysis, the main difference is that the variables are represented by an interval and not real punctual numbers.

It is also possible to extend the concept of intervals to more complex sets and consider functions intervals (also called Tubes) [Le Bars et al., 2011] or intervals of sets or subpavings as seen in [Jaulin, 2011]. Figure 3.2 illustrates those new intervals.

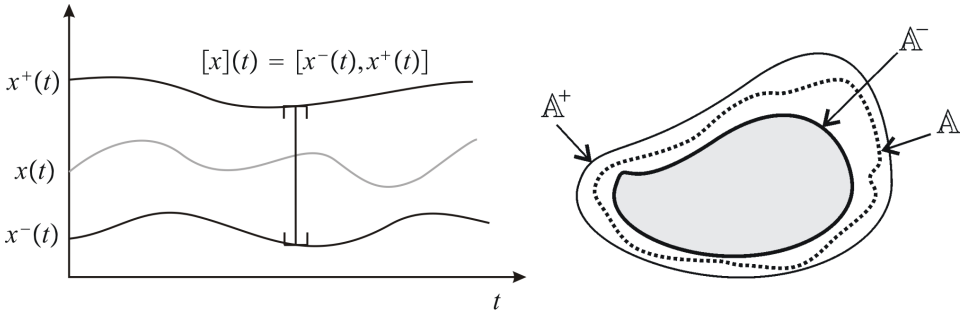

Figure 3.2: Other types of intervals : function intervals and set intervals

3.2 Definitions and notations

Definition 13 ***(Interval)*** *An interval is a connected and closed subset of* $\mathbb{R}$*. If* x *is a real variable we denote by* $[x]$ *the interval containing this variable.* $[x]$ *is called the domain of* x*. An interval has an upper and lower bound which we will note as follows* $[x] = [x^-, x^+]$*.* $\mathbb{IR}$ *is the set of all the real intervals.* $\mathbb{IN}$ *is the set of natural number intervals.* $w([x]) = x^+ - x^-$ *is called width of* $[x]$*.*

Example 15 $\emptyset, \{-1\}, [-1, 1], [-1, \infty], \mathbb{R}$ *are intervals.*

Definition 14 ***(Box)*** *A box of* $\mathbb{R}^n$ *is defined by a Cartesian product of intervals. A box can be also considered as an interval vector. If* $\mathbf{x} = (x_1, .., x_n) \in \mathbb{R}^n$ *is a real variable vector we denote by* $[\mathbf{x}] = ([x_1], .., [x_n])$ *the box containing this variable ([Jaulin et al., 2001])*

Example 16 $[1,3] \times [2,4]$ *is a box of* $\mathbb{R}^2$

3.3 Binary operations

As it is the case for real analysis, it is possible to define different operations on intervals such as sum or product. If $\diamond \in \{+, -, *, /, \max, \min\}$, where $*$ is the product, and if $[x]$ and $[y]$ are two intervals, we define

$$[x] \diamond [y] \triangleq [\{x \diamond y \mid x \in [x], y \in [y]\}]. \tag{3.1}$$

Therefore,

$$\begin{array}{ll} [x^-, x^+] + [y^-, y^+] & = [x^- + y^-, x^+ + y^+] \\ [x^-, x^+] \cdot [y^-, y^+] & = [\min(x^-y^-, x^+y^-, x^-y^+, x^+y^+), \\ & \qquad \max(x^-y^-, x^+y^-, x^-y^+, x^+y^+)] \\ \{1\}/[x^-, x^+] & = [\min(\tfrac{1}{x^-}, \tfrac{1}{x^+}), \max(\tfrac{1}{x^-}, \tfrac{1}{x^+})] \\ \max\left([x^-, x^+], [y^-, y^+]\right) & = [\max(x^-, y^-), \max(x^+, y^+)]. \end{array} \tag{3.2}$$

Remark 10 *Note All those operations are commutative and associative.*

Since $\{1\} \cdot [x] = [x]$ *then*

$$[x]/[y] = ([x] \cdot \{1\})/[y] = [x] \cdot (\{1\}/[y]). \tag{3.3}$$

Example 17 *Here are some examples of results of binary operations on intervals*

$$\begin{array}{rl} [-2,5] + [1,3] & = [-1,8], \\ [-2,5].[1,3] & = [-3,15], \\ [-2,5]/[1,3] & = [-\tfrac{2}{3}, 5], \\ \min\left([-2,5],[1,3]\right) & = [-2,3]. \end{array} \tag{3.4}$$

A more complex one

$$([1,2] + [-3,4]) * [5,6] = [-2,6] * [5,6] = [-12,36]. \tag{3.5}$$

3.4 Elementary functions

If $f \in \{\cos, \sin, \sqrt{()}, ()^2, \log, \exp, \ldots\}$, is an elementary function from $\mathbb{R}$ to $\mathbb{R}$, we define its interval extension as

$$f([x]) \triangleq [\{f(x) \mid x \in [x]\}]. \tag{3.6}$$

Example 18 *Here are some examples of the interval evaluation of elementary functions*

$$\begin{array}{l} \sin([0,\pi]) = [0,1], sqr([-2,3]) = [-2,3]^2 = [0,9], \\ abs([-5,1]) = [0,5], sqrt([-13,4]) = \sqrt{[-13,4]} = [0,2], \\ \exp([0,1]) = [1,e]. \end{array} \tag{3.7}$$

3.5 Inclusion functions

It is possible to evaluate any kind of function. In order to make an evaluation of the resulting interval/box by a function, interval functions called *inclusion functions* are used.

Definition 15 *Let* $\mathbf{f}$ *be a function from* $\mathbb{R}^n$ *to* $\mathbb{R}^m$*. The interval function* $[\mathbf{f}]$ *from* $\mathbb{IR}^n$ *to* $\mathbb{IR}^m$*, is an* inclusion function *of* $\mathbf{f}$ *if*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]). \tag{3.8}$$

Definition 16 *The* inclusion function $[\mathbf{f}]$ *is minimal if for each input box* $[\mathbf{x}]$ *the output box is the smallest possible box enclosing all the values of* $\mathbf{f}$ *on* $[\mathbf{x}]$ *i.e.*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{f}]([\mathbf{x}]) = \min_{\subset}([\mathbf{y}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{y}]). \tag{3.9}$$

There are many ways to obtain an inclusion function. One of them is the *natural inclusion function* that applies to functions which are a finite composition of operators $+, -, *, /$ and elementary functions $(\sin, \cos, \exp, \ln, \_^n, ...)$.

Consider such a function denoted by

$$\begin{array}{l} \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m \\ (x_1, .., x_n) \mapsto \mathbf{f}(x_1, .., x_n). \end{array} \tag{3.10}$$

The *natural inclusion function* $[\mathbf{f}]$ is obtained by replacing each real variable x_i by an interval variable $[x_i]$ and each operator or function by its interval counterpart. $[\mathbf{f}]$ is not always minimal. There are many other ways to create an better approximation inclusion function as *centered inclusion functions* or *Tailor inclusion functions* [Jaulin et al., 2001].

Example 19 *Consider*

$$\begin{array}{l} f : \mathbb{R} \to \mathbb{R} \\ x \mapsto x^2 + 2x + 1. \end{array} \tag{3.11}$$

For each formal expression of the same function f corresponds a different natural inclusion function. Consider three formal expressions of f

$$\begin{aligned} f_1(x) &= x^2 + 2x + 1 \\ f_2(x) &= (x+1) * (x+1) \\ f_3(x) &= (x+1)^2 \end{aligned} \tag{3.12}$$

Evaluating those functions for $[x] = [-2, 2]$ give different results for each function.

$$\begin{aligned} [f_1]([x]) &= [x]^2 + 2[x] + 1 = [0, 4] + [-4, 4] + 1 = [-3, 9] \\ [f_2]([x]) &= ([x] + 1) * ([x] + 1) = [-1, 3] * [-1, 3] = [-3, 9] \\ [f_3]([x]) &= ([x] + 1)^2 = [-1, 3]^2 = [0, 9]. \end{aligned} \tag{3.13}$$

The inclusion function $[\mathbf{f}]$ approximates the image set $\mathbf{f}([\mathbf{x}])$ by a box $[\mathbf{f}]([\mathbf{x}])$. This box contains elements which are not necessarily an image of the function. As such, composing multiple inclusion functions leads to an overestimation of the output box. This is called the *wrapping effect*. This effect becomes apparent when considering $[\mathbf{f}] \circ [\mathbf{f}^{-1}]$ as an inclusion function of $\mathbf{f} \circ \mathbf{f}^{-1}$. Figure 3.3 shows an example of such attempt where $\mathbf{f}$ is the plane rotation by a $\frac{\pi}{4}$ angle. Remark that $[\mathbf{f}] \circ [\mathbf{f}^{-1}]([\mathbf{x}])$ is a lot larger than $[\mathbf{x}]$ although $\mathbf{f} \circ \mathbf{f}^{-1} = I$ (identity).

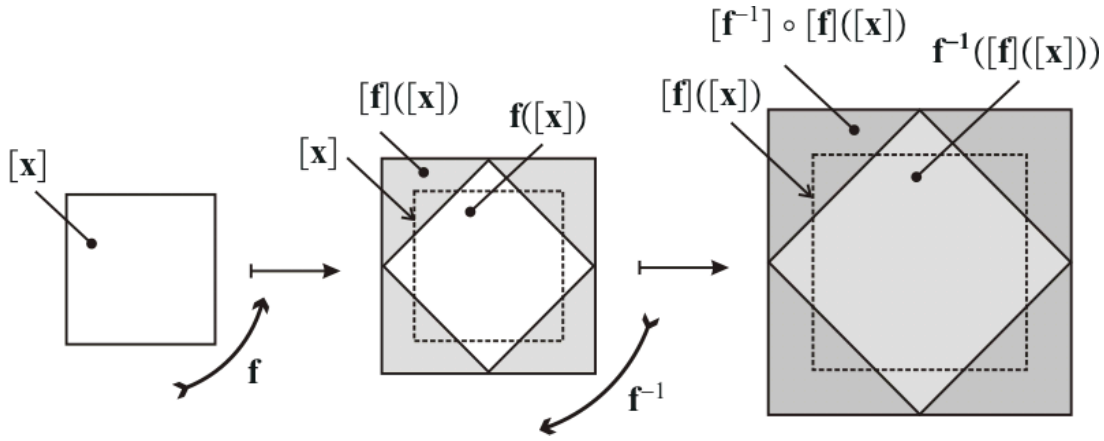

Figure 3.3: Wrapping effect caused by $\frac{\pi}{4}$ rotation function

3.6 Operations on boxes

3.6.1 Union and intersection

Union : Since we work with boxes and not with union of boxes, we define $\sqcup$ the box union operator defined in $\mathbb{IR}^n$. Consider three boxes $A, B, C \in \mathbb{IR}^n$ such as $C = A \sqcup B$. C is the smallest box containing both A and B [Hyvönen, 1992][Jaulin et al., 2001].

Intersection : The intersection of boxes in $\mathbb{R}^n$ is the same that the one applied to $\mathbb{R}^n$ subsets and is denoted $\cap$.

3.6.2 Relaxed intersection of boxes

Section 2.4 introduced the relaxed intersection of sets and its uses to define the solution of a relaxed CSP. In this subsection, the relaxed intersection of boxes is presented. The relaxed intersection of boxes is used in different algorithms such as RSIVIA [Jaulin, 2009] to implement the relaxed intersection of sets. A polynomial-time algorithm to compute the relaxed intersection of boxes is described in this subsection.

Relaxed intersection of intervals

The relaxed intersection of boxes is based on the relaxed intersection of intervals. Consider p intervals $[x_1], .., [x_p]$. Denote by $[x]$ the result of q-relaxed intersection of those intervals

$$[x] = \bigcap_{i\in\{1,..,p\}}^{\{q\}} [x_i]. \tag{3.14}$$

Consider $\mathbf{b}$ the vector of bounds of all the intervals $[x_i]$ which are sorted in ascending order. Consider a function ξ which tells if bound is a lower or upper bound of the corresponding interval such as if b_j is the bound of the interval $[x_i]$ then

$$\begin{aligned} \xi(b_j) &= 1 \text{ if } b_j = x_i^- \\ \xi(b_j) &= -1 \text{ if } b_j = x_i^+. \end{aligned}$$

For each bound b_j we associate a weight s_j defined by

$$s_j = \sum_{i\in\{1,..,j\}} \xi(b_i).$$

We have

$$\begin{aligned} x^+ &= \max_{j\in\{2,..,2p\}} (b_j, s_{j-1} \geq q) \\ x^- &= \min_{j\in\{1,..,2p\}} (b_j, s_j \geq q) \end{aligned}$$

Example 20 *Figure 3.4 shows two cases of* 1*-relaxed intersection of* 4 *intervals. In the (b) case, the result of the* 1*-relaxed intersection is two disjointed intervals but since we work with single intervals, the algorithm returns the union of those two intervals.*

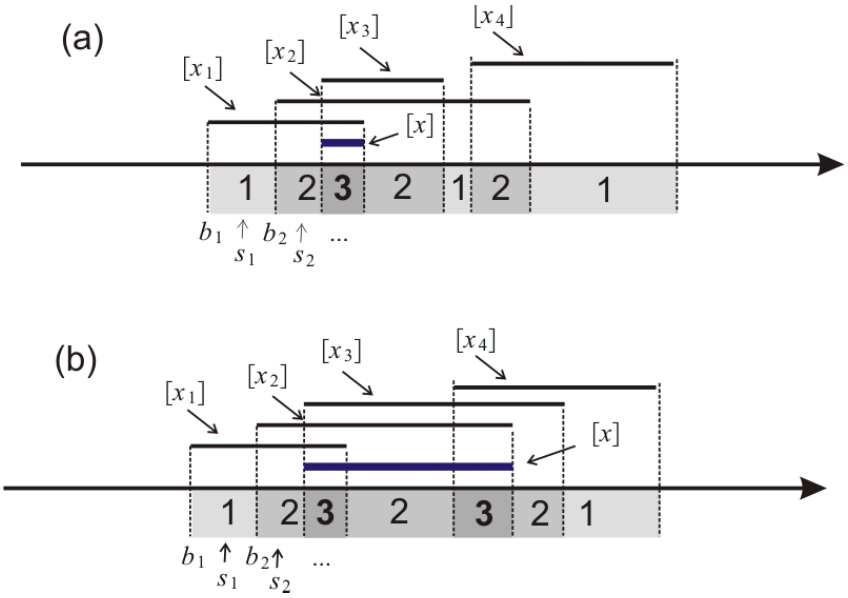

Figure 3.4: Example of a 1-relaxed intersection of 4 intervals

Reducing the relaxed intersection of boxes to relaxed intersection of intervals

Consider the boxes $[\mathbf{x}_1], ..., [\mathbf{x}_p] \in \mathbb{IR}^n$. Denote by $[\mathbf{x}]$ the result of the q-relaxed intersection of those boxes.

$$[\mathbf{x}] = \bigcap_{i \in \{1,..,p\}}^{\{q\}} [\mathbf{x}_i] \tag{3.15}$$

The basic method is to deduce the boxes intersection from the intersection of the intervals for each dimension $k \in \{1, .., n\}$ of the boxes.

$$[x_k] = \bigcap_{i \in \{1,..,p\}}^{\{q\}} [x_{ik}] \tag{3.16}$$

Example 21 *Figure 3.5 illustrates a* 1*-relaxed intersection of* 4 *boxes* $[\mathbf{x}_1], [\mathbf{x}_2], [\mathbf{x}_3]$ *and* $[\mathbf{x}_4]$ *using the algorithm described before. The actual result of the* 1*-relaxed intersection is the hatched part of the solution given by the algorithm. This pessimism is due to the fact of considering one dimension at a time. In practice, this pessimism doesn't affect much the computation time.*

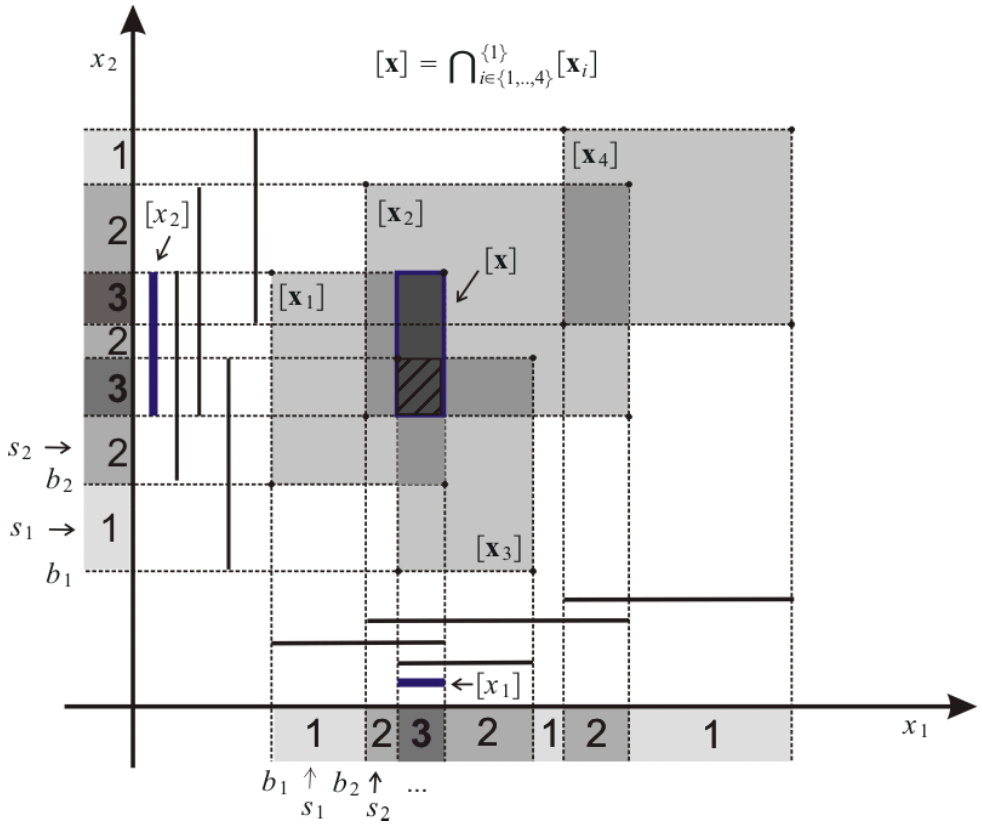

Figure 3.5: Illustration of the relaxed intersection algorithm using 4 boxes

3.7 Subpavings

Any set $\mathbb{S} \subset \mathbb{R}^n$ can be approximated by a box or a set of non overlapping boxes also called a *subpaving*. The more boxes there are in the *subpaving* the more the approximation can be precise. Figure 3.6 shows the different possible approximations of a set $\mathbb{S}$ in form of a disc using *subpavings*. The box in sub-figure (a) is *enclosing* the set $\mathbb{S}$. The *subpaving* in the sub-figure (b) is a *regular subpaving* and the one in the sub-figure (c) is called an *irregular subpaving*. Set membership CSP Solvers (such as SIVIA or *contractor* based algorithms used in QUIMPER Software [Chabert and Jaulin, 2009a]) generate subpavings as a mean to represent the solution of a CSP. It is possible to adapt set membership solvers to solve relaxed CSPs by actually creating adapted *contractors*. The next chapter introduces *contractors* which are an algorithmic entity used in some of those CSP solvers.

3.8 Conclusion

This chapter introduced the *interval analysis* theory which enables to represent the uncertainty in the form of intervals. If a calculus involves uncertain variables, by using intervals

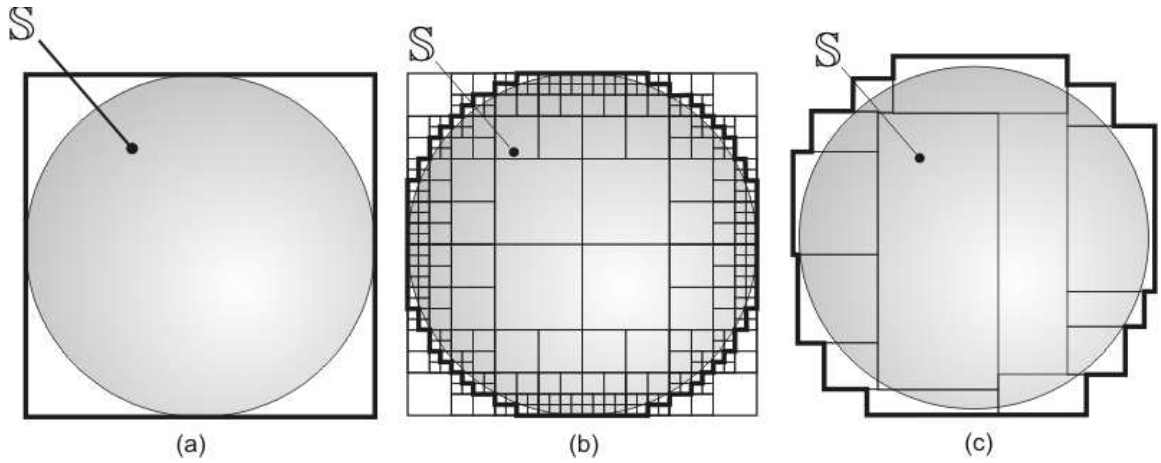

Figure 3.6: Using boxes to approximate sets

instead of real variables it is possible to propagate the uncertainty over all the variables involved in the calculus. Another use for interval analysis is to provide tools to represent and compute an approximation of any subset of $\mathbb{R}^n$. The $\mathbb{R}^n$ sets are approximated using *subpavings* (a set of boxes). Subpavings can be computed using solvers such as SIVIA (Set Inversion Via Interval Analysis) or *contractor* based solvers. The next chapter introduces those *contractors*.

Chapter 4

Contractors

4.1 Introduction

One of the set membership approaches to solve CSPs is to use an algorithmic representation of the constraints called *contractors*. If a constraint in the CSP defines a set of points which satisfy (or not) that constraint then the associated *contractor* enables to compute an approximation of that set on a computer. The *contractors* associated to set of constraints in CSP can be then combined to create a unique contractor allowing to compute the solution set of that CSP.

This chapter first defines the *contractors* and introduces some of their key properties. A *contractor* C is minimal if there is no better contracting *contractor* for the set associated to C. Section 4.6 introduces one of the contributions in the form of two theorems allowing to construct minimal contractors from other minimal contractors. The first theorem claims that the union of minimal contractors is also minimal. The second theorem claims that the transform of a minimal contractor (central symmetry, some axial symmetries, homothetie) is also minimal. As such it is possible to construct a complex minimal contractor from simple minimal contractors through transformations and unions. Section 4.7 presents applications of those two theorems.

Section 4.8 introduces another contribution to the contractor theory which is the *image contractor*. The *image contractor* is the contractor associated to the set defined by black (or white) pixels on a binary image. The *image contractor* allows to represent hardly parametrable sets such as maps in the context of localization. The *image contractor* can also be used to approximate complex contractors which require heavy computations. This section also presents an application of the image contractor for the localization in a city without GPS positioning.

4.2 Definition

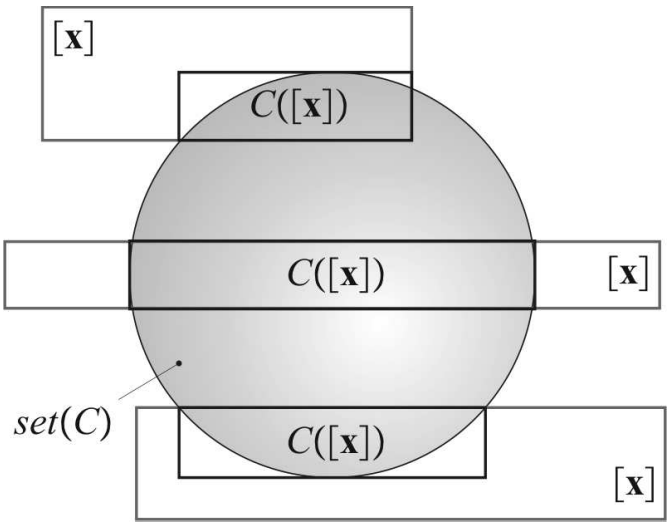

Figure 4.1: Three cases of contraction of a box $[\mathbf{x}]$

A *contractor* C [Jaulin et al., 2001] [Chabert and Jaulin, 2009a] is an algorithmic entity used to represent a specific set denoted $set(C)$. It is basically an operator applied on a box which shrinks it so that no point belonging to both $set(C)$ and the initial box find itself outside the box after shrinking. More precisely,

Definition 17 *The operator* $C : \mathbb{IR}^n \to \mathbb{IR}^n$ *is a* contractor *if*

(i) $\forall [\mathbf{x}] \in \mathbb{IR}^n, C([\mathbf{x}]) \subset [\mathbf{x}]$ *(contractance)*

(ii) $(\mathbf{x} \in [\mathbf{x}], C(\{\mathbf{x}\}) = \{\mathbf{x}\}) \Rightarrow \mathbf{x} \in C([\mathbf{x}])$ *(consistency)*

(iii) $C(\{\mathbf{x}\}) = \emptyset \Leftrightarrow (\exists \varepsilon > 0, \forall [\mathbf{x}] \subset B(\mathbf{x}, \varepsilon), C([\mathbf{x}]) = \emptyset)$ *(convergence)*

where $B(\mathbf{x}, \varepsilon)$ *is the ball which the center is* $\mathbf{x}$ *and the radius is* ε*.*

A box $[\mathbf{x}]$ is said to be "insensitive" to the *contractor* C if $C([\mathbf{x}]) = [\mathbf{x}]$. The property (i) ensures that by applying a *contractor*, a box can only get smaller. The property (ii) means that every box will keep, after contraction, all the elements $\mathbf{x}$ which are "insensitive" to the *contractor*. Lastly, (iii) ensures us that the set of the elements "sensitive" to the *contractor* forms an "open set" while the set of the elements "insensitive" to the *contractor* is "closed". The set associated to a *contractor* C is the set of all the singletons "insensitive" to C i.e

$$set(C) = \{\mathbf{x} \in \mathbb{R}^n, C(\{\mathbf{x}\}) = \{\mathbf{x}\}\}$$

Figure 4.1 shows the action of a *contractor* which set of insensitive points is a disk from $\mathbb{R}^2$. A *contractor* can be seen as a way to represent a subset of $\mathbb{R}^n$. *Contractors* are used in computer algorithms as a mean to manipulate sets in an easy way and allow to make intersection, union or set inversion. A Software called QUIMPER [Chabert and Jaulin, 2009a] was specifically designed to implement *contractor* computations.

We consider the following properties of contractors

$$\begin{array}{ll} C \text{ is } \textit{monotonic} \text{ if} & [\mathbf{x}] \subset [\mathbf{y}] \Rightarrow C([\mathbf{x}]) \subset C([\mathbf{y}]) \\ C \text{ is } \textit{minimal} \text{ if} & \forall [x] \in \mathbb{IR}^n, C([\mathbf{x}]) = [[\mathbf{x}] \cap \textit{set}(C)] \\ C \text{ is } \textit{idempotent} \text{ if} & \forall [x] \in \mathbb{IR}^n, C(C([\mathbf{x}])) = C([\mathbf{x}]). \end{array} \tag{4.1}$$

The *minimality* of a *contractor* is explained in more details in section 4.6.

4.3 From constraints to contractors

If a constraint (see 2.2) is the mathematical definition of the set then the *contractor* is the algorithm which enables to compute an approximation of the set in form of subpaving (see section 3.7).

Example 22 *Consider a constraint* $\mathbf{x} = (x, y) \in [\mathbf{x}], ax + by + c = 0$ *where* $a, b, c \in \mathbb{R}$ *are known constants. Figure 4.2 shows a* contractor C *associated to that constraint. The input boxes are represented in light gray and the contracted boxes are represented in dark gray.*

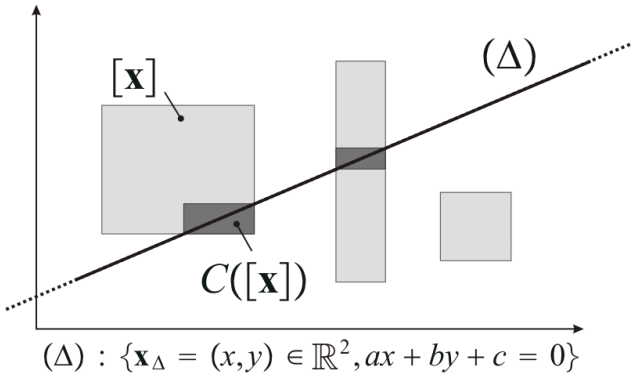

Figure 4.2: The contractor associated to the "on the line" constraint

Associated to appropriate solvers it is possible to generate subpavings representing a particular set defined by a constraint (see Figure 4.3.(a),(b)) or a set of constraints (as seen in Figure 4.3.(c)).

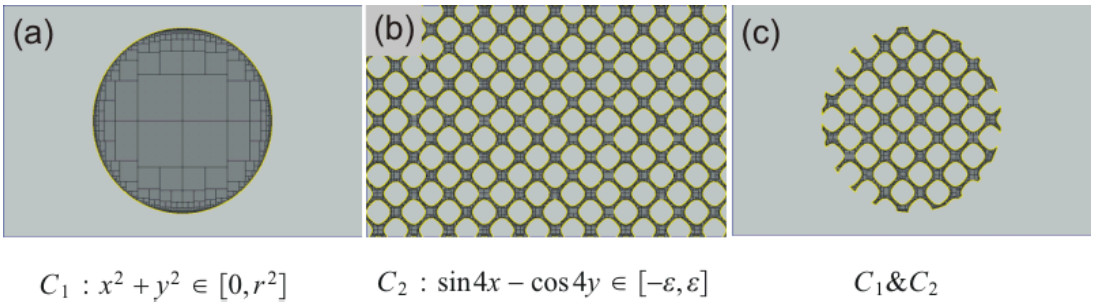

Figure 4.3: An example of subpavings generated using contractor based solvers

4.4 Operations on contractors and related theorems

The contractors are subject to the following operations:

intersection	$(C_1 \cap C_2)([\mathbf{x}]) \stackrel{def}{=} C_1([\mathbf{x}]) \cap C_2([\mathbf{x}])$
union	$(C_1 \sqcup C_2)([\mathbf{x}]) \stackrel{def}{=} C_1([\mathbf{x}]) \sqcup C_2([\mathbf{x}])$
composition	$(C_1 \circ C_2)([\mathbf{x}]) \stackrel{def}{=} C_1(C_2([\mathbf{x}]))$
repetition	$C_1^{\infty} \stackrel{def}{=} C_1 \circ C_1 \circ C_1 \circ ...$
repeated intersection	$C_1 \sqcap C_2 \stackrel{def}{=} (C_1 \cap C_2)^{\infty}$
repeated multiple intersection	$\prod_{i \in \{1,..,n\}} C_i \stackrel{def}{=} (C_1 \cap ... \cap C_n)^{\infty}$

(4.2)

Remark 11 *Note that contrarily to the others operations ($\cap, \cup, \sqcap, \sqcup$), the composition is not commutative (i.e., $C_1 \circ C_2 \neq C_2 \circ C_1$). Figure 4.4 shows an example of the composition of two line contractors which is not commutative.* $[\mathbf{x}]$ *is represented in light gray while* $C_i \circ C_j([\mathbf{x}])$ *is represented in dark gray.*

We define also more complex operations which are useful in solving relaxed CSPs

q-relaxed intersection	$\bigcap_{i \in \{1,..,n\}}^{\{q\}} C_i \stackrel{def}{=} \bigsqcup_{\mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K}) = n-q} \left(\bigcap_{i \in \mathbb{K}} C_i \right)$
q-relaxed composition	$\bigodot^{\{q\}} (C_n, .., C_1) \stackrel{def}{=} \bigsqcup_{\substack{i_{n-q} > .. > i_1 \\ \{i_1,..,i_{n-q}\} \subset \{1,..,n\}}} C_{i_{n-q}} \circ ... \circ C_{i_1}$
repeated q-relaxed intersection	$\prod_{i \in \{1,..,n\}}^{\{q\}} C_i \stackrel{def}{=} \left(\bigcap_{i \in \{1,..,n\}}^{\{q\}} C_i \right)^{\infty}$.

(4.3)

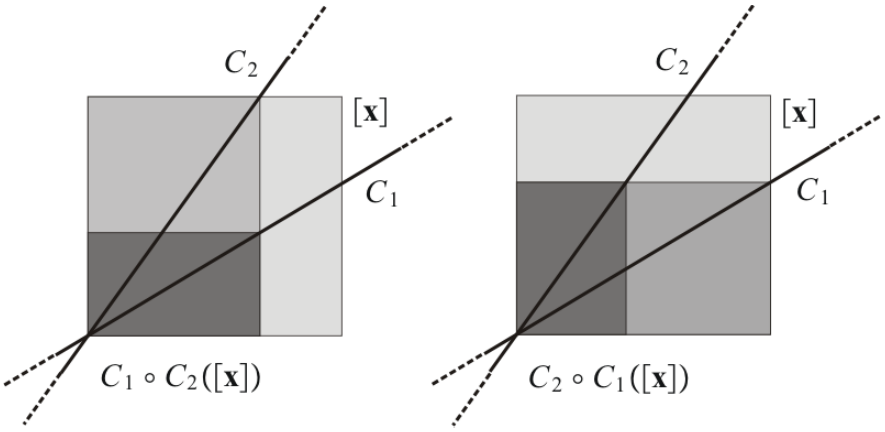

Figure 4.4: The composition of contractors is not commutative

Note that

$$\forall [\mathbf{x}], \bigsqcup_{\mathbb{K}\subset\{1,..,n\}, Card(\mathbb{K})=n-q} \left(\bigcap_{i\in\mathbb{K}} C_i([\mathbf{x}]) \right) = \bigcap_{i\in\{1,..,n\}}^{\{q\}} C_i([\mathbf{x}]). \tag{4.4}$$

where $\bigcap_{i\in\{1,..,n\}}^{\{q\}}$ is the q-relaxed intersection of boxes defined in subsection 2.4.3.

Theorem 14 *Consider C_1 and C_2 two monotonic contractors. We have*

$$C_1 \subset C_2 \Rightarrow C_1^\infty \subset C_2^\infty \tag{4.5}$$

proof. Consider $n \in \mathbb{N}$ for which

$$C_1^n \subset C_2^n. \tag{4.6}$$

For $n = 1$ the statement is verified.

Since C_1 is monotonic we have

$$C_1^{n+1} = C_1 \circ C_1^n \quad \subset C_1 \circ C_2^n. \tag{4.7}$$

Since $C_1 \subset C_2$ we have

$$C_1 \circ C_2^n \quad \subset C_2 \circ C_2^n = C_2^{n+1}. \tag{4.8}$$

The statement is then verified for all $n \in \mathbb{N}$. ■

Theorem 15 ***(L. Jaulin's theorem of unique repetition)*** *Consider C_1 and C_2 two monotonic contractors (not necessarily idempotent) then*

$$\begin{array}{lll} (C_1^\infty \cap C_2^\infty)^\infty & = (C_1 \cap C_2)^\infty & \\ or\ C_1^\infty \sqcap C_2^\infty & = C_1 \sqcap C_2 & (i) \\ (C_1^\infty \circ C_2^\infty)^\infty & = (C_1 \circ C_2)^\infty . & (ii) \end{array} \tag{4.9}$$

proof. (i) Consider two monotonic contractors C_1 and C_2. We have

$$\begin{array}{rl}(C_1 \cap C_2)^\infty & = (C_1 \cap C_2) \circ (C_1 \cap C_2)^\infty \\ & = (C_1 \circ (C_1 \cap C_2)^\infty) \cap (C_2 \circ (C_1 \cap C_2)^\infty)\,.\end{array} \tag{4.10}$$

which means that

$$\begin{array}{l}C_1 \circ (C_1 \cap C_2)^\infty = (C_1 \cap C_2)^\infty \\ C_2 \circ (C_1 \cap C_2)^\infty = (C_1 \cap C_2)^\infty\,.\end{array} \tag{4.11}$$

otherwise it would lead to an inconsistency ($(C_1 \cap C_2)^\infty$ strictly included in itself).

We deduce that

$$(C_1^\infty \cap C_2^\infty)^\infty \circ (C_1 \cap C_2)^\infty = (C_1 \cap C_2)^\infty\,. \tag{4.12}$$

On the other hand since the contractors are monotonic then

$$(C_1^\infty \cap C_2^\infty)^\infty \circ (C_1 \cap C_2)^\infty \subset (C_1^\infty \cap C_2^\infty)^\infty\,. \tag{4.13}$$

From (4.12) and (4.13) we infer that

$$(C_1 \cap C_2)^\infty \subset (C_1^\infty \cap C_2^\infty)^\infty\,. \tag{4.14}$$

The other inclusion is deduced from theorem 14. Since

$$C_1^\infty \cap C_2^\infty \subset C_1 \cap C_2. \tag{4.15}$$

and the contractors are monotonic, we have

$$(C_1^\infty \cap C_2^\infty)^\infty \subset (C_1 \cap C_2)^\infty\,. \tag{4.16}$$

The proof of (ii) is very similar to the proof of (i). ■

Example 23 *Given three contractors $C_1, .., C_3$ we have*

$$((C_1^\infty \sqcap C_2^\infty) \circ C_3^\infty)^\infty = ((C_1 \cap C_2) \circ C_3)^\infty\,. \tag{4.17}$$

Remark 12 *Remark that this theorem allows to save some computation resources when implementing contractors on computers since there are less contractor evaluations to be made.*

Remark 13 *Note that the unique repetition property is not valid for the union $\sqcup$ of contractors. Only the inclusion is guaranteed*

$$(C_1^\infty \sqcup C_2^\infty)^\infty \quad \subset (C_1 \sqcup C_2)^\infty\,. \tag{4.18}$$

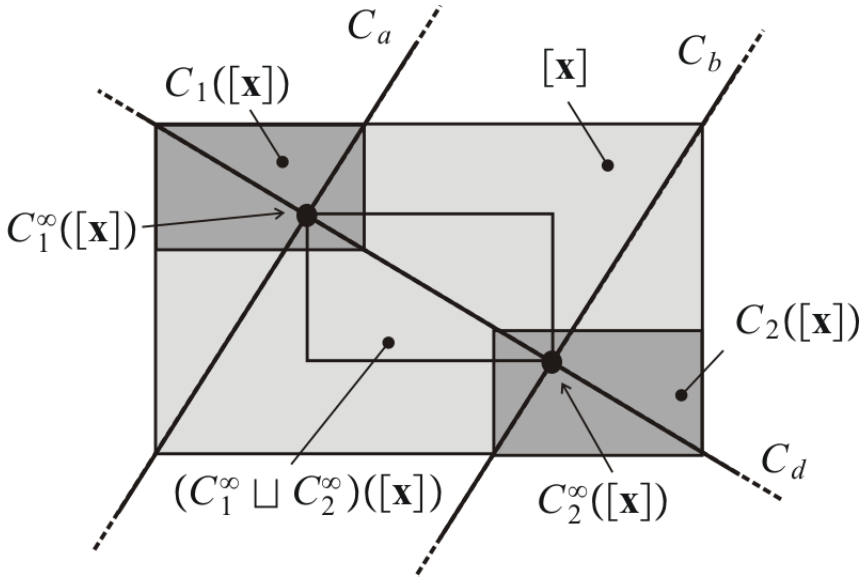

Figure 4.5: An example where the theorem of unique repetition is not viable for the $\sqcup$ operator.

Example 24 *Figure 4.5 shows an example which confirms what is said in the remark. Consider three contractors C_a, C_b and C_d which sets are represented on the Figure. Denote by*

$$\begin{array}{l} C_1 = C_d \circ C_a \\ C_2 = C_d \circ C_b. \end{array} \tag{4.19}$$

Note that

$$\forall\, [\mathbf{x}] \in \mathbb{R}^m, \forall k \in \mathbb{N}, \left(C_1 \sqcup C_2\right)^k ([\mathbf{x}]) = [\mathbf{x}].$$

We have

$$(C_1^\infty \sqcup C_2^\infty)^\infty \quad \neq (C_1 \sqcup C_2)^\infty$$

Theorem 16 *Consider C_1 and C_2 two contractors. We have*

$$(C_1 \circ C_2)^\infty = C_1 \sqcap C_2. \tag{4.20}$$

proof. Consider $[\mathbf{x}] \in \mathbb{IR}^m$ Denote by

$$\begin{array}{l} [\mathbf{y}] = (C_1 \circ C_2)^\infty ([\mathbf{x}]) \\ [\mathbf{z}] = (C_1 \sqcap C_2) ([\mathbf{x}]) \,. \end{array} \tag{4.21}$$

We have $(C_1 \circ C_2)([\mathbf{y}]) = [\mathbf{y}]$ as such $C_2([\mathbf{y}]) = [\mathbf{y}]$ and $C_1([\mathbf{y}]) = [\mathbf{y}]$.

As such

$$(C_1 \sqcap C_2)([\mathbf{y}]) = [\mathbf{y}]\,. \tag{4.22}$$

The contractor $(C_1 \sqcap C_2)$ is idempotent. Theorem 4.3 in [Jaulin et al., 2001], which is actually a special case of the Knaster–Tarski theorem [Tarski, 1955], says that an idempotent contractor C converges to the largest box $[\mathbf{a}]$ included in the box it contracts $[\mathbf{x}]$ such as $C([\mathbf{a}]) = [\mathbf{a}]$. This means that $[\mathbf{y}] \subset [\mathbf{z}]$.

On the other hand we have $(C_1 \cap C_2)([\mathbf{z}]) = [\mathbf{z}]$ as such $C_2([\mathbf{z}]) = [\mathbf{z}]$ and $C_1([\mathbf{z}]) = [\mathbf{z}]$.

As such

$$(C_1 \circ C_2)([\mathbf{z}]) = [\mathbf{z}]\,. \tag{4.23}$$

The contractor $(C_1 \circ C_2)$ is idempotent. Using the same theorem, we prove that $[\mathbf{z}] \subset [\mathbf{y}]$ and conclude that $[\mathbf{y}] = [\mathbf{z}]$ *i.e.*

$$(C_1 \circ C_2)^{\infty}([\mathbf{x}]) = (C_1 \sqcap C_2)([\mathbf{x}])\,. \tag{4.24}$$

■

Theorem 17 *The union of two idempotent and monotonic contractors is also idempotent* i.e. *given two idempotent and monotonic contractors C_1 and C_2 we have*

$$(C_1 \sqcup C_2)^2 = C_1 \sqcup C_2. \tag{4.25}$$

proof. Consider $[\mathbf{x}] \in \mathbb{IR}^m$ denote by $[\mathbf{y}] = (C_1 \sqcup C_2)([\mathbf{x}])$. We have

$$\begin{array}{ll} (C_1 \sqcup C_2)^2([\mathbf{x}]) & = C_1((C_1 \sqcup C_2)([\mathbf{x}])) \sqcup C_2((C_1 \sqcup C_2)([\mathbf{x}])) \\ & = C_1([\mathbf{y}]) \sqcup C_2([\mathbf{y}])\,. \end{array} \tag{4.26}$$

Theorem 4.3 in [Jaulin et al., 2001], which is actually a special case of the Knaster–Tarski theorem [Tarski, 1955], says that an idempotent contractor C converges to the largest box $[\mathbf{z}]$ included in the box it contracts $[\mathbf{x}]$ such as $C([\mathbf{z}]) = [\mathbf{z}]$ which means that $\forall\,[\mathbf{y}]\,,[\mathbf{z}] \subseteq [\mathbf{y}] \subseteq [\mathbf{x}] \Rightarrow C([\mathbf{y}]) = [\mathbf{z}]$.

Since $C_1([\mathbf{x}]) \subset [\mathbf{y}]$ and $C_2([\mathbf{x}]) \subset [\mathbf{y}]$ we have $C_1([\mathbf{y}]) = C_1([\mathbf{x}])$ and $C_2([\mathbf{y}]) = C_2([\mathbf{x}])$. As such

$$(C_1 \sqcup C_2)^2([\mathbf{x}]) = C_1([\mathbf{x}]) \sqcup C_2([\mathbf{x}]) = C_1 \sqcup C_2([\mathbf{x}])\,. \tag{4.27}$$

■

Proposition 18 *Consider n monotonic contractors $C_1, .., C_n$. We have*

$$\begin{array}{lll} \left(\bigcap\limits_{i\in\{1,..,n\}}^{\{q\}} C_i\right)^{\infty} & = \left(\bigsqcup\limits_{\mathbb{K}\subset\{1,..,n\},Card(\mathbb{K})=n-q} \left(\prod\limits_{i\in\mathbb{K}} C_i\right)\right)^{\infty} & (i) \\ & = \bigsqcup\limits_{\mathbb{K}\subset\{1,..,n\},Card(\mathbb{K})=n-q} \left(\prod\limits_{i\in\mathbb{K}} C_i\right). & (ii) \end{array} \tag{4.28}$$

proof. *The equation 4.28.(i) comes from the theorem of unique repetition 15. The equation 4.28.(ii) comes from the theorem.17.* ■

The last proposition is used in the implementation of contractor polynomials (not yet defined) used in subsection 5.1.

Proposition 19 *Consider the contractors C_1 ,..,C_n. We have*

$$\prod_{i\in\{1,..,n\}}^{\{q\}} C_i \subset \left(\bigodot^{\{q\}}(C_n,..,C_1)\right)^{\infty}. \tag{4.29}$$

proof. We have

$$\begin{aligned}
\prod_{i\in\{1,..,n\}}^{\{q\}} C_i. &\overset{def}{=} \bigsqcup_{\substack{i_{n-q}>..>i_1\\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} \left(C_{i_{n-q}}\sqcap ...\sqcap C_{i_1}\right) && \text{(i)}\\
&= \bigsqcup_{\substack{i_{n-q}>..>i_1\\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} \left(C_{i_{n-q}}\circ ...\circ C_{i_1}\right)^{\infty} && \text{(ii)}\\
&= \left(\bigsqcup_{\substack{i_{n-q}>..>i_1\\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} \left(C_{i_{n-q}}\circ ...\circ C_{i_1}\right)^{\infty}\right)^{\infty} && \text{(iii)}\\
&\subset \left(\bigsqcup_{\substack{i_{n-q}>..>i_1\\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} C_{i_{n-q}}\circ ...\circ C_{i_1}\right)^{\infty} && \text{(iv)}\\
&\subset \left(\bigodot^{\{q\}}(C_n,..,C_1)\right)^{\infty} && \text{(v)}
\end{aligned} \tag{4.30}$$

For (i) see the definition of the repeated relaxed intersection. Line (ii) is a consequence of the theorem 16. Line (iii) is a consequence of the theorem 17. In line (iv), we only have inclusion since the theorem of unique repetition (theorem 15) doesn't work for the $\sqcup$ operator as seen in remark 13. The last line (iv) uses the definition of the relaxed composition. ■

Proposition 20 *Given two contractors C_1 and C_2 then if C_1 is monotonic then*

$$C_1\circ C_2 \subset C_1\cap C_2. \tag{4.31}$$

proof. Consider $[\mathbf{x}]\in\mathbb{IR}^m$. Since $C_2([\mathbf{x}])\subset[\mathbf{x}]$ and C_1 is monotonic

$$(C_1\circ C_2)([\mathbf{x}])\subset C_1([\mathbf{x}]). \tag{4.32}$$

On the other hand C_1 is a contractor so

$$(C_1 \circ C_2)([\mathbf{x}]) \subset C_2([\mathbf{x}]). \tag{4.33}$$

Thus

$$(C_1 \circ C_2)([\mathbf{x}]) \subset C_1([\mathbf{x}]) \cap C_2([\mathbf{x}]). \tag{4.34}$$

■

Proposition 21 *If the contractors $C_1, .., C_n$ are monotonic then*

$$\overset{\{q\}}{\bigodot}(C_n, .., C_1) \subset \overset{\{q\}}{\bigcap_{i\in\{1,..,n\}}} C_i. \tag{4.35}$$

proof. Given two contractors C_1 and C_2 then if C_1 is monotonic then

$$C_1 \circ C_2 \subset C_1 \cap C_2. \text{ (see proposition 20)} \tag{4.36}$$

This property can be generalized (recursively) to any number of contractors. As such

$$\bigsqcup_{\substack{i_{n-q}>..>i_1 \\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} C_{i_{n-q}} \circ ... \circ C_{i_1} \subset \bigsqcup_{\substack{i_{n-q}>..>i_1 \\ \{i_1,..,i_{n-q}\}\subset\{1,..,n\}}} C_{i_{n-q}} \cap ... \cap C_{i_1} = \overset{\{q\}}{\bigcap_{i\in\{1,..,n\}}} C_i. \tag{4.37}$$

■

This last proposition proves that relaxed composition is more efficient than relaxed intersection of contractors. However, relaxed composition requires more computations than the relaxed intersection . An $O(n^2)$ algorithm for the relaxed composition is presented in subsection 5.1.3. An $O(n\log(n))$ algorithm for the q-relaxed intersection of boxes (here boxes contracted by the contractors) is presented in subsection 3.6.2)

4.5 Contractors composition

Composition and repetition of contractors can be beneficial to the process of contraction. This phenomenon, also called constraint propagation, allow sometimes to solve CSPs without bisecting the search space.

Example 25 *In Figure 4.6 we make successive composition of the line contractors C_1 and C_2* i.e. *$C_1 \circ C_2 \circ C_1 \circ ...([\mathbf{x}]) = (C_1 \circ C_2)^n([\mathbf{x}])$. After many iterations, we converge to the intersection of the two lines which actually corresponds to set$(C_1 \circ C_2)$ in this case.*

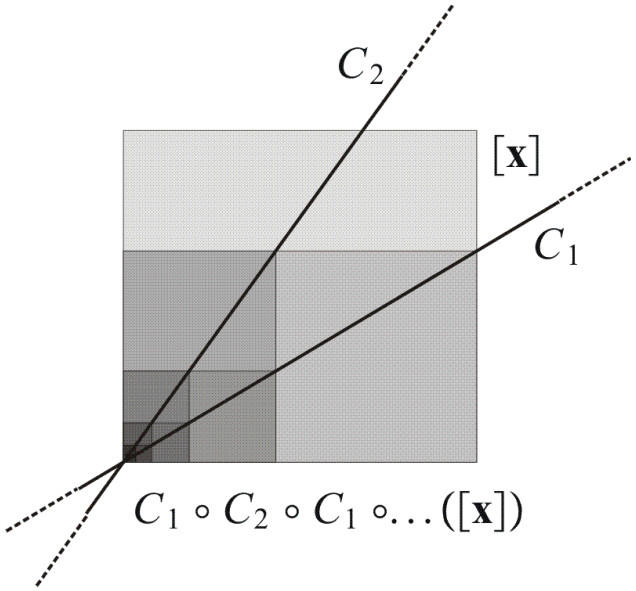

Figure 4.6: Composition and repetition of contractors

4.6 Theorems on minimality of contractors

4.6.1 Introduction

It is useful to have minimal contractors to speed up contractor computations. There are many methods to obtain minimal contractors in specific situations. As an example, [Chabert and Jaulin, 2009b] presents a method to construct an optimal contractor for any monotonic continuous functions. This section presents two theorems allowing to construct minimal contractors from other minimal contractors through union and/or transformation (such as translation, homothetie...) of contractors. The application of those two theorems is to construct complex minimal contractors having geometrical properties such as symmetry or repetition and is presented in section 4.7.

4.6.2 Definition

If $\mathbb{A}$ is a subset of $\mathbb{R}^n$ denote by $[\mathbb{A}]$ the smallest, with respect to the norm $\subset$, box of $\mathbb{R}^n$ containing this set *i.e.*

$$[\mathbb{A}] = \min_{\subset}([\mathbf{x}] \in \mathbb{IR}^n, \mathbb{A} \subset [\mathbf{x}]). \tag{4.38}$$

Definition 18 *A contractor C is minimal if it returns the smallest box enclosing all the insensible points of the contractor ($set(C)$)* i.e.

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, C([\mathbf{x}]) = [set(C) \cap [\mathbf{x}]]. \tag{4.39}$$

4.6.3 Union of two minimal contractors

One of the contributions to the PhD thesis is a theorem claiming that the union of two minimal contractors is also minimal. Before proving the theorem, there are several properties which have to be introduced.

Lemma 22 *The operator* [] *is idempotent* i.e. *for* $\mathbb{A} \subset \mathbb{R}^n$

$$[[\mathbb{A}]] = [\mathbb{A}]. \tag{4.40}$$

Lemma 23 *The operator* [] *is monotonic* i.e. *for* $\mathbb{A}, \mathbb{B} \subset \mathbb{R}^n$

$$\mathbb{A} \subset \mathbb{B} \Rightarrow [\mathbb{A}] \subset [\mathbb{B}]\,.$$

Lemma 24 *Given two sets* $\mathbb{A}, \mathbb{B} \subset \mathbb{R}^n$ *we have*

$$[\mathbb{A} \cup \mathbb{B}] = [[\mathbb{A}] \cup [\mathbb{B}]]. \tag{4.41}$$

proof. We have

$$\begin{array}{l} [\mathbb{A}] \subset [\mathbb{A} \cup \mathbb{B}] \\ [\mathbb{B}] \subset [\mathbb{A} \cup \mathbb{B}]. \end{array}$$

Since the union is monotonic we have

$$[\mathbb{A}] \cup [\mathbb{B}] \subset [A \cup \mathbb{B}], \tag{4.42}$$

and since [] is monotonic and idempotent we have

$$[[\mathbb{A}] \cup [\mathbb{B}]] \subset [[\mathbb{A} \cup \mathbb{B}]] = [\mathbb{A} \cup \mathbb{B}]. \tag{4.43}$$

In other hand

$$\begin{array}{l} \mathbb{A} \subset [\mathbb{A}] \\ \mathbb{B} \subset [\mathbb{B}]. \end{array} \tag{4.44}$$

as a consequence since $\cup$ is monotonic

$$\mathbb{A} \cup \mathbb{B} \subset [\mathbb{A}] \cup [\mathbb{B}], \tag{4.45}$$

and since [] is monotonic we have

$$[\mathbb{A} \cup \mathbb{B}] \subset [[\mathbb{A}] \cup [\mathbb{B}]]\,. \tag{4.46}$$

■

Lemma 25 *For two contractors C_1, C_2*

$$set(C_1 \sqcup C_2) = set(C_1) \cup set(C_2). \tag{4.47}$$

proof. Consider $\mathbf{x} \in \mathbb{R}$

$$\begin{array}{rcl} \mathbf{x} \in set(C_1 \sqcup C_2) & \Leftrightarrow & C_1 \sqcup C_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \\ & \Leftrightarrow & [C_1(\{\mathbf{x}\}) \cup C_2(\{\mathbf{x}\})] = \{\mathbf{x}\} \\ & \Leftrightarrow & C_1(\{\mathbf{x}\}) \cup C_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \\ & \Leftrightarrow & C_1(\{\mathbf{x}\}) = \{\mathbf{x}\} \text{ or } C_2(\{\mathbf{x}\}) = \{\mathbf{x}\} \\ & \Leftrightarrow & \mathbf{x} \in set(C_1) \cup set(C_2). \end{array}$$

■

Theorem 26 *The union of two minimal contractors C_1, C_2 is minimal*

proof. Consider C_1 and C_2 minimal contractors. We prove that $(C_1 \cup C_2)([\mathbf{x}]) = [set(C_1 \sqcup C_2) \cap [\mathbf{x}]]$. We have

$$(C_1 \sqcup C_2)([\mathbf{x}]) \overset{def}{=} [C_1([\mathbf{x}]) \cup C_2([\mathbf{x}])]. \tag{4.48}$$

Since C_1 and C_2 are minimal

$$(C_1 \sqcup C_2)([\mathbf{x}]) = [[set(C_1) \cap [\mathbf{x}]] \cup [set(C_2) \cap [\mathbf{x}])]]. \tag{4.49}$$

Using lemma 24 and lemma 25 we find that

$$\begin{array}{rl} (C_1 \sqcup C_2)([\mathbf{x}]) & = [set(C_1) \cap [\mathbf{x}] \cup set(C_2) \cap [\mathbf{x}])] \\ & = [(set(C_1) \cup set(C_2)) \cap [\mathbf{x}]]. \\ & = [(set(C_1 \sqcup C_2)) \cap [\mathbf{x}]]. \end{array} \tag{4.50}$$

■

4.6.4 Transformation of minimal contractors

This subsection introduces another contribution to this PhD thesis which is the concept of transformation of contractors (such as translation, homothetie...). The purpose is to define a set of transformations which preserve the minimality of a contractor. Such transformation allow to construct new minimal contractors from other minimal contractors.

Definition 19 *Consider $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ a continuous bijective function, C a contractor. Consider the transform T such as*

$$T(\mathbf{f}, C) = [\mathbf{f}] \circ C \circ [\mathbf{f}^{-1}]. \tag{4.51}$$

this transform is called the $\mathbf{f}$-transform of the contractor C.

Remark 14 *$T(\mathbf{f}, C)$ is not necessary a contractor.*

Proposition 27 *If $\forall[\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{f}]([\mathbf{x}]) = \mathbf{f}([\mathbf{x}])$ and $[\mathbf{f}^{-1}]([\mathbf{x}]) = \mathbf{f}^{-1}([\mathbf{x}])$ then $T(\mathbf{f}, C)$ is a contractor and $set(T(\mathbf{f}, C)) = \mathbf{f}(set(C))$. In that case it is possible to simplify the definition of the transform by putting*

$$T(\mathbf{f}, C) = \mathbf{f} \circ C \circ \mathbf{f}^{-1}. \tag{4.52}$$

proof. Contractance

$$\begin{array}{rcl} \forall[\mathbf{x}] \in \mathbb{IR}^n, C([\mathbf{x}]) \subset [\mathbf{x}] & \Rightarrow & C(\mathbf{f}^{-1}([\mathbf{x}])) \subset \mathbf{f}^{-1}([\mathbf{x}]) \\ & \Rightarrow & C(\mathbf{f}^{-1}([\mathbf{x}])) \subset \mathbf{f}^{-1}([\mathbf{x}]) \\ & \Rightarrow & \mathbf{f}(C(\mathbf{f}^{-1}([\mathbf{x}]))) \subset \mathbf{f}(\mathbf{f}^{-1}([\mathbf{x}])) \\ & \Rightarrow & T(\mathbf{f}, C)([\mathbf{x}]) \subset [\mathbf{x}]. \end{array} \tag{4.53}$$

Consistency

$$\begin{array}{rcl} \mathbf{x} \in [\mathbf{x}], T(f, C)(\{\mathbf{x}\}) = \{\mathbf{x}\} & \Rightarrow & \mathbf{f} \circ C \circ \mathbf{f}^{-1}(\{\mathbf{x}\}) = \{\mathbf{x}\} \\ & \Rightarrow & C(\mathbf{f}^{-1}(\{\mathbf{x}\})) = \mathbf{f}^{-1}(\{\mathbf{x}\}) \\ & \Rightarrow & \mathbf{f}^{-1}(\{\mathbf{x}\}) \in C(\mathbf{f}^{-1}([\mathbf{x}])) \\ & \Rightarrow & \{\mathbf{x}\} \in \mathbf{f}(C(\mathbf{f}^{-1}([\mathbf{x}]))) \\ & \Rightarrow & \mathbf{x} \in T(f, C)([\mathbf{x}]). \end{array} \tag{4.54}$$

Convergence

$$\begin{array}{rcl} \mathbf{x} \in \mathbb{R}^n, T(f, C)(\{\mathbf{x}\}) = \emptyset & \Rightarrow & C(\mathbf{f}^{-1}(\{\mathbf{x}\})) = \emptyset \\ & \Rightarrow & \exists \varepsilon > 0, \forall[\mathbf{y}] \subset B(\mathbf{f}^{-1}(\mathbf{x}), \varepsilon), C([\mathbf{y}]) = \emptyset \\ & \Rightarrow & \exists \alpha > 0, B(\mathbf{x}, \alpha) \subset \mathbf{f}(B(\mathbf{f}^{-1}(\mathbf{x}), \varepsilon) \text{ (since } \mathbf{f} \text{ is continuous)} \\ & \Rightarrow & \forall[\mathbf{x}] \subset B(\mathbf{x}, \alpha), \mathbf{f}^{-1}([\mathbf{x}]) \subset B(\mathbf{f}^{-1}(\mathbf{x}), \varepsilon) \Rightarrow C(\mathbf{f}^{-1}([\mathbf{x}])) = \emptyset \\ & \Rightarrow & \mathbf{f}(C(\mathbf{f}^{-1}([\mathbf{x}]))) = \emptyset \\ & \Rightarrow & T(f, C)([\mathbf{x}]) = \emptyset. \end{array} \tag{4.55}$$

$T(f, C)$ contractor set is $\mathbf{f}(set(C))$

$$\begin{array}{rcl} \mathbf{x} \in \mathbb{R}^n, T(f, C)(\{\mathbf{x}\}) = \{\mathbf{x}\} & \Leftrightarrow & \mathbf{f} \circ C \circ \mathbf{f}^{-1}(\{\mathbf{x}\}) = \{\mathbf{x}\} \\ & \Leftrightarrow & C(\mathbf{f}^{-1}(\{\mathbf{x}\})) = \mathbf{f}^{-1}(\{\mathbf{x}\}) \\ & \Leftrightarrow & \mathbf{f}^{-1}(\{\mathbf{x}\}) \in set(C) \\ & \Leftrightarrow & \{\mathbf{x}\} \in \mathbf{f}(set(C)). \end{array} \tag{4.56}$$

■

Remark 15 *If $[\mathbf{f}]([\mathbf{x}]) = \mathbf{f}([\mathbf{x}])$ and $[\mathbf{f}^{-1}]([\mathbf{x}]) = \mathbf{f}^{-1}([\mathbf{x}])$ then $[\mathbf{f}] \circ [\mathbf{f}^{-1}] = [I]$. This means that $[\mathbf{f}]$ is not inducing any* wrapping effect *(see section 3.5). In the case of 2D contractors, central symmetry, some axial symmetries, translation, homothetic transformation fall into that category*

Example 26 *An example of OY axis axial symmetry transformation is represented in Figure 4.7. The corresponding transformation function is*

$$\begin{array}{c} \mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2 \\ (x, y) \mapsto (-x, y). \end{array} \tag{4.57}$$

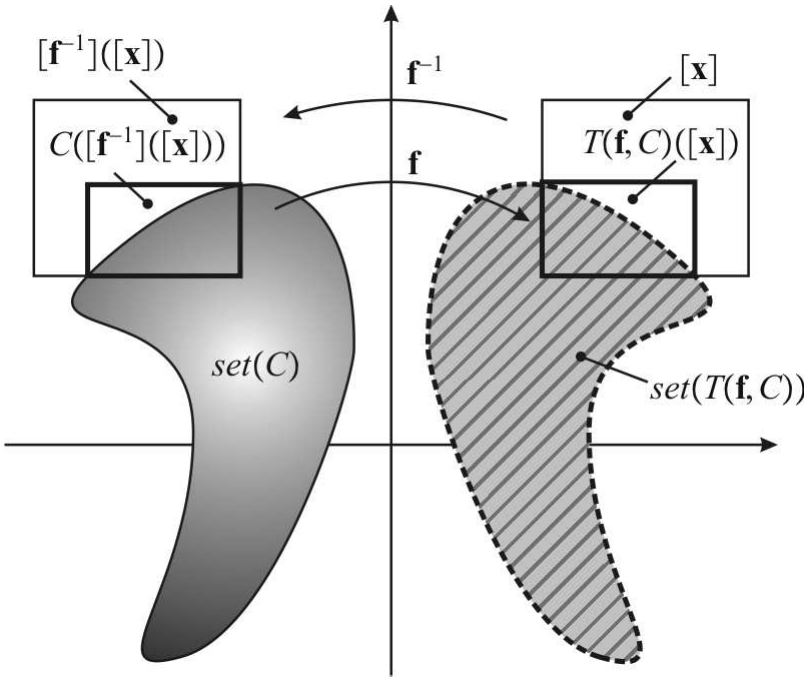

Figure 4.7: Creating the symmetric contractor

Lemma 28 *If* $\forall[\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{f}]([\mathbf{x}]) = \mathbf{f}([\mathbf{x}])$ *and* $[\mathbf{f}^{-1}]([\mathbf{x}]) = \mathbf{f}^{-1}([\mathbf{x}])$ *then given two sets* $\mathbb{A}, \mathbb{B} \subset \mathbb{R}^n$

$$[\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})] = \mathbf{f}([\mathbb{A} \cap \mathbb{B}]). \tag{4.58}$$

proof. We first prove $[\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})] \subset \mathbf{f}([\mathbb{A} \cap \mathbb{B}]$ then we prove the other inclusion. We have

$$\mathbb{A} \cap \mathbb{B} \subset [\mathbb{A} \cap \mathbb{B}]\,. \tag{4.59}$$

Thus

$$\mathbf{f}(\mathbb{A} \cap \mathbb{B}) \subset \mathbf{f}(\,[\mathbb{A} \cap \mathbb{B}]). \tag{4.60}$$

Since $\mathbf{f}$ is bijective

$$\mathbf{f}(\mathbb{A} \cap \mathbb{B}) = \mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B}). \tag{4.61}$$

Thus we prove

$$\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B}) \subset \mathbf{f}([\mathbb{A} \cap \mathbb{B}]. \tag{4.62}$$

Thus

$$[\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})] \subset \mathbf{f}([\mathbb{A} \cap \mathbb{B}]. \tag{4.63}$$

We use this last result to prove the other inclusion. We have

$$\left[\mathbf{f}^{-1} \circ \mathbf{f}(\mathbb{A}) \cap \mathbf{f}^{-1} \circ \mathbf{f}(\mathbb{B})\right] \subset \mathbf{f}^{-1}([\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})]). \tag{4.64}$$

Thus

$$[\mathbb{A} \cap \mathbb{B}] \subset \mathbf{f}^{-1}([\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})]). \tag{4.65}$$

Finally

$$\mathbf{f}([\mathbb{A} \cap \mathbb{B}]) \subset [\mathbf{f}(\mathbb{A}) \cap \mathbf{f}(\mathbb{B})]\,. \tag{4.66}$$

■

Theorem 29 *If* $\forall [\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{f}]([\mathbf{x}]) = \mathbf{f}([\mathbf{x}])$ *and* $[\mathbf{f}^{-1}]([\mathbf{x}]) = \mathbf{f}^{-1}([\mathbf{x}])$ *then if* C *is minimal then* $T(\mathbf{f}, C)$ *is minimal.*

proof. Since C is minimal

$$\begin{array}{rcl} T(\mathbf{f}, C)([\mathbf{x}]) & = & \mathbf{f}(C(\mathbf{f}^{-1}([\mathbf{x}]))) \\ & = & \mathbf{f}([set(C) \cap \mathbf{f}^{-1}([\mathbf{x}])]). \end{array} \tag{4.67}$$

Using lemma 28 we conclude that $T(\mathbf{f}, C)([\mathbf{x}])$ is minimal since

$$\begin{array}{rcl} T(\mathbf{f}, C)([\mathbf{x}]) & = & [\mathbf{f}(set(C)) \cap \mathbf{f} \circ \mathbf{f}^{-1}([\mathbf{x}])] \\ & = & [\mathbf{f}(set(C)) \cap [\mathbf{x}]]. \end{array} \tag{4.68}$$

■

4.7 Constructing minimal contractors using geometrical properties of their set

4.7.1 Introduction

Sometimes the contractors have some geometrical properties (symmetry, repeating patterns...) which can be used to simplify their implementation. In fact, theorem 29 ensures

that some particular transformations of contractors conserve their minimality. Besides, theorem 26 ensures that one can construct a minimal contractor as a union of minimal contractors. Thus one can construct an infinite number of complex minimal contractors based on simple initial minimal contractors called *core* contractors. New contractors are obtained from the *core* contractors through transformations such as symmetries, homothecies and translations and the union of those contractors defines the new contractor. Figure 4.8 shows the construction of a *snow* contractor. The sub-figures (a) and (e) represents the core contractors sets. Those contractors have to be minimal. The first part of the *flake* represented in sub-figures (d) is obtained using the core contractor in sub-figure (a) through symmetry (sub-figure (b)), then translation (sub-figure (c)) then a serie of different symmetries. The *flake* (sub-figure (g)) is obtained using the union of the parts obtained from both *core* contractors (sub-figure (d) and sub-figure (f)). Finally, the *snow* contractor is obtained by translating and resizing the *flake* contractor (homothetic transformations). The *snow* contractor is minimal.

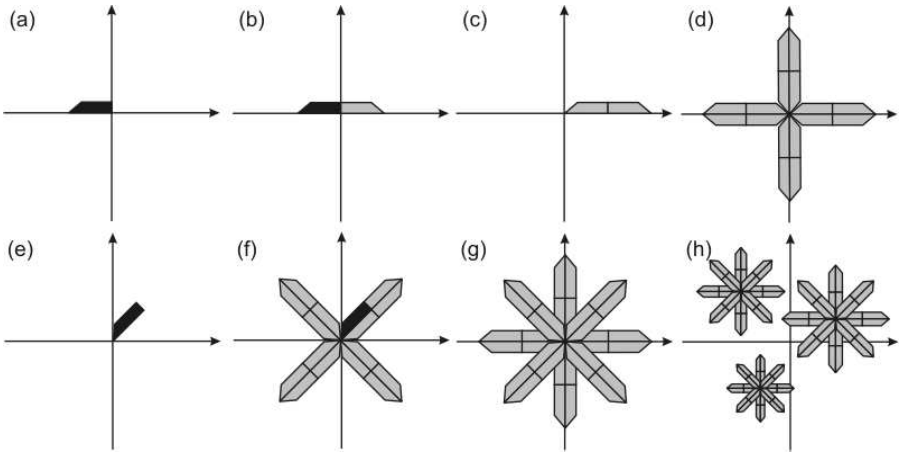

Figure 4.8: Snow contractor construction from two core contractors

4.7.2 Constructing the sinus contractor

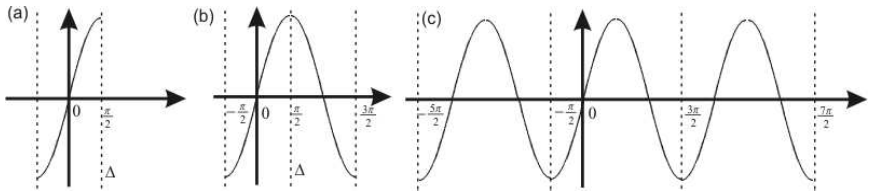

Figure 4.9: Construction of the sinus contractor using its geometrical properties

A more concrete application is the construction of the sinus contractor using its symmetry properties is represented in Figure 4.9. Consider a fragment of the sinus function defined on the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ (see sub-figure (a)) . A minimal contractor $C_{[-\frac{\pi}{2},\frac{\pi}{2}]}$ can be defined on this interval since the function is monotonic [Chabert and Jaulin, 2009b]. The contractor $C_{[-\frac{\pi}{2},\frac{\pi}{2}]}$ is the core contractor which will be used to construct the sinus contractor denoted C_{sin} corresponding to the sinus function defined on $\mathbb{R}$. First we use the fact that on the period $[-\frac{\pi}{2}, \frac{3\pi}{2}]$, the sinus function is symmetric with respect to the axis Δ defined by the equation $x = \frac{\pi}{2}$ (as seen in sub-figure (b)). Next we use "translation by $2k\pi, k \in \mathbb{Z}$" functions to construct C_{sin}.

Consider the contractor $C_{[\frac{\pi}{2},\frac{3\pi}{2}]}$ corresponding to the part of sinus defined on the interval $[\frac{\pi}{2}, \frac{3\pi}{2}]$. Consider the symmetry transformation

$$\begin{array}{l} \mathbf{f}_{\Delta-symmetry} : \mathbb{R}^2 \to \mathbb{R}^2 \\ (x, y) \mapsto (-x + \pi, y), \end{array} \tag{4.69}$$

Denote by $C_{[\frac{\pi}{2},\frac{3\pi}{2}]}$ the symmetric of the contractor $C_{[-\frac{\pi}{2},\frac{\pi}{2}]}$

$$C_{[\frac{\pi}{2},\frac{3\pi}{2}]} = T(\mathbf{f}_{\Delta-symmetry}, C_{[-\frac{\pi}{2},\frac{\pi}{2}]}). \tag{4.70}$$

The contractor $C_{[-\frac{\pi}{2},\frac{3\pi}{2}]}$ corresponding to the part of sinus defined on the whole period $[-\frac{\pi}{2}, \frac{3\pi}{2}]$ is the union of the minimal contractors $C_{[-\frac{\pi}{2},\frac{\pi}{2}]}$ and $C_{[\frac{\pi}{2},\frac{3\pi}{2}]}$ and thus it is minimal too.

$$C_{[-\frac{\pi}{2},\frac{3\pi}{2}]} = C_{[-\frac{\pi}{2},\frac{\pi}{2}]} \sqcup C_{[\frac{\pi}{2},\frac{3\pi}{2}]}. \tag{4.71}$$

Sinus function is 2π periodic. Consider the following $2k\pi$-translation function

$$\begin{array}{l} \mathbf{f}_{2k\pi} : \mathbb{R}^2 \to \mathbb{R}^2 \\ (x, y) \mapsto (x + 2k\pi, y). \end{array} \tag{4.72}$$

Denote by $C_{[-\frac{\pi}{2}+2k\pi,\frac{3\pi}{2}+2k\pi]}$ the contractor corresponding to the part of sinus defined on the period $[-\frac{\pi}{2} + 2k\pi, \frac{3\pi}{2} + 2k\pi]$. One have

$$C_{[-\frac{\pi}{2}+2k\pi,\frac{3\pi}{2}+2k\pi]} = T(\mathbf{f}_{2k\pi}, C_{[-\frac{\pi}{2},\frac{3\pi}{2}]}). \tag{4.73}$$

The sinus contractor C_{sin} which corresponds to the sinus function defined on whole $\mathbb{R}$ is then defined by

$$C_{sin} = \bigsqcup_{k\in\mathbb{Z}} C_{[-\frac{\pi}{2}+2k\pi,\frac{3\pi}{2}+2k\pi]}. \tag{4.74}$$

Finally, the sinus contractor can be expressed as unions and transformations of the core contractor $C_{[-\frac{\pi}{2},\frac{\pi}{2}]}$ using this compact formula

$$C_{sin} = \bigsqcup_{k\in\mathbb{Z}} T(\mathbf{f}_{2k\pi}, T(\mathbf{f}_{\Delta-symmetry}, C_{[-\frac{\pi}{2},\frac{\pi}{2}]}) \sqcup C_{[-\frac{\pi}{2},\frac{\pi}{2}]}). \tag{4.75}$$

Proposition. C_{sin} is the sinus contractor and it is minimal by construction.

proof. Direct result of the theorems 26 and 29. ■

4.7.3 Constructing the argument contractor

Introduction

The argument contractor C_{arg} is the contractor related to the constraint linking the angle θ and the argument of a vector $\vec{u}(x, y)$

$$\theta = \arg(\vec{u}). \tag{4.76}$$

The argument contractor is important in robotics since many sensors give angle measurements. The camera can give the view angle of some objects, some acoustic or electromagnetic sensors can point the direction of a source of acoustic/electromagnetic waves such as beacons. The argument contractor is the basic contractor in algorithms such as SLAM (Simultaneous Localization And Mapping) with punctual landmarks [Le Bars et al., 2010] [Joly, 2010] [Porta, 2005].

Construction using symmetries and translations

The argument contractor C_{arg} has the following geometrical properties

- $\vec{y}$ axis symmetry

$$\arg(x, y) = \pi - \arg(-x, y). \tag{4.77}$$

- $\vec{x}$ axis symmetry

$$\arg(x, y) = -\arg(x, -y). \tag{4.78}$$

- $2k\pi$ periodicity

the associated transformation functions are respectively

$$\begin{array}{l} \mathbf{f}_y : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ (x, y, \theta) \mapsto (-x, y, \pi - \theta) \\ \mathbf{f}_x : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ (x, y, \theta) \mapsto (x, -y, -\theta) \\ \mathbf{f}_{2k\pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ (x, y, \theta) \mapsto (x, y, \theta + 2k\pi). \end{array} \tag{4.79}$$

Consider the core contractor C_{arg_root} corresponding to the part of arg function defined on $\mathbb{R}^+ \times \mathbb{R}^+ \times [0, \frac{\pi}{2}]$ *i.e.* the contractor associated to the following set

$$set(C_{arg\_root}) = \{(x, y, \theta) \in \mathbb{R}^3, \arg(x, y) \in [0, \frac{\pi}{2}]\}. \tag{4.80}$$

The $\vec{y}$ axis symmetry extends the set associated to the of the resulting contractor to $\mathbb{R} \times \mathbb{R}^+ \times [0, \pi]$.

The $\vec{x}$ axis symmetry extends the set associated to the of the resulting contractor to $\mathbb{R} \times \mathbb{R} \times [-\pi, \pi]$.

Finally the $2k\pi$ periodicity extends the set associated to the of the resulting contractor to $\mathbb{R}^3$ by doing translations of the contractor defined on $\mathbb{R} \times \mathbb{R} \times [-\pi, \pi]$.

Thus what we obtain is the argument $C_{\arg}$.defined by the following formula

$$\begin{aligned} C_{\arg} &= \bigsqcup_{k\in\mathbb{Z}} T(\mathbf{f}_{2k\pi}, C_3) \\ C_3 &= T(\mathbf{f}_x, C_2) \sqcup C_2 \\ C_2 &= T(\mathbf{f}_y, C_{\arg\_root}) \sqcup C_{\arg\_root}, \end{aligned} \tag{4.81}$$

or in a more compact form

$$C_{\arg} = \bigsqcup_{k\in\mathbb{Z}} T(\mathbf{f}_{2k\pi}, T(\mathbf{f}_x, T(\mathbf{f}_y, C_{\arg\_root}) \sqcup C_{\arg\_root}) \sqcup T(\mathbf{f}_y, C_{\arg\_root}) \sqcup C_{\arg\_root}). \tag{4.82}$$

The contractor $C_{\arg}$ is minimal by construction.

The core contractor

The core of the contractor of the argument contractor C_{arg_root} is defined on $\mathbb{R}^+ \times \mathbb{R}^+ \times [0, \frac{\pi}{2}]$.Denote by $([x_c], [y_c], [\theta_c]) = C_{arg\_root}(([x], [y], [\theta]))$ we have

$$\begin{aligned} [x_c] &= \tfrac{[y]}{[\tan]([\theta])} \\ [y_c] &= [x] \cdot [\tan]\,([\theta]) \\ [\theta_c] &= [\arctan]\,(\tfrac{[y]}{[x]}), \end{aligned} \tag{4.83}$$

where [tan] and [arctan] are the inclusion functions of tan and arctan respectively. A representative case of contraction using C_{arg_root} is represented on Figure 4.10.

Remark 16 *Since C_{arg_root} is defined on the domain $\mathbb{R}^+ \times \mathbb{R}^+ \times [0^+, \frac{\pi}{2}]$, in the interval computations over that domain we suppose $\frac{1}{0} = +\infty$. Since $\frac{[y]}{[x]}$ is not defined for $([x], [y]) = (\{0\}, \{0\})$, we suppose that*

$$\forall \theta \in [0^+, \frac{\pi}{2}], C_{arg\_root}((\{0\}, \{0\}, [\theta])) = (\{0\}, \{0\}, [\theta]). \tag{4.84}$$

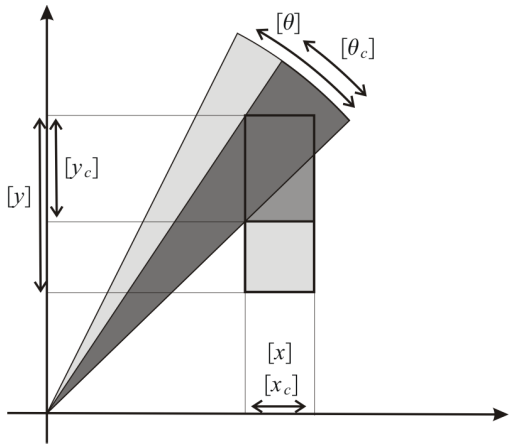

Figure 4.10: Example of contraction using the argument contractor where both the angle θ and the vector (x, y) are contracted

Remark 17 *The construction of the argument contractor using unions and transformations of the core contractor is useful to avoid too many special cases such as those in Remark 16 which would be numerous and redundant if the approach wasn't used. (Special cases are one of the most common source of program bugs).*

4.8 The image contractor

4.8.1 Introduction

There are cases where a set is hard to be modeled in terms of set of equations. Figure 4.11 shows an example of such a set. In robotics, this is the case of maps of unstructured environments. In fact, structured maps such as buildings are easily represented as a set of line segments or polynomials while it is complicated to model irregular environments like sea shore or road maps.

This section introduces a contractor C which set of insensitive points $set(C)$ is represented using a binary image. The new contractor is called the *image contractor* and is the second main contribution to the PhD thesis. For more simplicity, the *image contractor* is first defined for continuous 2D binary image. The *image contractor* is meant to be used in for example localization algorithms (to represent maps). The *image contractor* needs then to be implemented on a computer. We define the discrete form of the *image contractor* (defined for 2D discrete images) in subsection 4.8.5. The discrete *image contractor* only contracts discrete boxes (natural number boxes). Subsection 4.8.6 explains how to use the discrete *image contractor* to contract real number boxes. The case where the image

Figure 4.11: Example of a set difficult to model with equations

is higher dimension can be easily derived from the 2D case.

4.8.2 Continuous image contractor

In this part, the *image contractor* is defined using a continuous binary image in $\mathbb{R}^2$. Consider a continuous binary image defined by

$$\mathbf{f} : \mathbb{R}^2 \to \{0, 1\}. \tag{4.85}$$

We consider the 1-valued pixels as the pixels of interest and they are colored in black or gray. The 0-valued pixels are left white.

Remark 18 *Note that* $\mathbf{f}$ *is the characteristic function of the set* $\mathbb{S}$ *represented on the binary image.*

Definition 20 *The* image contractor C *associated to the image defined by the function* $\mathbf{f}$ *is the minimal contractor which set of insensitive points is*

$$set(C) = \{\mathbf{x} \in \mathbb{R}^2, \mathbf{f}(\mathbf{x}) = 1\}. \tag{4.86}$$

Figure 4.12 shows the action of the image contractor.

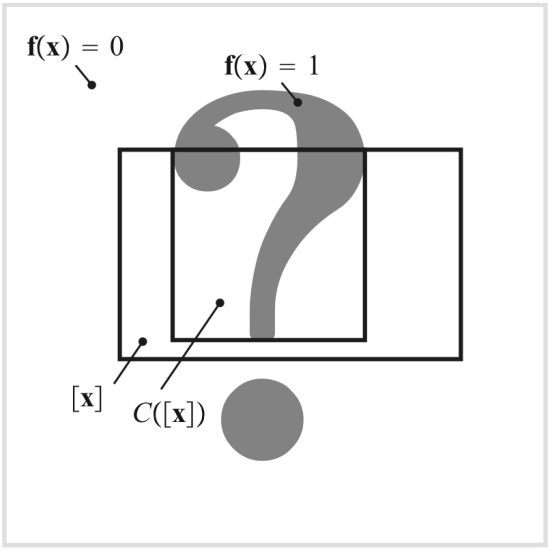

Figure 4.12: Example of contraction using the image contractor

4.8.3 Inclusion test

Consider the function ϕ

$$\begin{array}{l} \phi : \mathbb{IR}^2 \rightarrow \mathbb{R}, \\ [\mathbf{x}] \mapsto \int_{(x,y)\in[\mathbf{x}]} \mathbf{f}(x,y)dxdy. \end{array} \tag{4.87}$$

The function ϕ characterizes the quantity of 1-valued pixels in a box $[\mathbf{x}]$. As an example, if $\phi([\mathbf{x}]) = 0$ means that there are no 1-valued pixels in the box $[\mathbf{x}]$. As such the function ϕ, is used to build an inclusion function for the constraint associated to the set represented on the image.

Consider the function ψ

$$\begin{array}{l} \psi : \mathbb{R}^2 \rightarrow \mathbb{R} \\ \mathbf{x} \mapsto \int_0^{x_1} \int_0^{x_2} \mathbf{f}(x,y)dxdy. \end{array} \tag{4.88}$$

The ψ function associates for each pixel $\mathbf{x}$ the quantity of 1-valued pixels in the box $([0,x_1],[0,x_2])$. As such, the ψ function defines a continuous grayscale image which is called the *ψ-transform* of the binary image. The pixels $\mathbf{x} \in \mathbb{R}^2$ such as $\psi(\mathbf{x}) = 0$ are colored in white and the higher the value of $|\psi(\mathbf{x})|$, the darker is the color of the pixel. Figure 4.13 shows the *ψ-transform* (on the right) of a binary image (on the left). Note that the origin $(0,0)$ is the left upper corner of the image since that is the common convention for images.

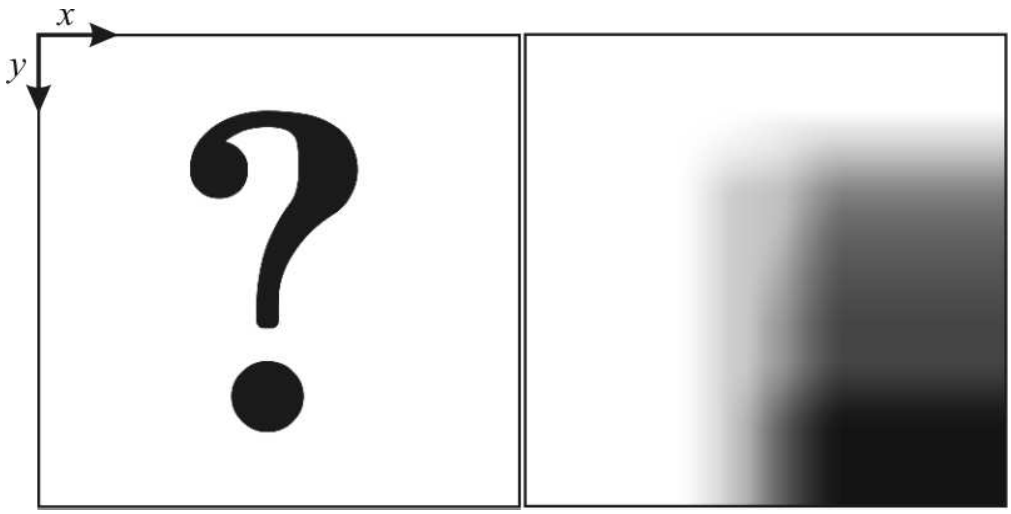

Figure 4.13: Example of a binary image and the corresponding ψ transform

The function ϕ can be obtained from ψ.

$$\phi([\mathbf{x}]) = \psi(x_1^+, x_2^+) - \psi(x_1^-, x_2^+) - \psi(x_1^+, x_2^-) + \psi(x_1^-, x_2^-). \tag{4.89}$$

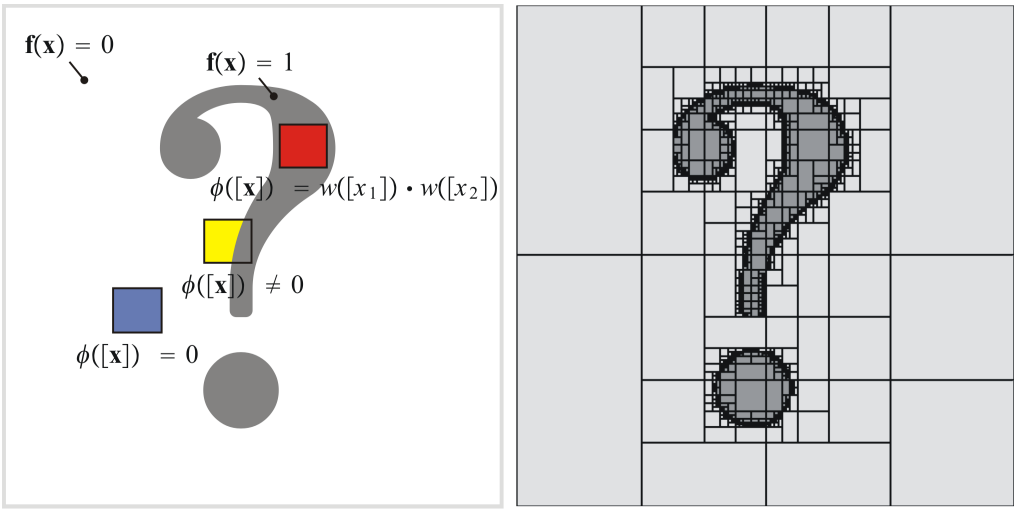

Figure 4.14: Defining an inclusion function using the ϕ function (on the left) and its application in the SIVIA algorithm (on the right)

In practice, the idea is to compute the *ψ-transform* only once and store the result in computers memory as a grayscale image for example. As a result ϕ can be evaluated almost instantly for every box. The function ϕ can be used to define the inclusion function necessary for the SIVIA (Set Inversion Via Interval Analysis) algorithm as seen in Figure 4.14. Considering a box $[\mathbf{x}] \subset \mathbb{R}^2$

- If $\phi([\mathbf{x}]) = 0$ then the box doesn't contain any 1-valued pixels *i.e.* $[\mathbf{x}]$ is outside of the set defined by the binary image.
- If $\phi([\mathbf{x}]) \neq 0$ then the box contains some 1-valued pixels *i.e.* $[\mathbf{x}]$ intersects with the set defined by the binary image.
- If $\phi([\mathbf{x}]) = w([x_1]) \cdot w([x_2])$ then all the points in the box are 1-valued pixels *i.e.* $[\mathbf{x}]$ is included in set defined by the binary image.

Figure 4.14 also shows the subpaving generated of SIVIA algorithm using image constraint inclusion function.

4.8.4 Contraction algorithm

Denote by C the image contractor. Consider $[\mathbf{x}] \in \mathbb{IR}^2$ and $C([\mathbf{x}]) = [\mathbf{y}]$. We have

$$\begin{array}{l} y_1^- = \max(x \in [x_1], \phi([x_1^-, x] \times [x_2]) = 0) \\ y_1^+ = \min(x \in [x_1], \phi([x, x_1^+] \times [x_2]) = 0) \\ y_2^- = \max(x \in [x_2], \phi([x_1] \times [x_2^-, x]) = 0) \\ y_2^+ = \min(x \in [x_2], \phi([x_1] \times [x, x_2^+]) = 0). \end{array} \tag{4.90}$$

The min and max can be computed using dichotomy which has logarithmic complexity. Figure 4.15 illustrates the dichotomy used to compute y_1^-. The box on which the dichotomy is applied is the box $[x_1^-, x] \times [x_2]$ where x is the unknown.

4.8.5 Discrete form of the image contractor

Binary images which can be manipulated on a computer usually are discrete and bounded. Consider then a discrete binary image with the width w and the height h. This image can be described by the following discrete function

$$\mathbf{f}_d : \{1, .., w\} \times \{1, .., h\} \rightarrow \{0, 1\}. \tag{4.91}$$

The definition of the discrete contractor $C_d([\mathbf{n}])$ is the same as in the continuous case except that the contracted boxes are natural number boxes in the image workspace ($[\mathbf{n}] \in \mathbb{IN}$). See Figure 4.16 (on the left) for an example of contraction using this contractor.

In discrete case the function ψ_d is defined as follows

$$\begin{array}{l} \psi_d : \{1, .., w\} \times \{1, .., h\} \rightarrow \mathbb{N} \\ \psi_d(\mathbf{n}) = \sum_{i=1}^{\min(n_1, w)} \sum_{j=1}^{\min(n_2, h)} \mathbf{f}_d(i, j). \end{array} \tag{4.92}$$

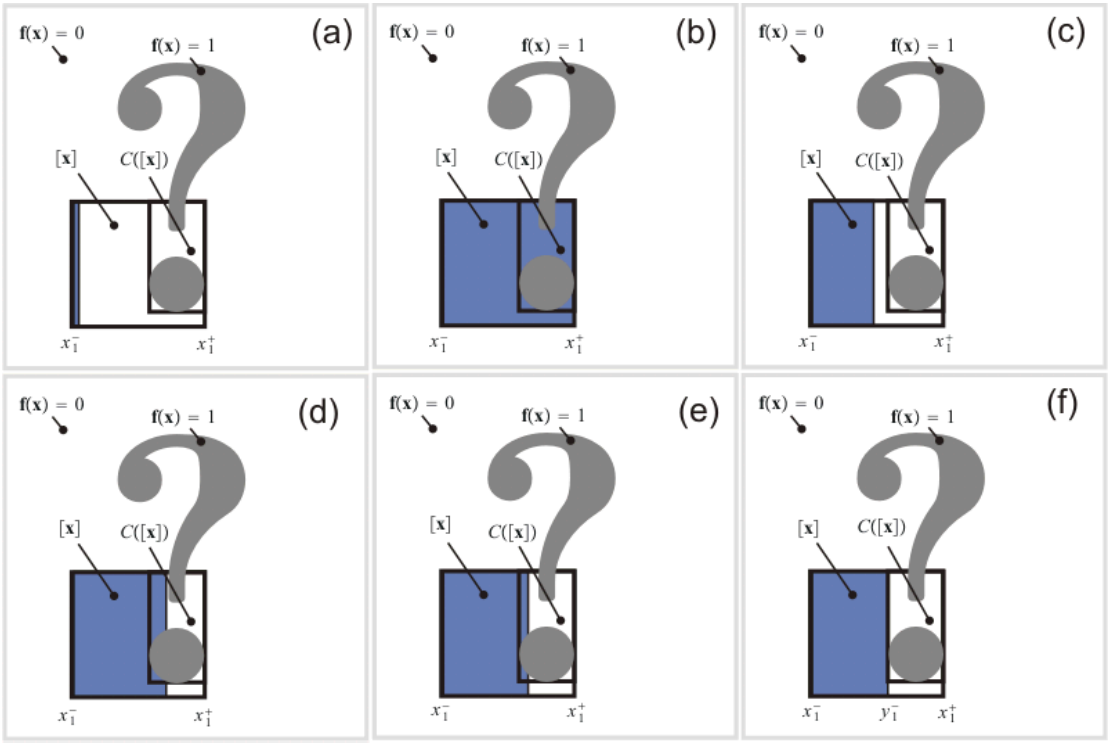

Figure 4.15: Computing one bound of $C([\mathbf{x}])$ using dichotomy

In discrete case, the function ϕ_d is defined by

$$\begin{aligned} &\phi_d : \mathbb{IN}^2 \to \mathbb{N} \\ &\phi_d([\mathbf{n}]) = \psi_d(n_1^+, n_2^+) - \psi_d(n_1^-, n_2^+) - \psi_d(n_1^+, n_2^-) + \psi_d(n_1^-, n_2^-). \end{aligned} \tag{4.93}$$

The contractor C_d can be constructed using dichotomy same as the continuous contractor defined in 4.8.4.

4.8.6 Using the discrete image contractor in continuous problems

Only the discrete image contractor can be implemented on a computer. However, the problems we solve require continuous ($\mathbb{R}^2$) contractors. This subsection presents a method to overcome this problem. Consider a discrete image and denote by R_d the image workspace. The binary image is actually used to represent a set $\mathbb{S}_d$ (the map in the case of localization) defined in a specific workspace R_S as seen in Figure 4.16. The set $\mathbb{S}_d$ has to be bounded (since the image is bounded). Denote by C the contractor which set of insensitive points is $set(C) = \mathbb{S}_d$. Denote by C_d the *image contractor* associated to the discrete image represented in the R_d workspace. The contractor C_d contracts in the R_d workspace

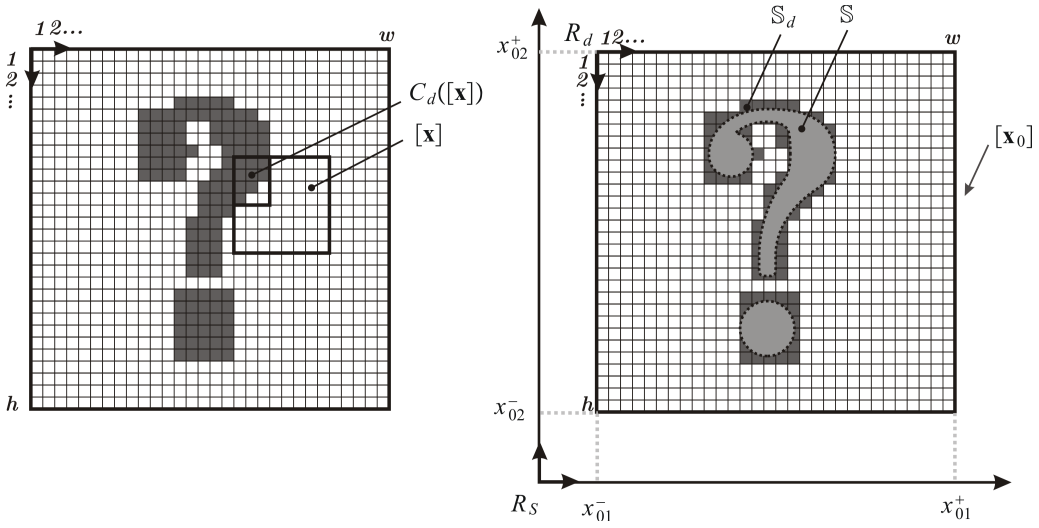

Figure 4.16: Example of contraction using a discrete image contractor (on the left) and how to use this new contractor to approximate a continuous image contractor (on the right)

and is defined as seen in subsection 4.8.5. The idea is to find an expression linking C and C_d. The key is to find the transformation function between the real workspace R_S and the discrete workspace R_d.

Remark 19 *The set $\mathbb{S}_d$ can be seen as the approximation of a usually unknown set $\mathbb{S}$ corresponding to the real constraint we search to characterize (In Figure 4.16 it is the question mark). It is like "*taking a picture*" of the set $\mathbb{S}$ using a digital camera. On the other hand, even if $\mathbb{S}$ is known, because the image contractor is fast, it is sometimes useful to transform complex and resource consuming contractors into image contractors. The first drawback is the loss of precision because of the approximation. The second drawback is the increase in storage space required for the ψ-*transform *which increases the more the precision of the approximation (*i.e. *the number of pixels) is high.*

Note that the workspace R_d is vertically flipped with regards to the workspace R_s because of image index convention. Consider a real box $[\mathbf{x}] \in \mathbb{IR}^2$ to be contracted in the R_S workspace. Consider the homothetic transformation $h : \mathbb{IR}^2 \to \mathbb{IR}^2$ such as $h([\mathbf{x}_0]) = [1, w] \times [1, h]$. Consider the box discretization function $d : \mathbb{IR}^2 \to \mathbb{IN}^2$ which for each box of $\mathbb{IR}^2$ returns the smallest natural number box enclosing the real number box. Finally, consider the flip function $v : \mathbb{IN}^2 \to \mathbb{IN}^2$ which vertically flips a natural box in the image workspace (The expression is $v([\mathbf{n}]) = ([n_1], h + 1 - [n_2]), [\mathbf{n}] \in \mathbb{IN}^2$). We have

$$C([\mathbf{x}]) = [\mathbf{x}] \cap h^{-1}(v^{-1}(C_d(v(d(h([\mathbf{x}])))))). \tag{4.94}$$

4.8.7 Application for the localization of a vehicle in the city without GPS

The example in Figure 4.17 illustrates a possible application of the *image contractor* when dealing with a map $\mathbb{M}$ of a road represented as a binary image. In the example a vehicle moves on city roads represented in black. It is possible to measure the speed v_k of the vehicle (by looking at the speed panel or hacking the CAN bus) and the orientation θ_k of the vehicle using a magnetic compass. The position (x_k, y_k) of the robot is characterized by discrete-time dynamic equations

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + v_k * \cos(\theta_k) * dt \\ y_k + v_k * \sin(\theta_k) * dt \end{pmatrix}.$$

The initial position is supposed to be known. Sub-Figure 4.17.(a) represents the solution of the equations above (the reconstructed trajectory) in the form of a serie of boxes. Since speed and orientation measurements are subject to noise, the position estimation error diverges. Sub-Figure 4.17.(b) represents the solution of the equations above considering an additional constraint $\forall k, (x_k, y_k) \in \mathbb{M}$ i.e. "the vehicle is on the road" constraint. This constraint is implemented on the computer using the *image contractor*. The position estimation error is bounded as seen in Sub-Figure 4.17.(b).

It is possible to solve more complex problems where initial position is unknown (global localization) and/or the compass measurements are replaced by gyroscope measurements (rotation speed) using this method.

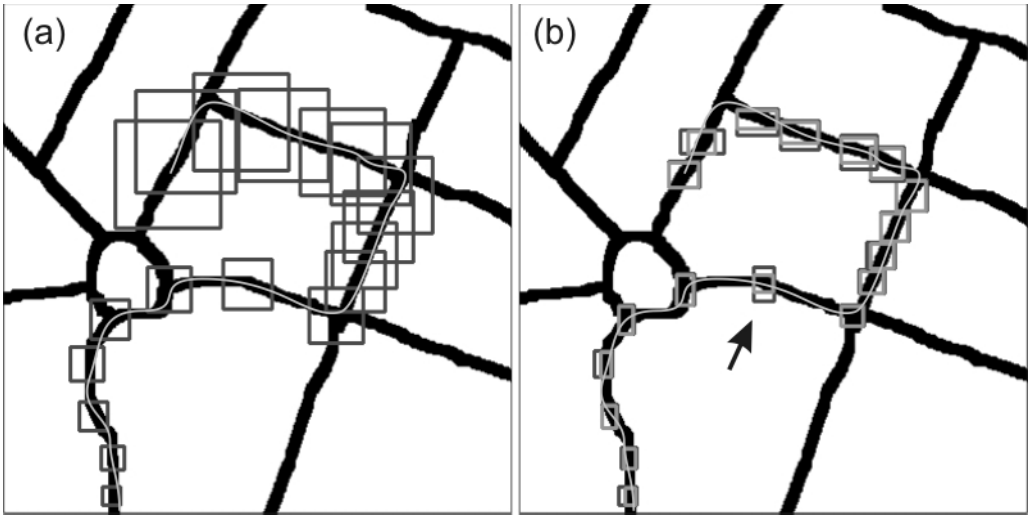

Figure 4.17: Localization in the city using odometry and compass only

4.8.8 Conclusion

The *image contractor* is a new way to represent constraints and contractors. The advantage is that the contractor formalism allows the *image contractor* to be used along classic contractors this extending the possibilities of contractor solvers. The advantage of the method is the high speed of execution which is also similar for all image contractors of same size. The main disadvantage is the storage space required for storing the ψ-*transform* which exponentially increases with the dimension of the constraint. There are then many prospects of research such as the compression of the ψ-*transform*.

Chapter 5

Relaxed CSP computer solvers

5.1 Implementation of set polynomials based solvers

5.1.1 Introduction

Section 2.5 introduced the concept of set polynomials and their use in representing the solution of a relaxed CSP. The coefficients of a set polynomial are continuous sets which cannot be directly represented on a computer. Contractors, as seen in 4, are algorithmic entities which can represent a particular set. Given a contractor, an appropriate algorithm (such as branch and prune algorithm) can be used to compute a subpaving enclosing the set associated to the contractor. The idea introduced in this chapter is to use contractors as coefficients of the polynomial instead of sets thus defining a *contractor polynomial*. Taking benefit of contractor arithmetics, it is possible to define the sort transform of a vector of contractors in the same way that in the case of sets. This chapter also introduces a recursive algorithm based on polynomial arithmetics to solve a relaxed CSP when the solution is punctual. The solution is represented in the form of a box polynomial. An example is provided to show the viability of the approach.

5.1.2 Contractor polynomials

A possible implementation for set polynomials defined in section 2.5 are polynomials with contractor valued coefficients also called *contractor polynomials*. The set of contractors has a lattice structure (see [Chabert and Jaulin, 2009a]).

- The order is the inclusion " $\subset$ " such as

$$C_1 \subset C_2 \Leftrightarrow \forall [\mathbf{x}], C_1([\mathbf{x}]) \subset C_2([\mathbf{x}]). \tag{5.1}$$

- The supremum is obtained through contractor union $\sqcup$ and the infinimum is obtained through the repeated intersection $\sqcap$
- The contractor set maximum is the identity contractor which is denoted $c^{\top}$ and the contractor set minimum is the contractor which returns an empty set for all boxes denoted by $c^{\perp}$

$$\forall [\mathbf{x}] \in \mathbb{IR}^m, C^{\top}([\mathbf{x}]) = [\mathbf{x}], C^{\perp}([\mathbf{x}]) = \emptyset.$$

It is thus possible to define *contractor polynomials* on this lattice as seen in subsection 2.5.4 taking benefit of all the properties defined in the subsection. As such, the sort transform is also defined. Consider the contractor vector $\mathbf{C} = (C_1, .., C_n)$. Consider the polynomial sort transform (see definition 6) of this vector

$$C^*(s) = \prod_{k=1}^{n} (C_k s + c^{\top}). \tag{5.2}$$

Using polynomial arithmetics, the polynomial $C^*(s)$ can be expanded into

$$\begin{aligned} C^*(s) &= \sum_{k=1}^{n} \left(\sum_{\mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K})=k} \left(\prod_{i \in \mathbb{K}} C_i \right) \right) s^k + C^{\top} && \text{(i)} \\ &= \sum_{k=1}^{n} \left(\bigsqcup_{\mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K})=k} \left(\prod_{i \in \mathbb{K}} C_i \right) \right) s^k + C^{\top} && \text{(ii)} \\ &= \sum_{k=1}^{n} \left(\prod_{i \in \{1,..,n\}}^{\{n-k\}} C_i \right) s^k + C^{\top}. && \text{(iii)} \end{aligned} \tag{5.3}$$

where $\prod_{i \in \{1,..,n\}}^{\{q\}} C_i$ is the repeated q-relaxed intersection of contractors (see definition (4.2)). Line (i) is the polynomial expansion. In line (ii), the "$*$" and "$+$" operations are replaced by "$\sqcap$" and "$\sqcup$"respectively. In line (iii), proposition 18 is used. Consider the relaxed CSP defined by the constraints associated to the contractors $\mathbf{C} = (C_1, .., C_n)$. Consider $X^*(s)$ the solution set polynomial of the relaxed CSP (see subsection 2.5.5)

$$X^*(s) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{5.4}$$

The contractor polynomial $C^*(s)$, which we can call the solution contractor polynomial, is a nested contractor polynomial where the k^{th} coefficient C_k^* represents the contractor associated to the k^{th} coefficient of the solution set polynomial $X^*(s)$. Using contractor based solvers such as the branch and prune algorithm it is possible to generate a subpaving enclosing each of the coefficients $\mathbb{S}_q, q \in \{0, .., n-1\}$. The following subsection proposes a another practical algorithm to obtain an approximation of the polynomial $C^*(s)$.

5.1.3 Computing the solution contractor polynomial

This subsection proposes three methods which enables to compute an approximation of the contractor polynomial $C^*(s)$ associated to the solution set polynomial of a relaxed CSP. Denote by $C^*_{(1)}(s)$ and $C^*_{(2)}(s)$ and $C^*_{(3)}(s)$ the polynomials which will be used to approximate the polynomial $C^*(s)$.

First approximation

The repeated intersection $\sqcap$ of contractors means that the intersection $\cap$ of contractors should normally be repeated an infinite number of times. In practice, the intersection is repeated a finite number of times. The first contractor polynomial $C^*_{(1)}(s) = \sum_{k=1}^{n} C^*_{(1)k} s^k + C^\top$ is defined such as

$$C^*_{(1)k} = \left(\bigcap_{i\in\{1,..,n\}}^{\{n-k\}} C_i \right)^p . \tag{5.5}$$

where $\bigcap_{i\in\{1,..,n\}}^{\{q\}} C_i$ is the q-relaxed intersection of contractors and p is usually chosen so that there is no notable difference between that contractor for p and $p-1$. The contractor polynomial $C^*_{(1)}(s)$ converges to the polynomial $C^*(s)$ when $p \to \infty$ (see proposition 18 in section 4.4).

Second approximation

There are other ways to approximate the contractor polynomial $C^*(s)$. The idea is to replace the repeated intersection "$\sqcap$" of contractors by the composition "$\circ$" of contractors. The set of contractors provided with the operations " $\circ$ " and " $\sqcup$ " is not a lattice. Despite the loss of lattice structure, it is still possible to keep the definition of the sort transform and polynomial arithmetics unchanged. Considering a vector of contractors $\mathbf{C} = (C_1, .., C_n)$, the corresponding polynomial sort transform is denoted

$$C^{(1*)}(s) = \prod_{i\in\{1,..,n\}} (C_i s + c^\top) = (C_n s + c^\top) \circ ... \circ (C_1 s + c^\top) = \sum_{k=1}^{n} C^{(1*)}_k s^k + C^\top. \tag{5.6}$$

Note that by expanding the $C^{(1*)}(s)$ polynomial we obtain

$$C^{(1*)}_k = \bigsqcup_{\substack{i_k>..>i_1 \\ \{i_1,..,i_k\}\subset\{1,..,n\}}} C^{(1*)}_{i_k} \circ ... \circ C^{(1*)}_{i_1} = \bigodot^{\{n-k\}} (C_n, .., C_1). \tag{5.7}$$

where $\overset{\{q\}}{\odot}(C_n,..,C_1)$ is the q-relaxed composition of contractors.

As seen in proposition 21 in section 4.4, the contractor $\overset{\{q\}}{\odot}(C_n,..,C_1)$ is a better contractor than the q-relaxed intersection of contractors $\overset{\{q\}}{\bigcap\limits_{k\in\{1,..,n\}}} C_k$. The next subsubsection introduces an algorithm used to compute of the polynomial $C^{(1*)}(s)$ and as a consequence, compute the relaxed composition of contractors.

This leads to the second possible approximation of $C^*(s)$ which is the contractor polynomial $C^*_{(2)}(s) = \sum_{k=1}^{n} C^*_{(2)k} s^k + C^\top$ defined by

$$C^*_{(2)k} = \left(\overset{\{n-k\}}{\odot} (C_n,..,C_1) \right)^p. \tag{5.8}$$

where p is usually chosen so that there is no notable difference between that contractor for p and $p-1$. The contractor polynomial $C^*_2(s)$ when $p \to \infty$ converges to a contractor polynomial which encloses the polynomial $C^*(s)$ (see proposition 19 in section 4.4).

Third approximation

In the two previous approximations, the coefficients of the polynomial $C^*_{(1)}(s)$ and $C^*_{(2)}(s)$ were not fully obtained using polynomial arithmetics. The question would be how represent the repetition necessary for convergence using polynomial arithmetics. The approach presented here is not very efficient in terms of computation complexity but provides an example of the use of polynomial arithmetics which could inspire future research on new algorithms. For the new algorithm, the idea is to consider the polynomial sort transform of the contractor vector $\mathbf{C}^{(p)} = \underbrace{(C_1,..,C_n,...,C_1,..,C_n)}_{p \textbf{ times}}$ denoted by

$$C^{(p*)}(s) = \prod_{k\in\{1,..,n\}} (C_k s + c^\top) = \sum_{k=1}^{p*n} C_k^{(p*)} s^i + C^\top. \tag{5.9}$$

$C^{(p*)}(s)$ is a polynomial of the degree $p*n$. The third possible approximation of $C^*(s)$ is the contractor polynomial $C^*_{(3)}(s) = \sum_{k=1}^{n} C^*_{(3)k} s^k + C^\top$ such as

$$\forall k \in \{1,..,n\}, C^*_{(3)k} = C^{(p*)}_{k*p}. \tag{5.10}$$

Proposition 30 *The contractor polynomial $C^*_3(s)$ is enclosing the contractor polynomial $C^*(s)$* i.e.

$$\forall p \in \mathbb{N}^*, \forall k \in \{1,..,n\}, C^*_k \subset C^*_{(3)k}. \tag{5.11}$$

proof. We have

$$\begin{aligned} C^{(p*)}(s) &= \left(C^{(1*)}(s)\right)^p \\ &= \left(\sum_{k=1}^{n} C_k^{(1*)} s^k + C^\top\right)^p. \end{aligned} \tag{5.12}$$

By doing polynomial expansion we deduce that

$$\forall k \in \{1,..,n\}, C_{k*p}^{(p*)} = \left(C_k^{(1*)}\right)^p + C_{rest}. \tag{5.13}$$

where C_{rest} is a contractor which also results from the expansion but is not important for the demonstration.

As such

$$\left(C_k^{(1*)}\right)^p \subset C_{k*p}^{(p*)}. \tag{5.14}$$

On the other hand, using proposition 19 we find that

$$\left(C_k^{(1*)}\right)^\infty = C_k^*. \tag{5.15}$$

As such

$$\forall k \in \{1,..,n\}, C_k^* \subset C_{k*p}^{(p*)}. \tag{5.16}$$

■

As such we proved $C_{(3)}^*(s)$ is a valid approximation of $C^*(s)$ however the convergence was not proved. One of the prospects of the research is to check if the contractor polynomial $C_{(3)}^*(s)$ converges to the polynomial $C^*(s)$ defined in 5.3 when $p \to \infty$ *i.e.*

$$\lim_{p\to\infty} (C_{k*p}^{(p*)}) = C_k^*. \tag{5.17}$$

As for now only a demonstration example is provided in the next subsubsections.

Algorithm for the relaxed composition of contractors

In this subsubsection, we illustrate the fact that an evaluation of a contractor polynomial naturally leads to an algorithm allowing to compute the relaxed composition of contractors. Consider the polynomial $C^{(1*)}(s)$ defined in subsubsection 5.1.3. We remind that

$$C^{(1*)}(s) = (C_n s + C^\top) \circ ... \circ (C_1 s + C^\top). \tag{5.18}$$

The result of contractor computations are usually stored in form of boxes. Naturally, the result of computations using contractor polynomials are stored in the form of *box*

polynomials i.e. polynomials with box valued coefficients. For a given initial box $[\mathbf{x}] \in \mathbb{R}^n$, the evaluation of the contractor polynomial $C^{(1*)}(s)$ is the *box polynomial* denoted by $C^{(1*)}(s)([\mathbf{x}])$ such as

$$C^{(1*)}(s)([\mathbf{x}]) = \sum_{i=0}^{n} C_i^{(1*)}([\mathbf{x}])s^i + [\mathbf{x}]. \tag{5.19}$$

Definition 21 *Consider a* box polynomial *denoted by* $[X](s)$

$$[X](s) = \sum_{i=0}^{n} [\mathbf{x}_i]s^i. \tag{5.20}$$

The contraction of a box polynomial by a contractor C is defined by

$$C([X](s)) = \sum_{i=0}^{n} C([\mathbf{x}_i])s^i. \tag{5.21}$$

The following recursive formulation shows how to obtain the box polynomial $C^{(1*)}(s)([\mathbf{x}])$.

Proposition 31 *Consider the box polynomial $[X_n](s)$ constructed using the following recursive formula (the Horner scheme)*

$$[X_{k+1}](s) = C_{k+1}([X_k](s)) * s + [X_k](s). \tag{5.22}$$

such as the initial condition is

$$[X_0](s) = [\mathbf{x}]. \tag{5.23}$$

The box polynomial $[X_n](s)$ is the evaluation of the contractor polynomial $C^{(1*)}(s)$ i.e.

$$[X_n](s) = C^{(1*)}(s)([\mathbf{x}]). \tag{5.24}$$

proof. By construction. It is just the evaluation of the polynomial $C^{(1*)}(s)$ using the non expanded form (equation 5.18). ■

Remark 20 *Computing the box polynomial $[X_n](s)$ requires $\frac{n(n+1)}{2}$ contraction operations and $\frac{n(n-1)}{2}$ box union operations hence the $O(n^2)$ complexity of the algorithm.*

Figure 5.1 illustrates the recursive relation for *interval polynomials*.

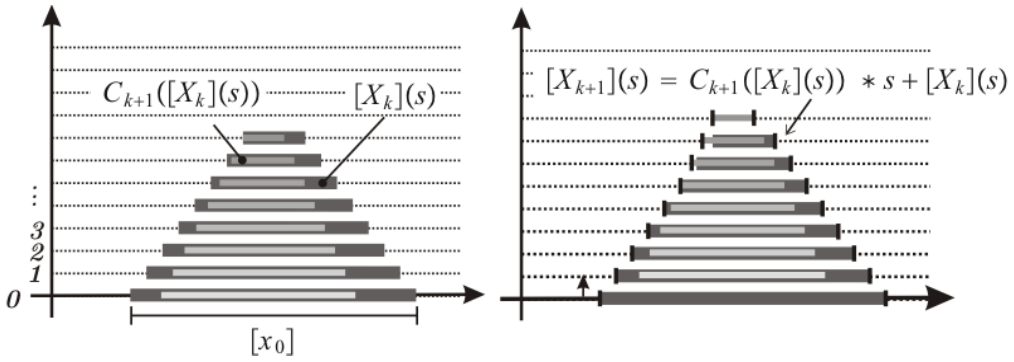

Figure 5.1: One dimensional example of recursive evaluation of the contractor polynomial sort transform using interval polynomials

5.1.4 Finding line intersection

This subsection illustrates the application of the recursive formulation defined in the previous subsection to find the intersection of lines. Consider n lines $l_1, .., l_n$ where q among them doesn't intersect in the same place. We call those lines outlier lines. The purpose is to find the point where most of the lines intersect. Denote by $C_1, .., C_n$ the contractors which sets of insensitive points are the points of the lines $l_1, .., l_n$ respectively. If there are no outlier lines ($q = 0$), the problem can be solved using constraint propagation by doing repeated composition of the contractors $C_1, .., C_n$ (See subsection 4.5). This composition can be represented by a contractor C which contracts the input box into the point of intersection of the lines. The contractor C is defined by

$$C = (C_1 \circ .. \circ C_n)^{\infty}. \tag{5.25}$$

In the case of outlier lines this method would return an empty set. As seen in subsection 5.1.2, the contractors $C_1, .., C_n$ are used to define the contractor polynomial $C^*(s)$ which represent the solution of the problem. Subsection 5.1.3 presented different methods to compute an approximation of the contractor polynomial $C^*(s)$. The two first methods uses the relaxed intersection of contractors and the relaxed composition of contractors to define the coefficients of the two approximating contractor polynomials $C^*_{(1)}(s)$ and $C^*_{(2)}(s)$. The last method is based on contractor polynomial arithmetics to compute the approximating contractor polynomial $C^*_{(3)}(s)$. In the example, the evaluation of the polynomials $C^*_{(1)}(s)$ and $C^*_{(3)}(s)$ and the associated $C^{(p*)}(s)$ are presented for different values of p (repetition factor). Figure 5.2 illustrates a computer evaluation of those polynomials with 20 lines where 7 of them are not passing through the same point. The sub-figures 5.2.(a),(b),(c) and (d) represents the lines as well as the box polynomials in different cases (the darker the box the higher the degree of the box coefficient). The sub-figures 5.2. (a'),(b'),(c') and (d') represent the interval polynomials corresponding to the projection of the box

polynomials on the horizontal axis. Here, the vertical axis represents the degree of the interval coefficient. The purpose of the projection is to be able to easily see the appearance of the solution (in the form of a peak). Sub-figure 5.2.(a) illustrates the resulting box polynomial $C^*_{(3)}(s)([\mathbf{x}])$ for $p = 1$ (no repetition). The white box represents the box corresponding to the $(n-q)^{th}$ coefficient of the polynomial. This white box contains the guaranteed solution. Note that the white box is still large at this point. Sub-figure 5.2.(b) illustrates the polynomial $C^{(p*)}(s)([\mathbf{x}])$ for $p = 10$. Sub-figure 5.2.(c) illustrates the polynomial $\hat{C}^*(s)([\mathbf{x}])$ for $p = 10$. Sub-figure 5.2.(c) illustrates the polynomial $C^*_{(1)}(s)([\mathbf{x}])$ for $p = 30$. Remark that when $p = 10$ ($p = 30$ for $C^*_{(1)}(s)$ since the method is less efficient) there is a clear peak appearing in the polynomial. This peak corresponds to the point where the intersection of most of the lines takes place. The white box also converges to that point of intersection. Having a peak doesn't ensure that there is a solution in the boxes of the peak but it can be used as a motivation to verify the peak box with other guaranteed methods. Remark that there is no notable difference between $C^*_{(1)}(s)([\mathbf{x}])$ and $C^*_{(3)}(s)([\mathbf{x}])$ when p is high.

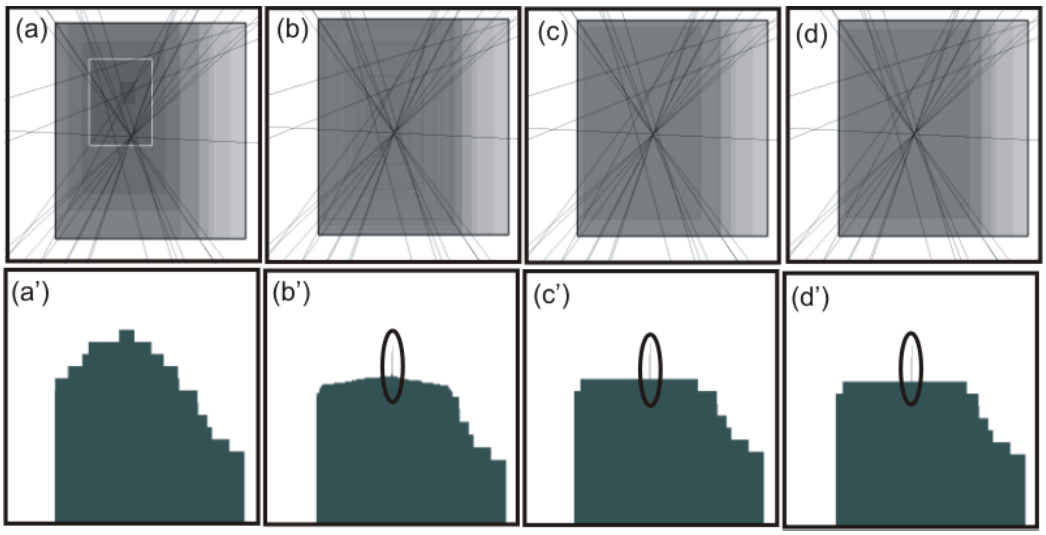

Figure 5.2: Finding the point where most lines intersect using box polynomials

5.1.5 Conclusion

This section proposed an example of implementation of set polynomials using contractor polynomials. An algorithm used to evaluate this contractor polynomial was presented. This algorithm allows to solve relaxed CSP with punctual solutions where the number of inconsistent constraints is unknown. To characterize larger more complex sets, the idea, which is one of the prospects in the research of this PhD thesis, is to create an algorithm similar to the branch and prune algorithm using box polynomials instead of boxes. The

bisection of a box polynomial (*i.e.* of all its coefficients) would be done along the same axis corresponding to the bisection of the box coefficient with the highest degree.

5.2 Implementation of the accumulators

5.2.1 Introduction

This section presents an implementation of the accumulator firstly introduced in section 2.6. The accumulator represents the solution of a relaxed CSP in form of a function which for each element from the search space (domain of the relaxed CSP) returns the number of constraints it satisfies. In order to implement the accumulator on the computer, the idea is to discretize that function. The search space is discretized by using a subpaving in the form of a grid. Each box of this subpaving is characterized by two numbers being the maximum and minimum number of constraints satisfied by an element of this box. Subsection 5.2.2 explains this implementation. Subsection 5.2.3 presents a possible application of the accumulators for the localization of a robot using range measurements to known landmarks.

5.2.2 Discrete accumulator

This subsection describes how to implement a continuous accumulator on a computer. Consider the relaxed CSP

$$\begin{array}{l} \mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i] \\ \mathbf{x} \in [\mathbf{x}_0], [\mathbf{y}_i] \subset \mathbb{R}^{p_i}, i \in \{1, .., n\}. \end{array} \tag{5.26}$$

Denote by $\mathcal{A} : \mathbb{X} \rightarrow \mathbb{N}$ the solution accumulator associated to this relaxed CSP (as explained in section 2.6). The main idea is to discretize the search space $[\mathbf{x}_0]$ into a subpaving (see 3.7). Each box $[\mathbf{x}] \subset [\mathbf{x}_0]$ of this subpaving is to be characterized by two numbers being the maximum and minimum of constraints that are satisfied by an element of the box. Denote by $[\mathcal{A}]$ the inclusion function of $\mathcal{A}$. The natural number interval $[\mathcal{A}]([\mathbf{x}])$ bounds are respectively the maximum and minimum of constraints satisfied by any element in $[\mathbf{x}]$

$$[\mathcal{A}]([\mathbf{x}]) = [\min_{\mathbf{x} \in [\mathbf{x}]}(\mathcal{A}(\mathbf{x})), \max_{\mathbf{x} \in [\mathbf{x}]}(\mathcal{A}(\mathbf{x}))] = [[\mathcal{A}]([\mathbf{x}])^-, [\mathcal{A}]([\mathbf{x}])^+]. \tag{5.27}$$

In practice rather than computing $[\mathcal{A}]([\mathbf{x}])$ it is simpler to compute an approximation denoted $[\mathcal{A}^*]([\mathbf{x}])$ which uses the inclusion tests associated to the constraints in the relaxed

CSP such as

$$[\mathcal{A}^*]([\mathbf{x}])^+ = Card(\{i \in \{1,..,n\}, [\mathbf{f}_i]([\mathbf{x}]) \cap [\mathbf{y}_i] \neq \emptyset, [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}),$$

and

$$[\mathcal{A}^*]([\mathbf{x}])^- = Card(\{i \in \{1,..,n\}, [\mathbf{f}_i]([\mathbf{x}]) \subset [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}). \tag{5.28}$$

The following proposition validates this approximation.

Proposition 32 *Given a box* $[\mathbf{x}] \in \mathbb{R}^m$*,*

$$[\mathcal{A}]([\mathbf{x}]) \subset [\mathcal{A}^*]([\mathbf{x}]). \tag{5.29}$$

proof. Consider $\mathbf{x} \in [\mathbf{x}]$ satisfying $[\mathcal{A}]([\mathbf{x}])^+$ constraints of the relaxed CSP in (5.26)

$$\exists \mathbf{x} \in [\mathbf{x}], \exists \mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K}) = [\mathcal{A}]([\mathbf{x}])^+, \forall k \in \mathbb{K}, \mathbf{f}_k(\mathbf{x}) \in [\mathbf{y}_k], \tag{5.30}$$

as a consequence

$$\exists \mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K}) = [\mathcal{A}]([\mathbf{x}])^+, \forall k \in \mathbb{K}, [\mathbf{f}_k]([\mathbf{x}]) \cap [\mathbf{y}_k] \neq \emptyset, \tag{5.31}$$

thus

$$[\mathcal{A}]([\mathbf{x}])^+ \leq [\mathcal{A}^*]([\mathbf{x}])^+. \tag{5.32}$$

On the other hand

$$\exists \mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K}) = [\mathcal{A}^*]([\mathbf{x}])^-, \forall k \in \mathbb{K}, [\mathbf{f}_k]([\mathbf{x}]) \subset [\mathbf{y}_k], \tag{5.33}$$

thus

$$\exists \mathbb{K} \subset \{1,..,n\}, Card(\mathbb{K}) = [\mathcal{A}^*]([\mathbf{x}])^-, \forall k \in \mathbb{K}, \forall \mathbf{x} \in [\mathbf{x}], [\mathbf{f}_k](\mathbf{x}) \in [\mathbf{y}_k], \tag{5.34}$$

this means that all the elements in $[\mathbf{x}]$ satisfy at least $[\mathcal{A}^*]([\mathbf{x}])^-$ constraints. Since $[\mathcal{A}]([\mathbf{x}])^-$ is the minimum number of constraints satisfied by an element of $[\mathbf{x}]$ it can't be lower than $[\mathcal{A}^*]([\mathbf{x}])^-$.

$$[\mathcal{A}^*]([\mathbf{x}])^- \leq [\mathcal{A}]([\mathbf{x}])^-. \tag{5.35}$$

■

Definition 22 *Consider a non-empty box* $[\mathbf{a}] \in \mathbb{R}^m$ *and a vector of natural numbers* $\mathbf{s} = (s_1,..,s_m) \in \mathbb{N}^m$*. A* grid *of boxes denoted* $\mathbf{Grid}([\mathbf{a}], s)$ *is a subpaving in the form of a matrix of the size* $\mathbf{s}$ *of* $\mathbb{R}^m$ *boxes such as each box touches the neighboring box in the matrix (two boxes are touching if their intersection contains a singleton interval). This leads to the following expression of the* grid

$$\begin{aligned}\mathbf{Grid}([\mathbf{a}], s) = \{([a_1^- + \tfrac{w([a_1])}{s_1} * (i_1 - 1), a_1^- + \tfrac{w([a_1])}{s_1} * i_1], .. \\ , [a_m^- + \tfrac{w([a_m])}{s_m} * (i_m - 1), a_m^- + \tfrac{w([a_m])}{s_m} * i_m]) \\ , i_k \in \{1,..,s_k\}, k \in \{1,..,m\})\}.\end{aligned} \tag{5.36}$$

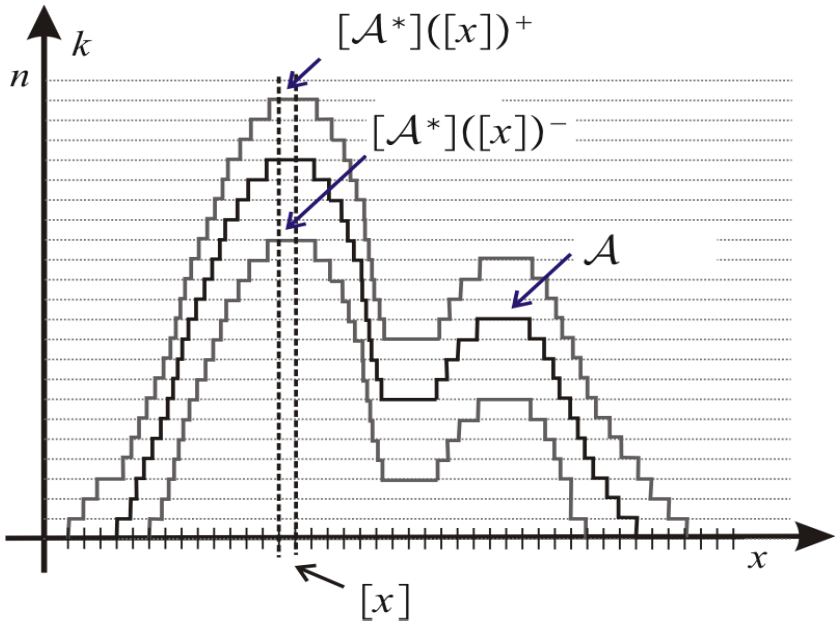

Figure 5.3: Illustration of the interval solution accumulator which contains the solution accumulator

The continuous accumulator $\mathcal{A}$ can then be discretized using a subpaving in form of a grid $\mathbf{Grid}([\mathbf{a}], s), \mathbf{s} \in \mathbb{N}^m, [\mathbf{a}] \subset \mathbb{R}^m$. Each box $[\mathbf{x}]$ of the grid is weighted with two natural numbers corresponding to the bounds of the interval $[\mathcal{A}^*]([\mathbf{x}])$. The interval $[\mathcal{A}^*]([\mathbf{x}])$ is computed on a computer for each box of the grid and stored in computer's memory.

Definition 23 *The function* $[\mathcal{A}^*]$ *which associates to each box* $[\mathbf{x}]$ *of a grid* $\mathbf{Grid}([\mathbf{x}_0], s), \mathbf{s} \in \mathbb{N}^m$ *the natural numbers* $[\mathcal{A}^*]([\mathbf{x}])^-$ *and* $[\mathcal{A}^*]([\mathbf{x}])^+$ *is called the* interval solution accumulator *of the relaxed CSP (5.26).*

Remark 21 *The use of the grid is to simplify computer implementation. It is possible to use any kind of weighted subpaving.*

Figure 5.3 illustrates the approximation of a one dimensional accumulator $\mathcal{A}$ (corresponding to a hypothetical one dimensional relaxed CSP) using the *interval solution accumulator* $[\mathcal{A}^*]$ evaluated on a subpaving in form of a grid.

For a box $[\mathbf{y}] \in [\mathbf{x}_0]$ which does not belong to the grid, it is possible to evaluate $[\mathcal{A}^*]([\mathbf{y}])$ based on the previously computed intervals $[\mathcal{A}^*]([\mathbf{x}])$ for $[\mathbf{x}] \in \mathbf{Grid}([\mathbf{x}_0], s)$ (stored on computer's memory in form of a grayscale image) without having to redo the inclusion tests for $[\mathbf{y}]$. Basically

$$\begin{aligned}[\mathcal{A}^*]([\mathbf{y}])^+ &= \max([\mathcal{A}^*]([\mathbf{x}])^+, [\mathbf{x}] \in \mathbf{Grid}([\mathbf{x}_0], s), [\mathbf{x}] \cap [\mathbf{y}] \neq \emptyset) \\ [\mathcal{A}^*]([\mathbf{y}])^- &= \min([\mathcal{A}^*]([\mathbf{x}])^-, [\mathbf{x}] \in \mathbf{Grid}([\mathbf{x}_0], s), [\mathbf{x}] \cap [\mathbf{y}] \neq \emptyset).\end{aligned} \tag{5.37}$$

In our applications, we usually only compute the upper bound $[\mathcal{A}^*]([\mathbf{x}])^+$ of $[\mathcal{A}^*]([\mathbf{x}])$ for all the boxes $[\mathbf{x}]$ of the grid. In the case of localization, the position we seek is a small set. As such, $[\mathcal{A}^*]([\mathbf{x}])^-$ usually gives a poor estimation of $[\mathcal{A}]([\mathbf{x}])^-$ when the box $[\mathbf{x}]$ is large.

The image by a function of an accumulator is important to enable the exploitation of the solution of the relaxed CSP. In the case of localization, the image by a function of a discrete accumulator is used to either deduce the global position of detected objects from robots position or to deduce the next position of the robot using its evolution function.

Definition 24 *Consider the* discrete solution accumulator $[\mathcal{A}^*]$ *associated to a grid of boxes* $\mathbf{Grid}([\mathbf{x}_0], s), \mathbf{s} \in \mathbb{N}^m$. *The image by a function* $\mathbf{f}: \mathbb{R}^m \to \mathbb{R}^m$ *of* $[\mathcal{A}^*]$ *is a* discrete accumulator *denoted* $\mathbf{f}([\mathcal{A}^*])$ *associated to the grid* $\mathbf{Grid}([\mathbf{f}]([\mathbf{x}_0]), s)$ *defined by*

$$\forall [\mathbf{x}] \in \mathbf{Grid}([\mathbf{f}]([\mathbf{x}_0]), s), \mathbf{f}([A^*])([\mathbf{x}]) = [A^*]([\mathbf{f}^{-1}]([\mathbf{x}])). \tag{5.38}$$

A 2D accumulator can be stored on a computer as a grayscale image and a 3D accumulator can be stored as a video. This makes this type of accumulators easy to implement on a computer. Besides, several algorithms for image/video compression could be used to compress an accumulator so that it would take less space resources.

5.2.3 Application to localization

Problem

Consider a robot defined by a point $R(x, y)$ and N marks defined by $M_i(x_i, y_i)$ in a 2D workspace. With appropriate sensors, the robot measures the distance d_i to the marks. Some of the measurements are outliers. In the example shown on Figure 5.4 d_4, the measurement of distance to M_4, is an outlier. Each measurement d_i constrains the robot to stay on a circle centered in M_i with the radius d_i. This leads to the following constraint

$$C_i : (x - x_i)^2 + (y - y_i)^2 - d_i^2 = 0. \tag{5.39}$$

Remark 22 *A minimum of 3 correct measurements are needed for proper localization. In that case, the points have to be non collinear.*

Simulation results

Consider a set of 3 distance measurements $\{[d_1], [d_2], [d_3]\}$ where 1 of them is an outlier. As seen in remark 22, it is impossible to localize the robot with such data. Those three

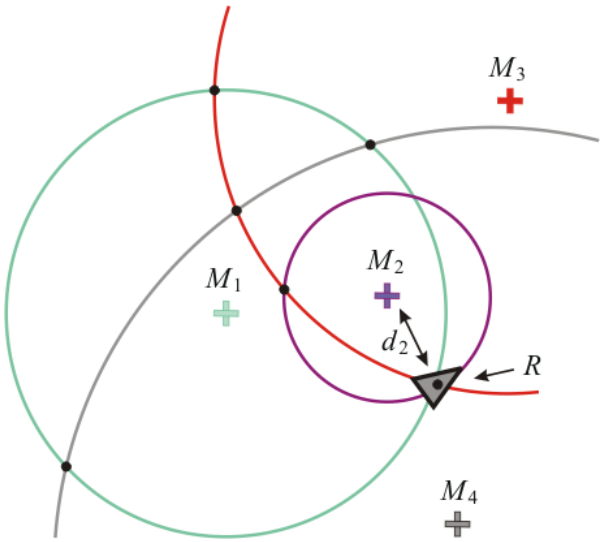

Figure 5.4: Localization using goniometric data

measurements lead to a relaxed CSP defined by 3 constraints $\{C_1, C_2, C_3\}$ as seen in (5.39). Denote by $\mathcal{A}_1$ the solution accumulator corresponding to this first relaxed CSP. The discrete approximation of the accumulator $\mathcal{A}_1$ (see subsection 5.2.2) is a 2D discrete accumulator and can be represented on a grayscale image represented in Figure 5.5. The figure also represents the true position of the robot marked by a big cross and the detected marks marked with small crosses. Remark that there are 3 different maxima corresponding to 3 potential positions of the robot.

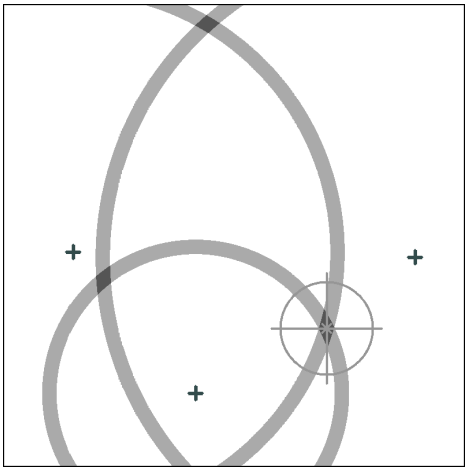

Figure 5.5: The accumulator $\mathcal{A}_1$ corresponding to the case of 3 distance measurements with 1 outlier

Consider another set of 4 distance measurements from another source $\{[d_4], [d_5], [d_6], [d_7]\}$ where 2 of them are outliers. As seen in remark 22, it is impossible to localize the robot

with such data. Those four measurements lead to a relaxed CSP defined by 4 constraints $\{C_4, C_5, C_6, C_7\}$ as seen in (5.39). Denote by $\mathcal{A}_2$ the solution accumulator corresponding to this second relaxed CSP. The discrete approximation of the accumulator $\mathcal{A}_2$ is represented on Figure 5.6. The Figure also represents the true position of the robot marked by a big cross and the detected marks marked with small crosses. Remark that there are several different maxima corresponding to the potential positions of the robot.

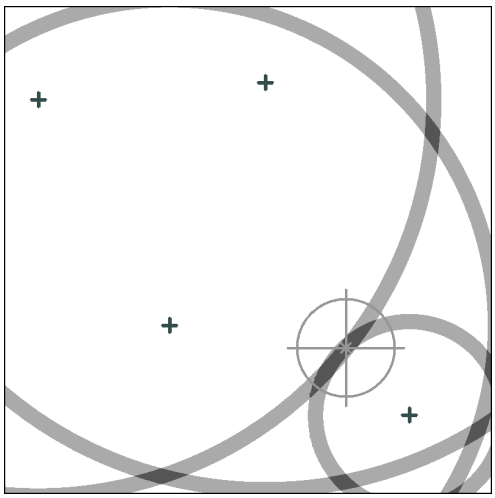

Figure 5.6: The accumulator $\mathcal{A}_2$ corresponding to the case of 4 distance measurements with 2 outliers

Consider the accumulator $\mathcal{A}$ which is the sum of the accumulators $\mathcal{A}_1$ and $\mathcal{A}_2$.

$$\mathcal{A} = \mathcal{A}_1 + \mathcal{A}_2. \tag{5.40}$$

The accumulator $\mathcal{A}$ actually corresponds to the solution accumulator for the relaxed CSP defined by the constraints $\{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$. The accumulator $\mathcal{A}$ is the solution of the localization problem considering all the 7 measurements. The accumulator $\mathcal{A}$ is the result of the sum of the discrete forms of the accumulators $\mathcal{A}_1$ and $\mathcal{A}_2$ and its discrete form is represented in Figure 5.7. Note that the accumulator $\mathcal{A}$ is not computed from scratch. Remark that now, since 4 measurements are correct, there is only one maxima and it is possible to localize the robot hence the usefulness of the approach.

Remark 23 *The accumulators $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$ can be computed on different machines. Besides, the accumulators $\mathcal{A}_1$ and $\mathcal{A}_2$ can be stored as compressed grayscale images and easily exchanged between machines. As a consequence, the approach is useful for distributed computing of problems with outliers.*

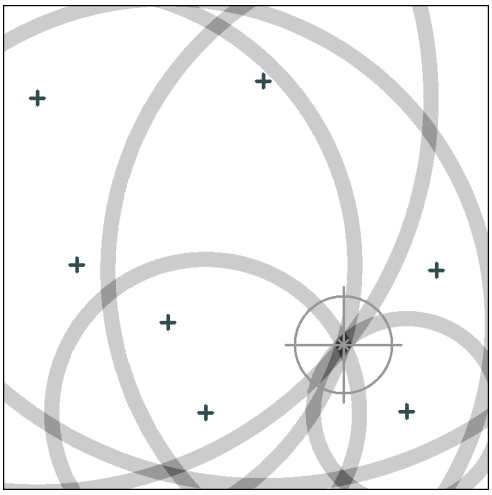

Figure 5.7: The sum of both accumulators $\mathcal{A}_1$ and $\mathcal{A}_2$ leads an accumualtor $\mathcal{A}$ corresponding to solution of the problem of localization considering all the measurements.

5.2.4 Conclusion and perspectives

In this section a possible implementation of the accumulator was presented. The idea was to discretize the domain of the relaxed CSP in form of a grid. Each box of the grid is characterized by an interval defining the maximum and minimum of constraints satisfied by an element in the box. The representation of the solution in the form of accumulators allows distributed computations. An application of accumulators was presented in the case of robot localization using distributed computations.

Chapter 6

Application to the localization of autonomous underwater vehicles

6.1 Introduction

In this part, an application of the set membership methods described in this thesis is presented. The algorithms were tested on a dataset derived from experiments carried out in a marina located in the Costa Brava (Spain) with Girona university AUV [Ribas et al., 2008]. A comparison between set membership methods and Particle Filtering methods is also presented. The comparison was made in collaboration with Heriott Watt University in Edinburgh.

The same algorithms are used on our school's underwater robot *Sauc'isse* since its sensors are the same as Girona AUV. The purpose of the robot *Sauc'isse* is to participate to a competition of autonomous underwater vehicles called SAUC'E. The robot and the competition are presented in the last part of the manuscript.

6.2 The experiment

The localization experiment to be considered here has been designed in order to illustrate a method for underwater SLAM (see [Ribas et al., 2008]). The data was gathered during an extensive survey of a abandoned marina in the Costa Brava (Spain). Girona university *Ictineu* AUV gathered a data set along a $600m$ trajectory which included a small loop around the principal water tank and a $200m$ straight path through an outgoing canal. The data set included measurements from the Imaging Sonar (a Tritech Miniking), DVL - Doppler Velocity Log - (a SonTek Argonaut) and MRU - Motion Reference Unit - sensors

(Xsens MTi). For validation purposes, the vehicle was operated close to the surface attached to a GPS equipped buoy used for registering the real trajectory (ground truth).

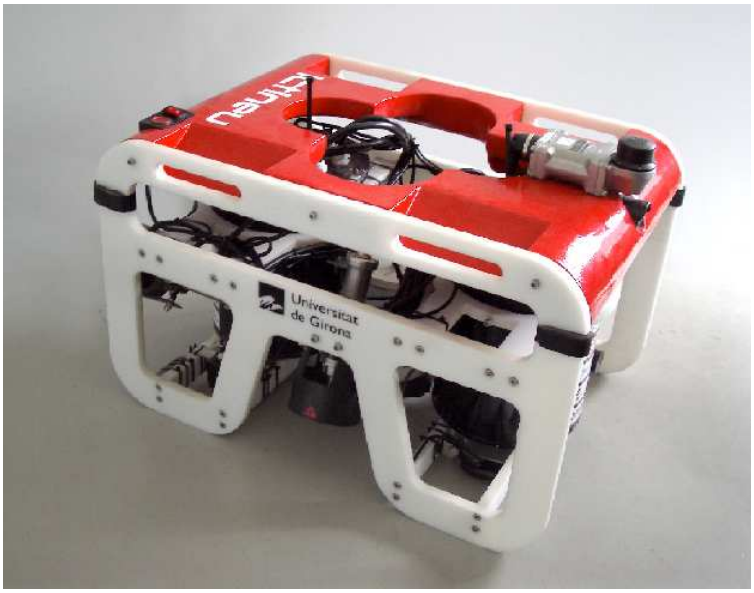

Figure 6.1: Girona University AUV *Ictineu*

6.3 Using an imaging sonar

Before talking about solving the problem of localization, we will talk about the most important sensor to perform this task which is the imaging sonar. The operating principle of a imaging sonar is represented in Figure 6.2. A real image sonar taken in a rectangular pool is represented in Figure 6.3. The sector scan sonar contains a rotating head which contains a directional ultrasonic transmitter and receiver (the transmitter and receiver are very often one component made using a piezoelectric material). For each angle step of the head rotation, the transmitter emits an short burst of an ultrasonic wave. The wave then reflects on the eventual obstacles in water (walls, fish, divers, water surface, bubbles....) those echoes are then received by the receiver with a certain amplitude. This data can be represented in a time/amplitude workspace as seen in Figure 6.2. The farther the obstacle the longer it takes to the echo to return to the receiver. On the other hand, the bigger the object, the bigger the amplitude of the echo. So usually, we keep echoes with an amplitude superior to a specific threshold. The distance to an object is proportional to the time it takes to receive the echo caused of by the object. As a consequence, what we obtain in the end is a set of distance intervals $\{[d_0], .., [d_n]\}$ corresponding to highly reflecting objects. In Figure 6.2, the interval $[d_1]$ corresponds to the distance to the first reflecting object which is the wall. The interval $[d_2]$ corresponds to the echo of the second. This

type of measurement is due to multiple rebounds of the acoustic wave and is considered as an outlier in our computations. Sometimes the object which to be detected is out of range (too far or too close). $[d_0]$ and $[d_3]$ represent the domains of distance to eventual highly reflecting objects which are *"invisible"* to the sonar.

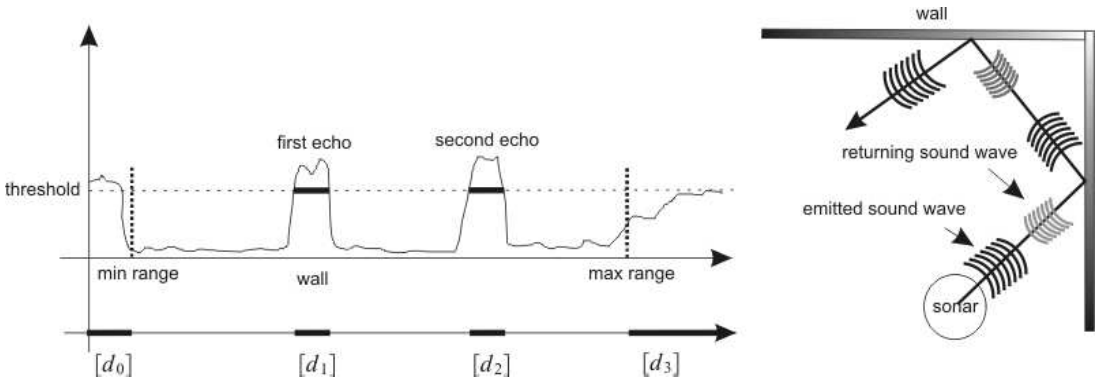

Figure 6.2: The principle of sonar functionning

In order to localize in the Marina, what we are interested in are the echoes due to the walls. All the other echoes will be considered as outliers. Figure 6.4 shows a sample of sonar data, the dots show where the sonar beam found an obstacle which is most of the case the wall from the marina. The long segments mean that obstacles are out of range i.e. beyond $50m$. As one can see, the data is noisy and contains a lot of outliers.

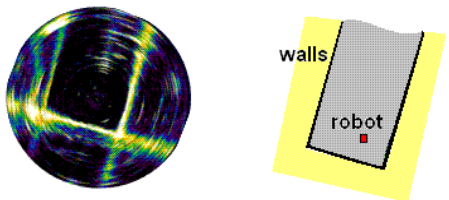

Figure 6.3: Real sonar image taken in a rectangular pool

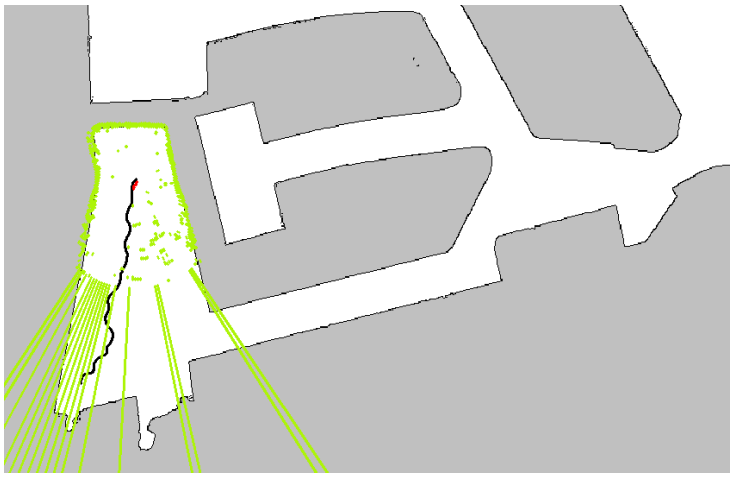

Figure 6.4: 360° scan of the marina using sector scan sonar

6.4 Setting the problem into a system of equations

A dynamic system such as an underwater robot can usually be characterized by discrete-time dynamic equations

$$\begin{aligned} &\mathbf{f_k} : \mathbb{R}^m \rightarrow \mathbb{R}^m, \mathbf{g_k} : \mathbb{R}^m \rightarrow \mathbb{R}^\ell \\ &\mathbf{x_{k+1}} = \mathbf{f_k}(\mathbf{x_k}) \\ &\mathbf{y_k} = \mathbf{g_k}(\mathbf{x_k}). \end{aligned} \tag{6.1}$$

where $\mathbf{x_k}$ is the state of the system, $\mathbf{y_k}$ is the output vector, $\mathbf{f_k}$ is the evolution function and $\mathbf{g_k}$ the observation function. The input of the system is enclosed in the expression of $\mathbf{f_k}$. The noise (due to model imperfection) is neglected to simplify the different formulations. In our case, $\mathbf{x}_k$ is the robots pose, $\mathbf{f_k}$ characterizes robots dynamics, $\mathbf{y}_k$ is the measurement vector (here sonar distance to first obstacle measurements). $\mathbf{y}_k$ and $\mathbf{x}_k$ are related by the observation function $\mathbf{g_k}$ which express in our case geometrical relations between the position, the measurements and the map. Denote by $[\mathbf{y_k}]$ the box containing the measurement $\mathbf{y_k}$. Using state equation in (6.1), the problem of estimation of $\mathbf{x_k}$ can be cast into the following constraint satisfaction problem

$$\begin{cases} \mathbf{g}_k(\mathbf{x}_k) = \mathbf{y}_k \\ \mathbf{g}_{k-1} \circ \mathbf{f}_{k-1}^{-1}(\mathbf{x}_k) = \mathbf{y}_{k-1} \\ ... \\ \mathbf{g}_{k-n-1} \circ \mathbf{f}_{k-n-1}^{-1} \circ ... \circ \mathbf{f}_{k-1}^{-1}(\mathbf{x}_k) = \mathbf{y}_{k-n} \\ \mathbf{x}_k \in \mathbb{R}^m, \mathbf{y}_i \in [\mathbf{y}_i], i \in \{k-n, .., k\}. \end{cases} \tag{6.2}$$

In this CSP, we assume that the state $\mathbf{x}_k$ (position of the robot) at the time step k is *a priori* completely unknown (prior domain of membership is $\mathbb{R}^m$). In the localization

jargon, this corresponds to the *global localization*. It is possible to exploit the fact that the current state of the system depends of its previous state since

$$\begin{array}{l} \mathbf{f_k} : \mathbb{R}^m \rightarrow \mathbb{R}^m \\ \mathbf{x_k} = \mathbf{f_{k-1}}(\mathbf{x_{k-1}}). \end{array} \tag{6.3}$$

Denote by $\mathbb{X}_{k-1}$ the solution set of the CSP corresponding to the problem of estimation of $\mathbf{x}_{k-1}$ using set membership methods. We have

$$\mathbf{x_k} \in [\mathbf{f_{k-1}}]([\mathbb{X}_{k-1}]). \tag{6.4}$$

Consider the CSP

$$\left\{ \begin{array}{l} \mathbf{g}_k(\mathbf{x}_k) = \mathbf{y}_k \\ \mathbf{g}_{k-1} \circ \mathbf{f}_{k-1}^{-1}(\mathbf{x}_k) = \mathbf{y}_{k-1} \\ ... \\ \mathbf{g}_{k-n-1} \circ \mathbf{f}_{k-n-1}^{-1} \circ ... \circ \mathbf{f}_{k-1}^{-1}(\mathbf{x}_k) = \mathbf{y}_{k-n} \\ \mathbf{x}_k \in [\mathbf{f_{k-1}}]([\mathbb{X}_{k-1}]), \mathbf{y}_i \in [\mathbf{y}_i], i \in \{k-n, .., k\}. \end{array} \right. \tag{6.5}$$

In this CSP, we assume that the prior domain of membership of the state $\mathbf{x}_k$ depends on the domain of $\mathbf{x}_{k-1}$ which is supposed to be computed. In the localization jargon, this corresponds to the *dynamic localization*. The purpose is to reduce the size of the search space and as a consequence reduce the computation time for the same precision requirements.

6.5 Robots equations

6.5.1 Evolution function

Robot's evolution can be characterized by the following differential equation

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v * \cos(\theta) \\ v * \sin(\theta) \\ \dot{\theta} \end{pmatrix}.$$

Using Euler discretization we obtain

$$\begin{array}{l} x_{k+1} = x_k + v_k \cos(\theta_k) dt \\ y_{k+1} = y_k + v_k \sin(\theta_k) dt \\ \theta_{k+1} = \theta_k + \omega_k dt, \end{array} \tag{6.6}$$

where (x_k, y_k, θ_k) is the state of the robot, v_k is its speed, θ_k is its orientation and ω_k is the rotation speed at time step k.

θ_k is obtained using magnetic compass and v_k is obtained using DVL (Doppler Velocity Log) data and the rotation speed $\dot{\theta}_k$ can be obtained using the inertial unit gyroscope.

6.5.2 Map

We consider the map $\mathbb{M}$ as set of points which corresponds to the walls of the Marina. The map is stored in a binary image as the one see in Figure 6.5.(b). In fact, as seen in section about image contractors 4.8 set membership methods allow to manipulate non parameterized sets by representing them in the form of binary images. Basically, the binary image corresponds to a pixelization of the characteristic function of that set. For our algorithm, we first took a satellite image from Google Maps (Sub-figure 6.5.(a)). Then we obtained the binary image by using edge detection and manually filtering the result (Sub-figure 6.5.(b)). Then we computed the ψ-transform of the binary image (Sub-figure 6.5.(c)) which is necessary to define both image contractor and image inclusion function (as seen in section 4.8). Sub-figure 6.5.(d) shows the subpaving corresponding to the result using SIVIA Solver [Jaulin and Walter, 1993].

6.5.3 Observation function

We consider the map $\mathbb{M}$ as set of points which corresponds to the walls of the Marina. With the sonar we measure the distance d_k to first obstacle - here marina walls - along a vector defined by the sensor angle α_k relative to the robot (see Figure 6.6). There is a geometrical relationship between the position and the measurement which leads to the following observation function

$$d_k = dist(x_k, y_k, \theta_k, \mathbb{M}). \tag{6.7}$$

Where $dist(x_k, y_k, \theta_k, \mathbb{M})$ corresponds to the constraint distance to obstacle. Denote by $\mathbf{M}_k = (x_k, y_k)$ the position of the robot, $\mathbf{H} = (x_{map}, y_{map})$ a slack variable. This variable represents the points of the map which are supposed to be detected by the sonar. This constraint $dist(x_k, y_k, \theta_k, \mathbb{M})$ can be decomposed into a set of simpler constraints.

$$\begin{aligned} C_1 &: \|\mathbf{M}_k - \mathbf{H}\| = d_k \\ C_2 &: \arg(\mathbf{H} - \mathbf{M}_k) = \alpha_k + \theta_k \\ C_3 &: \mathbf{H} \in \mathbb{M}. \end{aligned} \tag{6.8}$$

where C_1 represents the constraint on the measured distance, C_2 represents the constrains on the absolute angle of the sonar beam, C_3 represents the constraint $\mathbf{H}$ belongs to the map. Note that C_3 is implemented using the image contractor.

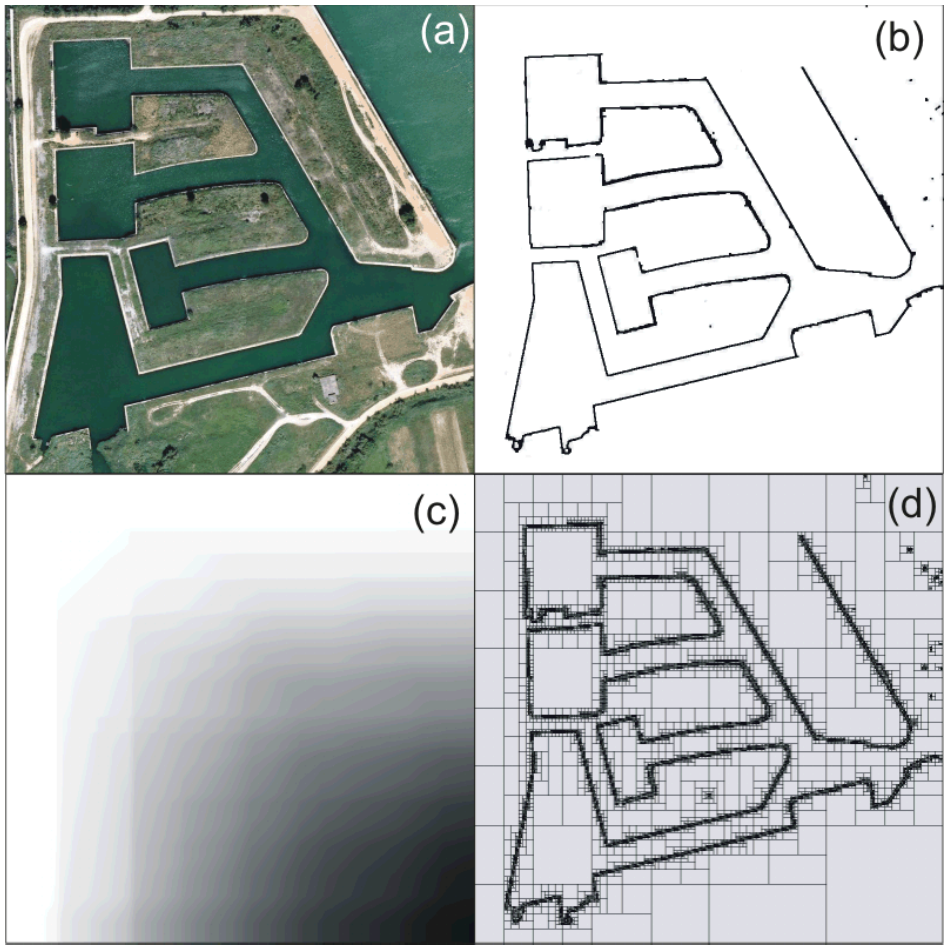

Figure 6.5: The different steps to compute and use the image contractor

6.6 Results

Figure 6.7 shows a comparison between the reference GPS trajectory (in black - for those having colored image) and the Dead Reckoning trajectory (in blue) which is obtained by merging DVL (Doppler Velocity Log) and MRU (Motion Reference Unit) data. We can observe that Dead Reckoning trajectory suffers from an appreciable drift even causing it to go outside the canal. The trajectory computed using set membership approach is represented in Figure 6.8. The algorithm returns the trajectory as a set of boxes (in pink) but we usually take the center of the box as the actual position (in red). The computed trajectory follows the GPS trajectory.

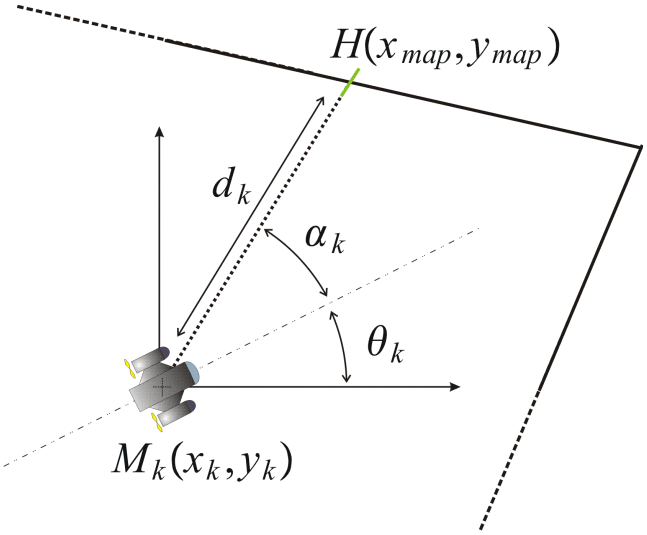

Figure 6.6: Sonar data interpretation

6.7 Comparison with Particle Filtering method

6.7.1 Results

In order to compare set membership methods and Particle Filtering (first introduced in [Gordon et al., 1993]), together with Dr. Francesco Maurelli from Heriott Watt university in Edinburgh, Scotland, we used our respective methods to deal with the Marina dataset. Figure 6.9 represents the result of localization using Particle Filtering. In the Particle Filter formalism, the position is represented by a cloud of weighted particles. Basically, particles are then filtered leaving those which are the most consistent with the measurements (in probabilistic terms). The number of particles is left constant by doing resampling (reproducing the particles with the highest weight). The next step is the prediction of the next position for each particle (since the robot moves). Those different steps (resampling, prediction and update through measurement) are repeated and the trajectory is reconstructed.

6.7.2 Considerations of the two techniques

Both techniques have been proven reliable and are proven to be valid for AUV localization in a man-made underwater environment. It is important to stress that both techniques can work only when the sonar sensor measures are distinctive enough, like marinas and underwater structures. Both do not work well in open sea, if the environment is featureless.

Although radically different in the mathematical background, both techniques have com-

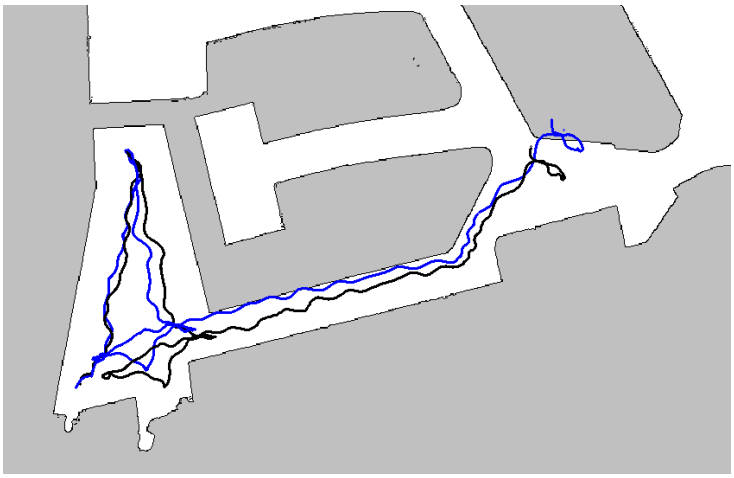

Figure 6.7: Comparing GPS and Dead Reckoning trajectories

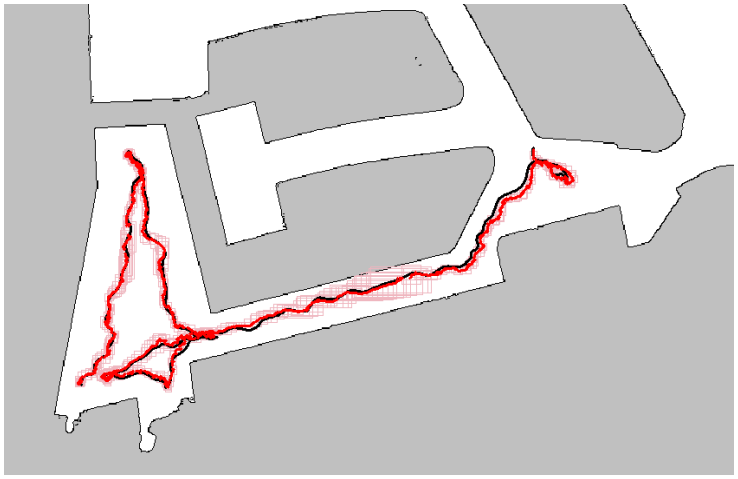

Figure 6.8: Comparing GPS and set membership methods trajectories

mon points. Both methods do not make any assumption on the initial position. They are therefore able to perform global localization and not just position tracking. Both algorithms are very robust against outliers in the sonar measures. Having an estimation of the AUV motion is helpful in both methods, but it is not necessary. However, for environments which are not very distinctive, it is almost an essential information. In the field trial discussed, it is the case of the corridor. In that area the sensor measure would always return similar values, and an estimation of the motion is therefore necessary.

Both methods can be easily parallelized. The computation of the simulated sensor measure from each particle represents the most expensive part for the Particle Filter algorithm. However, each particle is independent and its associated sensor measure can be computed in parallel. The simulated sensor measures could also be precomputed for all possible

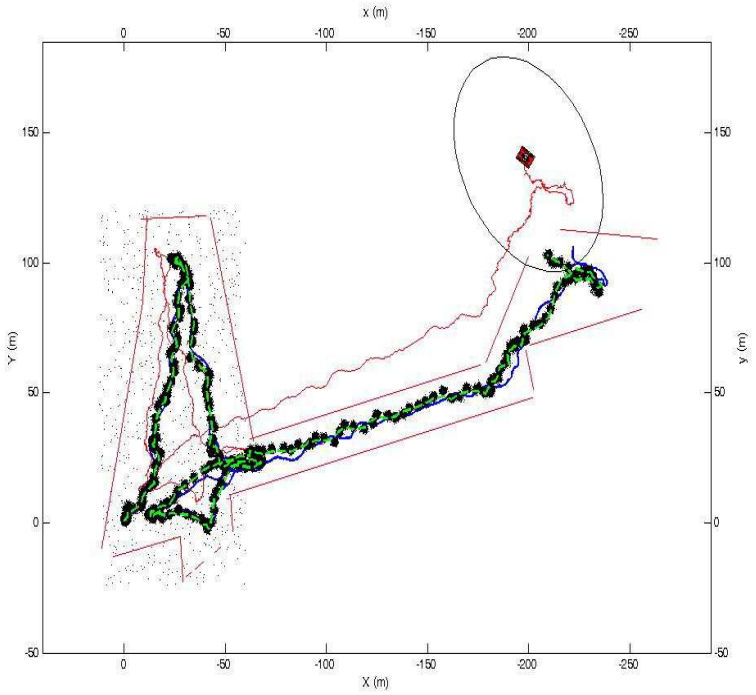

Figure 6.9: A 2D plot of the environment, with the particles, plotted for all the timestamps, the DGPS trajectory (blue), the dead reckoning trajectory (red), the uncertainty ellipse from the dead reckoning, and the trajectory inferred by the particles (green). 600 particles are spread over an area of 10,368 square meters.

position, and the information retrieved with an hash table, thus speeding up the process. Again, the weight calculation for each particle can be computed in parallel. As for the set membership approach, the search space which is initially represented by a box is usually divided into smaller boxes which can be processed separately. The process of characterization of each box i.e. the process of associating the number of consistent measurements with a position box is in fact independent for every box.

Both algorithms are able to recover from wrong convergence and inconsistent situations. The particle filter approach is able to recover *dynamically*, i.e. without changing state of the algorithm. Through the resampling step, a portion of random particles is generated, thus allowing a wide exploration of the environment and the recover from a wrong convergence. The Set Membership approach can detect inconsistent situations and perform again global localization, called from the authors *static* recovery. The algorithm state changes to global localization and, when a convergence is reached, changes back to position tracking.

Particle Filter Techniques can handle non Gaussian and non linear processes. Set Membership Techniques can also handle non linear processes. Set Membership Techniques are not a probabilistic methods. Those methods require assumption on the membership of the variables of the problem. If the assumptions are correct so is the solution. Knowing the probabilistic distribution of a variable can be useful to make assumptions on its membership. As an example, consider a variable x following a Gaussian distribution and x_0 a measurement of this variable. It is possible to assume that the real value of x variable is included in the 99% likelihood interval $[x_0 - 3\sigma, x_0 + 3\sigma]$. Note that the 1% case where that assumption is not true the measurement x_0 can be considered as an outlier and is taken into account by the algorithm.

A clear disadvantage for both methods is represented by the computational requirements. However they are both feasible for real-time execution and they have been used integrated in the AUV architecture. The field trials in the Marina took about 1 hour to be performed. The Particle Filter algorithm has been tested postprocessing the data, with 31 minutes needed on Matlab, on a Core2 2.2 GHz. The Set Membership approach was implemented using C/C++ and needed 1hour 10minutes to execute on one core of a Centrino duo T2500 at 2GHz.

6.8 Conclusion

This part presented an application of set membership methods to the localization of an autonomous robot. The localization problem can be cast into a relaxed CSP. The different methods used for the resolution of the relaxed CSP are introduced in chapter 2 has been used to compute the trajectory of the robot. The image contractor presented in Section

4.8 has been used to represent the map of the environment.

Chapter 7

Autonomous underwater robotics at ENSTA-Bretagne

7.1 Introduction

Since 2007, the engineering school I work in ENSTA-Bretagne (ex-ENSIETA) (Ecole Nationale Supérieure de Techniques Avancées) participates to SAUC-E (Student Autonomous Underwater Challenge - Europe). The team is composed of a professor, PhD students, 1st and 2nd year students and trainees. Some of us were there as students and are now continuing to work on underwater robots as PhD students. This is indeed important to keep the technology from one year to another. We have thus built two low-cost robots SARDINE and SAUC'ISSE (see Figure 7.1) that will be presented in this paper. The first robot, named SAUC'ISSE, is presented on the left of Figure 7.1. The second robot (almost a clone of SAUCISSE), named SARDINE, has also been built in order to study the feasibility of underwater robot collaboration. Our main achievement is that we did not change the overall design of our robot for three years which proves its robustness. This might be due to our approach in building robots. In our robotics club we try in fact to follow three principles. The first is the KISS (Keep It Simple Stupid) principle since only simple things survive. The second is to always try to participate in a competition since it boosts the group motivation. The last principle is to use Commercial Off-The-Shelf components as much as possible since it enhances the quality and reproducibility of the hardware. Our motivation is first being able to contribute in the development of new technologies for underwater robots. The second is to be able to meet other people doing the same and thus exchanging ideas and advancing even further. One of the main problems encountered is to design or find solutions for software architecture suitable for robots.

In this paper, we will first detail the physical architecture (mechanical and electronic) of

our robots. Then, we will explain how we will make each robot autonomous by defining missions and introducing some of our algorithms. Next we will talk about software architecture. Finally, we will conclude with a small paragraph on swarm robotics.

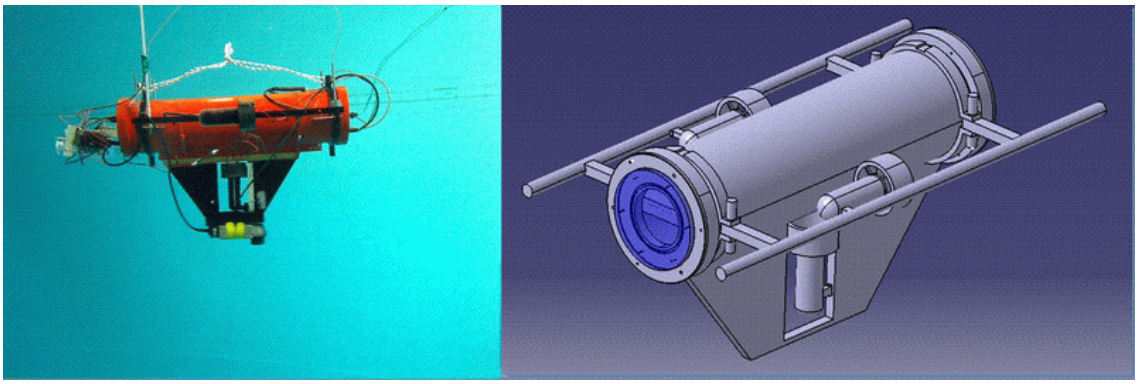

Figure 7.1: Sauc'isse (on the left) and Sardine (on the right)

7.2 SAUCE Competition

7.2.1 Introduction

In 2010 and 2011, the competition is held in the NURC (NATO Underwater Research Center) in La Spezia in Italy. Unlike before, the robots will have to evolve in a port in sea water which complicates the detection of underwater targets.

7.2.2 Tasks to perform during the mission

The AUV must perform a series of tasks autonomously, with no control, guidance, or communication from a person,or from any off-board computer including the GPS system. Each AUV will be tracked via acoustic positioning system in the form of a beacon placed on each of the vehicles.

Task 1 : Validation gate

Move from launch/release point and submerge. The team can choose to initiate an autonomous mission from either Start 1 or Start 2 point. If Start 1 is chosen and the next task is completed the team would be awarded 100 extra points. Pass through the

validation gate4 – without contacting any part of the 'structure'. The gate will be constructed of 2 orange buoys on a rope, 4 m apart (lights5 will be added to the ropes to aid the competitors). Failure to successfully negotiate the validation gate will result in the run being terminated.

Task 2 : Perform the "underwater structure" inspection

Figure 7.2: The pipeline to be inspected

The structure will be constructed of 0.5 m diameter by 1.5 m cylinders to form a pipeline and are shown in Figure 7.2. The structure will be consisting of cylinders but they will not form a straight pipeline but other irregular structure. The structure will be placed on the bottom but will be moved during the competition. The task is follow the stationary structure while maintaining a 0.5 m stand-off distance from it.

Task 3 : Free a mid-water target

A mid-water target will be tethered to the ground by a light rope - a fishing line with 1 mm diameter, 5 kg tension. The vehicles are required to find the

target, go around it – perform obstacle avoidance manoeuvre, return to the initial approach heading (at that point 50 % of Task 3 will be completed) and then part (cut/melt/etc.) the rope tether between the mid-water target and another orange buoy (located about 1 m from the floor). At that point second 50% of Task 3 will be completed. The target will

Figure 7.3: Mid-water target to be freed

be a soft reflective object (both acoustically and optically). The buoy is spherical and colored as shown in Figure 7.3.

Task 4 : Make a wall survey

The wall will be ~10 m from the mid-water target. The objective is to maintain a position > 2m from the wall for the duration of the survey. The wall will not be straight.

Task 5 : Mobile object tracking

Perform tracking (below) of the moving NURC's ASV which will move (slowly) in the competition area. An acoustic pinger will be placed at the center point of ASV about 1.5 m below it. The teams might choose to follow the ASV by either 1) detecting the pinger signal or 2) looking up with an on-board sensor. Completing the task by both methods will result in bonus of 200 points.

Task 6 : Surface in the surfacing zone

Surface in the surfacing zone – the surfacing zone location will be sent from the NURC's ASV via an acoustic signal (20 – 24 kHz). A compact modem will be given to each of teams. The teams will be required to accommodate the modem, provide power to it and

write software to be able to receive modem commands per Interface Control Document (ICD) written by NURC (will be available on the competition web). The surfacing zone must

be attempted last.

Required task

Each team will produce a log file of the mission within around 10 minutes of the end of the run. The format of the log file will be a comma separated ASCII file of the format: Time, position, action, a comment between simple quotes. (SSSSS,XXX.x,YYY.y,ZZZ.z,AA.aa). Logged data will be plotted by plotting routine written by the organizing committee. This will be used to score the log file. For ASV tracking task the additional file of range and bearing data from the AUV to the pinger will need to be provided.

7.3 Physical description

7.3.1 External architecture

Figure 7.4: Sauc'isse mechanical design

The main body of SAUC'ISSE (Figure 7.4) is made with an aluminum tube. The aluminum has been chosen due to its amagnetism (important since a magnetic compass is

used) and its resistance to corrosion. The robot is 70 cm long to ease its transportation and reduce its weight. The corresponding space is sufficient to enclose all the needed devices. The diameter of the tube has been chosen to 17 cm to be able to put an embedded computer (such as an EEE-PC or a PC/104). The propulsion is produced by three thrusters STB150 from SEABOTIX, an American manufacturer specialized in ROVs (Remote Operated Vehicles). Each propeller produces approximately a force of ± 7 N (or ± 0.7 Kg). The two lateral thrusters control the speed and the direction of the robot. The roll and pitch are not controlled by thrusters since the robot is stable thanks to a weighted keel. This stability is a consequence of the fact that the volume center is above the gravity center. The keel also supports the sonar and the vertical thruster (Figure 1). The vertical thruster is used to adjust the depth of the robot. To save money, we made the choice of a single vertical propeller (which is unusual), and thus we had to fix it just below the tube. Unfortunately, when the robot sinks, the water flux is directed toward the tube which brakes down its performances: the corresponding force is now ± 3N (instead of ± 7N). However, the buoyancy of the AUV is inside the interval $[0, 2]$ N, and thus, the vertical propeller is efficient enough to make the AUV sinking or floating. Additionally, we have a system to adjust the overall ballast of the submarine: breakthrough mass lead can be added on 4 threaded rods placed in the four corners of the submarine so we can reach the limit zone of buoyancy. As a result we just need a propelling force very weak to make the submarine go under the surface, and when the vertical engine is shut down, it goes itself to the surface. The water tightness of each tube is made by two aluminum covers (see Figure 2). Waterproof connectors (Switchcraft and Bulgin Buccaneer) are placed on the covers for connections with sensors and actuators of the robot. The water tightness is provided by three stainless fastener screws. The fixation of the screws is done with a pawn center. The extraction of the covers is done with three extraction screws. The front cover of the SARDINE has a window to enable the use of a webcam directly in the tube. All the waterproof connectors are on the back cover which should only be opened rarely, as changing the batteries and switch on the EeePC/PC 104 can be done via the front cover only.

7.3.2 Internal architecture

Rails (Figure 7.5) with glue for aluminum enable us to drag a 6mm thick Plexiglas plaque which is the main support base for the internal electronic devices of SAUC'ISSE. Below the plaque, another sliding support contains the batteries (Ni-MH). We can therefore readily access the batteries without having to touch any of the other electronic devices. These are put above the main Plexiglas plaque.

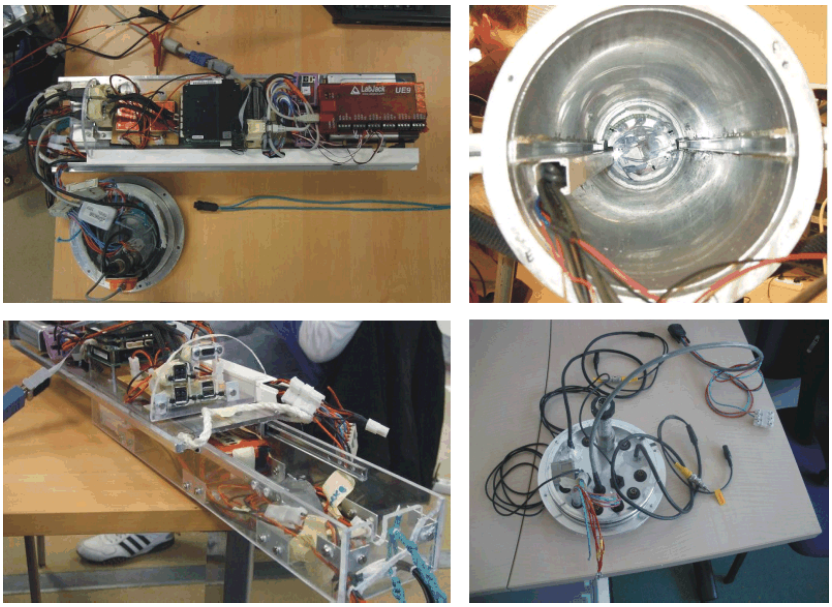

Figure 7.5: Sauc'isse internal architecture

7.4 Electronics

7.4.1 Thruster control

Servo controllers Robbe Rokraft send powered PWM (Pulse Width Modulation) signals to the thrusters. These PWM correspond to a power amplification of other powerless PWM signals generated by an interface module (Labjack UE9) placed between the computer and the servo controllers. The Labjack UE9 is a professional USB device that provides several IO pins to connect to electronic devices.

7.4.2 Computer

The embedded computer of SAUC'ISSE is a PC/104 from EUROTECH with a Pentium M 1.4 GHz CPU and 512 MB of RAM. The operating system and the programs are stored on a hard drive 2.5 of 320 GB. Moreover, 8 USB, 1 Ethernet, 2 RS232 and 1 VGA ports provide all we need to connect external devices and to communicate with the computer. The PC is powered directly from 12 or 24 V batteries thanks to a power supply module compliant with the PC/104 standard that provides regulated 3.3, 5 and $\pm$12. The embedded computer of SARDINE is an ASUS EeePC T91MT. It is cheaper than the PC/104, thin and has an integrated battery allowing autonomy of up to 5 hours, (up to 3 hours with all the sensors connected) with almost the same technical characteristics (CPU, RAM. . .). It just needs a USB hub to connect to all needed devices.

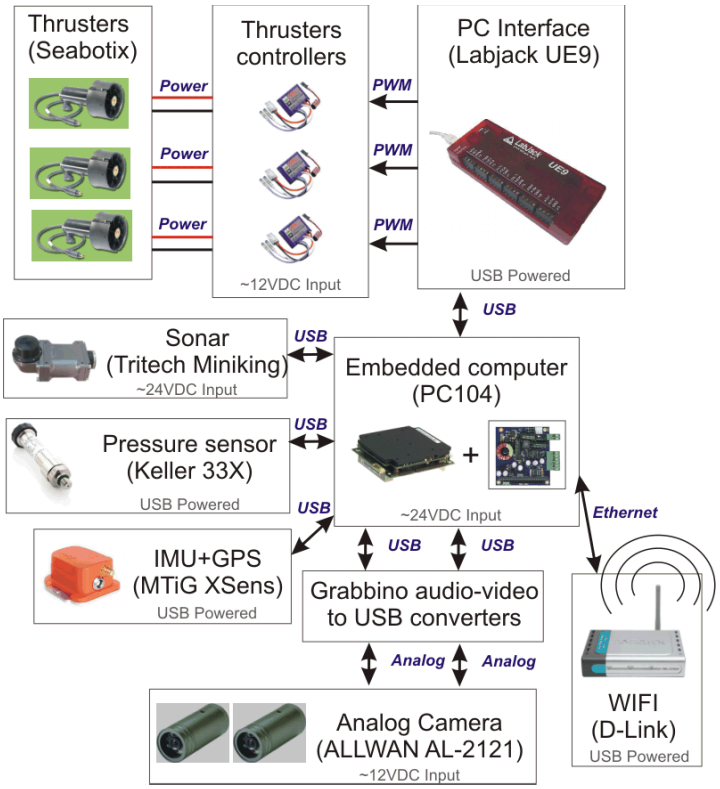

Figure 7.6: Electronics architecture

7.4.3 Sensors

vision

To detect the objects (targets, pipeline ...) SARDINE has standard webcams Logitech Quickcam Pro 9000, put directly in the tube behind the front window. It gets pictures up to a very high resolution (1600x1200). Moreover, the common defaults in webcam pictures such as distortions and light or color problems are automatically handled by its integrated filter. Their integrated microphone could eventually be used to communicate with the robot with audible sounds. For SAUC'ISSE, we bought analog waterproof cameras ALLWAN AL-2121 that are connected to the embedded computer via audio-video to USB converters from Grabbino.

Depth

To get the depth of the submarines, we use a professional pressure sensor Keller PAA33X connected to the computer with a RS485 to USB converter. The sensor is fixed permanently on the back cover.

Inertial Measurement Unit

An IMU (Inertial Measurement Unit) Xsens MTi is used to get the orientation of the robot. It has a Kalman filter that uses magnetic data and gyroscopes to get an estimation of the orientation even in case of magnetic disturbances. It is connected to the embedded computer via a RS232 to USB converter. An MTi-G (the same, but with a GPS) is used in SARDINE.

Sonar

The sonar, MiniKing imaging sonar from Tritech, is used to get the position of the robot in the pool. . It is also connected to the embedded computer via a RS232 to USB converter. Figure 5 in the part about localization algorithms shows an example of sonar data.

Power supply

The power supply of SAUC'ISSE is divided into two parts. The power cards feeding the propellers are powered by a 12 V battery. The PC/104, the wireless access point (via the 5 V provided by the power supply PC/104 module) and the sonar are powered by a 24 V battery. All the other devices (pressure sensor, IMU, Labjack, webcams.) are powered via the USB ports of the computer. SARDINE has only one 12 V battery to power its thrusters, all the other devices are powered by the integrated battery of the EeePC, via the 5 V of its USB ports.

7.5 Communication

The ability to communicate with the robot when it is underwater is important since it allows having a feedback and eventually taking over the AUV in case of a problem. The problem is that electromagnetic waves don't pass through water. Despite acoustic waves being able to do so, acoustic communication is expensive and have low data transfer rate (several KB/s) which is not suitable for video transfer. During the tests of the AUV and data acquisition, many competitors choose to put an umbilical wire on their vehicle like in ROVs. Doing so requires a long cable which may hinder AUV movements and also requires a person to do cable management. In our case, since the depth of sinking is low, we build a buoy which is connected to the robot through an Ethernet cable and spreads the WIFI signal on the surface as seen in Figure 7.7.a. During the competition (see Figure 7.7.b), we use a shorter antennae which allows teleoperation when the robot is on the surface since it is forbidden to communicate with the robot when it is underwater.

A wireless access point DWL G700AP in combination with an external antenna of 1 m enables the robot to communicate with human operators when it is near the water surface. If the robot needs to be controlled at higher depths, the antenna is put on a buoy connected to the submarine with a wire of up to 5 m (using SMB Bulgin Buccaneer waterproof connectors with a RG174 cable).

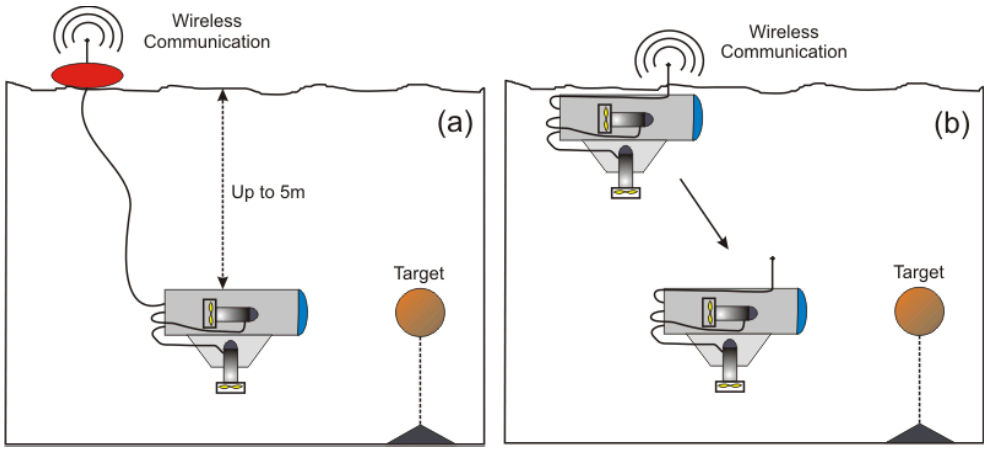

Figure 7.7: Communication with the robot

7.6 Autonomy and mission planning

We have several methods to do the competition tasks: some are interesting because they are simple to find or implement, some are a good compromise between simplicity, reliability and accuracy, some are useful in case a sensor is not available for any reason (hardware failure, perturbations...), other are challenging and could have an academic interest. Some of the methods allow controlling the movement of the robot; others allow enhancing the perception of the robot as computer vision algorithms or localization algorithms.

7.6.1 Basic movements and regulations

From the point of view of the control part of the submarine, we can consider all the competition tasks as a succession and combination between a depth, orientation and distance regulations.

Depth regulation

Due to the very slow dynamics of the robot, the depth regulation algorithm is very simple: it is a three state controller. If the submarine is below the desired depth, the vertical thruster is turned on at its maximal speed. If the submarine is near the desired depth, it is off. If the submarine is above, the thruster is turned on in the other direction. Depth regulation requires a depth sensor which is actually our pressure sensor.

Orientation regulation

Using two lateral thrusters we are able to perform orientation regulation by doing a control loop using heading measurements from the MTi. Even if sophisticated controllers can be applied to control such a nonlinear system [Fantoni and Lozano,], we have chosen to implement a simple PID (Proportional-Integral-Derivative) control law.

Distance and speed regulation

We use experimental methods to determine the dynamics of the robot (speed at different values of thrust, time to reach the maximum speed, time to stop after shutting down the thrusters ...). By mastering the dynamics of our robot we are able to be quite precise even controlling the robot in an open loop.

Another simpler method is to record reference line trajectories (1m, 2m, 3m....) and replay them to compose the desired trajectory. This method also proves to be very accurate if the water has no current.

7.6.2 Higher level algorithms

Localization algorithms

We use the same localization algorithm that was validated using data from Girona University. The robot is now capable to localize itself in real-time and follow a trajectory defined by waypoints. Figure 7.8 shows a trajectory of the SAUC'ISSE robot computed in real-time during pipeline inspection during the training session. We were the first team to discover the curved shape of the pipeline.

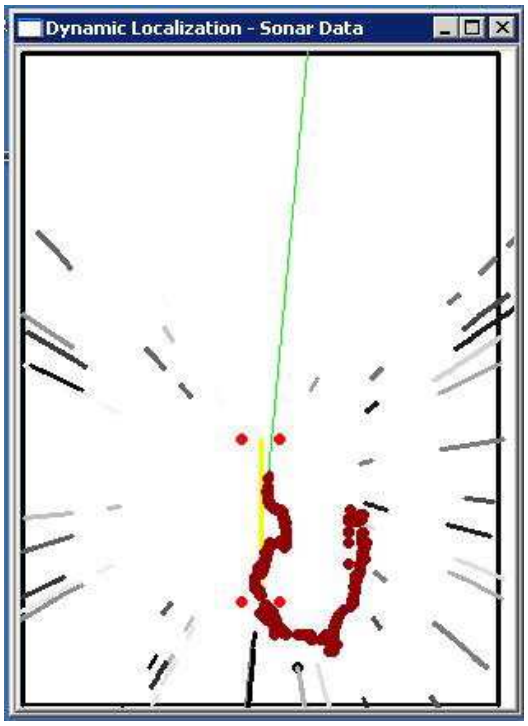

Figure 7.8: Localization of the SAUC'ISSE robot during he SAUC'E competition

Computer vision

As for computer vision, we are able to use many different methods. One of them is color detection in marine environments. In sea water, red color is more absorbed than blue color that is why if we put an orange buoy in water, its color will become bluer the farther we go. However, despite this color modification we can tell if the object was originally orange by using a formula which states that there are classes of equivalent colors (see [Bazeille, 2008]). Figure 6 illustrates the functioning of this formula.

We can also perform

- Simple shape detection (lines, circles, rectangles, assembly of those elements...) using Hough Transform (see [Russ, 2002] [Bovik, 2000]) or its interval equivalent [Jaulin and Bazeille, 2009 useful for pipeline and ball detection.

- Optical flow algorithm [Shi and Tomasi, 1994] which consist in detecting the same points of interest in two consecutive images allowing for example to compute the speed of the robot using a downward looking camera (sea floor have to be visible).

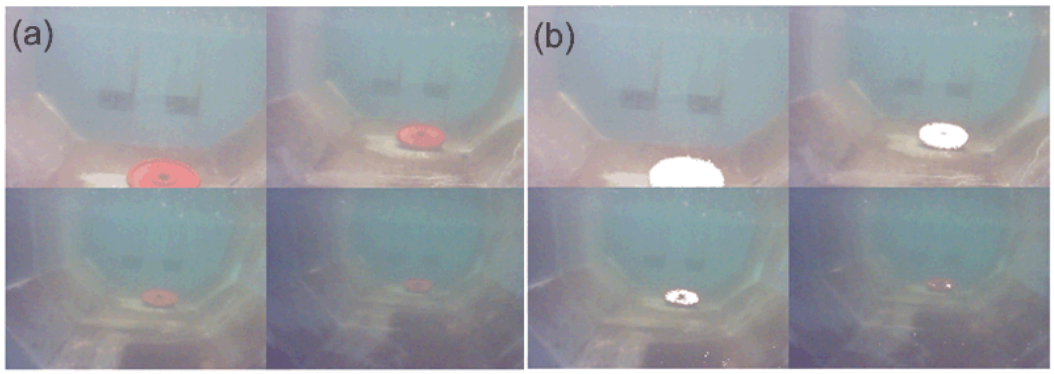

Figure 7.9: Detecting objects by its color taking color absorbtion into consideration

7.7 Autonomy through script

Higher level actions such as going to a point specified by coordinates, following a trajectory, searching for the submarine buoy, hitting the submarine buoy, wall following... can be performed by applying the algorithms using sensor data and then executing basic movements according to the result of those algorithms. Those actions are implemented in the intelligent program of the robot (using C/C++). The global mission that the robot has to execute during the competition will be a succession of such actions. The global mission is not specified in the program but it is the user who writes those specifications in form of a script. This enables to change the mission quickly without having to recompile the program. Figure 7.10 illustrates an example of such script.

7.8 Software architecture

Having good software architecture is critical if one want a functioning robot. In fact, such architecture allows to have a source code which is

- **Clear and understandable**: Each year the SAUC'E competition members change. That is why the faster new members can master the code the more time they will have to improve it.

- **Modular**: Having a modular code allow splitting the job of coding on many people. 1 Man – 1 Module.

- **Easy to modify and evolve**

- **Practical** to use: user friendly Human-Machine Interface, Log files....

Figure 7.11 shows an overview of our software architecture.

```
% this is a commentary. It is ignored by the program
% Example 1: go to depth -3m
%% step1: start depth regulation
depthreg -3
%% step2: wait 15s for the robot to actually get there
wait 15
% Example 2: some random movements at -3m
heading 1
wait 5
thrust 1
wait 10
stop
heading 1.57
wait 10
% Example 3: high level action
configureWallFollowing 12.5 1.2 -1.57 0 0.3 0.2 0.5
startWallFollowing
wait 20
stopWallFollowing
generalstop
depthreg 0
```

Figure 7.10: Mission execution script

7.9 Robot swarm

Since 2010 we started the project of using several collaborating robots to accomplish tasks autonomously. The idea is to build many robots which are not necessary hardware clones but rather complementary. As an example we could have a robot using only sonar and another one using only cameras. Why collaborating? Imagine you have to park your car in a very narrow space. The job would be a lot easier if you had the help of a friend who would be your "eye". In 2010 edition of SAUC'E we built the new robot SARDINE which is a camera robot which first purpose was to film our first submarine SAUC'ISSE. The use of a swarm of complementary robots has another major advantage. In case of failure of one of the robots, we can still continue the mission using the remaining robots while we would be in pinch if we had only one highly advanced robot that doesn't work. This situation actually happened to us during the competition qualifications where SAUC'ISSE failed due to power supply problems. If we didn't have the other robot, we wouldn't have qualified.

7.10 Conclusion

In this chapter we showed an example of autonomous underwater robot design. The final design is the fruit of many years of experience but also thanks to the existence of

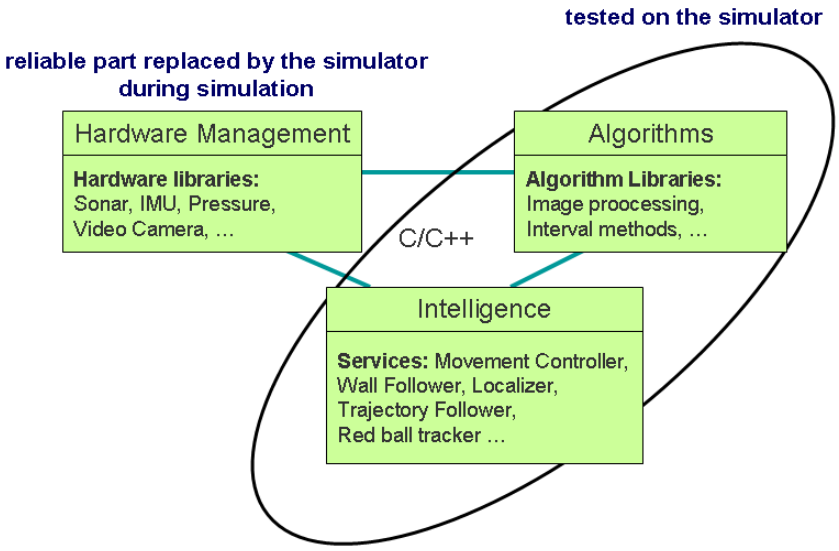

Figure 7.11: Sauc'isse software architecture

competitions such as SAUC'E without which our robots might never have existed.

Chapter 8

Other robotics activities at ENSTA-Bretagne

During this PhD thesis I was involved in many robotic activities. First, I was part of the robotics club which purpose is to popularize robotics within the student body. In the club, we make several presentations, propose projects, make introductionary courses in control, image processing. We develop underwater, sailing, mobile and flying robots. The mobile robots (see Figure 8.1 on the left) are developed in the context of the CAROTTE (CArtographie par RObot d'un TErritoire) Challenge (as open candidate). The purpose is to make a swarm of robots which purpose is to explore an unknown indoor environment (see Figure 8.1 on the right) and return its map and the location of specific objects of interest. The robot might be hindered by obstacles (visible/transparent), smokescreens and traps. In the future, we ought to participate in the EuroBot competition.

Underwater robotics aside, one of the robotic field we were more implicated in was the autonomous sailing robotics. The objective of the project is to build autonomous sailing

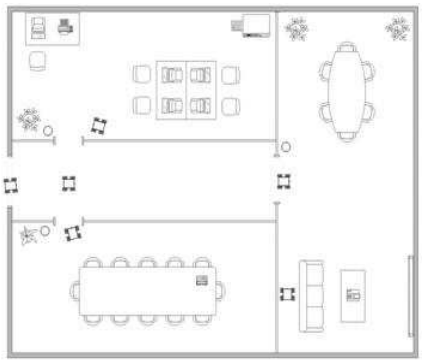

Figure 8.1: Mobile robots for CAROTTE Challenge.

Figure 8.2: The autonomous sailboat "Breizh Spirit" and the first challenge : crossing Brest harbour.

robots which purpose is long last navigation on the ocean. The main motivation for developing such robots was to participate in the MicroTransat challenge which purpose is to cross the Atlantic ocean in complete autonomy. The robot must send its position (through satellite communication) every 24 hours so that the judges can validate the crossing. The robot must be small and light and safe (visible the day and the night by both vision and radar).

I participated to the realization of the first prototype "Breizh Spirit" represented on the left of Figure 8.2 with which we crossed the Brest Harbour. The GPS trace is represented on the right of Figure 8.2. Three conference publications were published during the project [Sliwka et al., 2009][Sliwka et al., 2011b][Xiao et al., 2011].

Being implicated in the robotics club was a valuable experience in all different aspects be it technical (mechanics, electronics, computer science), logistic (organizing robot tests, buying the different components, organizing club activities), human (interacting with students, teachers and administrative body, management, politics) and so on.

Chapter 9

Conclusion

This PhD thesis introduced several methods to deal with relaxed constraint satisfaction problems *i.e.* CSP where not all the constraints have to necessarily be satisfied. Several contributions to contractor theory were also presented. One of them is the image contractor which enables to represent hardly parametrable sets (such as maps or shore lines). The image contractor allows to include constraints such as *"a point belong to the map"* in a CSP/relaxed CSP formalism. The image contractor uses a special transform of the binary image called the ψ-transform which is computed once and stored on the computer in the form of a grayscale image. The main drawback of using an image contractor is the storage space used for the ψ-transform which exponentially grows with the dimension of the contractor. Compressing the ψ-transform would be one of the prospects of the research.

The methods described above has been used in the context of underwater robot localization. The localization problem when there are outliers can be cast into a relaxed CSP where the position of the robot is the unknown. Relaxed CSP solvers defined in the manuscript are used to compute the position of the robot. On the other hand, the map of the environment is represented using a binary image from which the image contractor is build. In a nutshell, using the newly developed tools, we propose an approach enabling robust to outliers robot localization in any unstructured environment.

The solution of a relaxed CSP can be represented in two forms. The first form of the solution of a relaxed CSP is a polynomial with set valued coefficients otherly known as set polynomials. This form takes benefit from polynomial arithmetics to describe the relaxation process *i.e.* how to obtain the solution set polynomial from the set of constraints in the relaxed CSP. This representation of the solution allows to make distributed computations. The result of distributed relaxed CSP computations which are in the form of set polynomials is the *product* of those set polynomials.

The second form of the solution of a relaxed CSP is a function called accumulator which

returns the number of constraints satisfied by an element of the search space of the relaxed CSP. This representation of the solution also allows to make distributed computations. The result of distributed relaxed CSP computations which are in the form of accumulators is the *sum* of those accumulators.

The accumulators and set polynomials based solvers are also explained in this manuscript. There is still a lot of room to optimize and develop those solvers. As an example, the solver based on set polynomials is only developed when the real solution can be approximated by a single box polynomial (it is the case of punctual solutions). In the case of more complex sets, algorithms inspired from the branch and prune algorithm which uses box polynomials instead of simple boxes could be defined. In the case of solvers based on accumulators, the search space is discretized in the form of a grid of boxes and accumulator's inclusion function is evaluated for each box of the grid. The result is stored on the computer. The drawback of this method is the memory usage to store the discretized accumulator function especially when the dimension is high. A possible idea to overcome this problem would be to use weighted regular subpaving instead of a grid of boxes to discretize the search space.

Finally, the methods presented here were used to solve several localization problems but could be used to solve more challenging problems such as SLAM problems which have high dimension. Three years might appear long for a PhD thesis but as one can say eating whets the appetite. That said, the more you explore the more leads you find, and the more leads you find the more you need to explore. But everything has to come to an end.

THE END

Chapter 10

French Summary

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE
Mention : Robotique
École Doctorale SICMA

présentée par
Jan Śliwka
préparée à l'ENSTA Bretagne (ex ENSIETA), Equipe OSM, Pôle STIC

Méthodes ensemblistes pour une localisation robuste de robots sous-marins

Soutenue le 6 décembre 2011
devant le jury composé de :

Luc JAULIN
Professeur des Universités, ENSTA Bretagne / *directeur de thèse*

Olivier REYNET
Maître de Conférence, ENSTA Bretagne / *co-directeur de thèse*

Laurent HARDOUIN
Professeur des Universités, Université d'Angers ISTIA / *rapporteur*

Bruno JOUVENCEL
Professeur des Universités, Université de Montpellier / *rapporteur*

Rogelio LOZANO
Directeur de recherche, CNRS / *président du Jury*

Table des matières

Table des figures

Chapitre 1

Résumé en langue française de la thèse

1.1 Localisation de robots sous-marins

Pour qu'un robot autonome puisse interagir proprement avec son milieu, ce dernier doit connaitre d'une part l'environnement dans lequel il évolue et d'autre part son état dans cet environnement. En particulier, un robot doit savoir où il est pour savoir où il doit aller. Utilisant plusieurs capteurs et connaissant la carte de l'environnement, le robot calcule sa position et son orientation avec des algorithmes de localisation plus ou moins complexes. Depuis l'apparition du GPS, le problème de la localisation a été pratiquement résolu pour les robots terrestres. Le GPS ne fonctionne pas sous l'eau car les ondes électromagnétiques haute fréquence ne s'y propagent pas bien. Toutefois, le nombre d'opérations sous-marines augmenter de manière significative chaque année. Cela est dû au développement des domaines tels que l'exploitation des ressources (par exemple l'inspection des structures offshore en eau profonde), l'océanographie, la biologie, l'exploration des épaves, la sécurité (la surveillance des ports) et le domaine militaire (le déminage, la pose de mines). Compte tenu de la dangerosité de ces opérations, il y a un besoin croissant de robots sous-marins. Pour effectuer ces missions, on a souvent recours à des véhicules téléopérés aussi appelés ROVs (Remotely Operated Vehicles). Les ROVs sont connectés à la station de contrôle par le biais d'un câble ombilical utilisé pour la transmission d'énergie et la communication. Ils nécessitent donc une infrastructure importante (surtout pour les modèles haute profondeur).

L'alternative est d'utiliser des véhicules sous-marins autonomes aussi appelés AUV (Autonomous Underwater Vehicle) (Voir [Veres et al., 2008] pour des designs d'AUV). Les AUVs n'ont besoin d'aucun câble ombilical. Cependant, cela rends la communication avec le centre de contrôle est limité et notamment il est impossible de transmettre de

l'image. La téléopération d'un AUV devenant difficile, ce dernier doit être intelligent et s'appuyer sur des algorithmes avancés tels que la localisation, le SLAM (localisation et cartographie simultanées), traitement d'image et la planification de trajectoire. L'AUV (d'habitude trés cher) doit également être en mesure de rentrer à la base de façon autonome après la mission [Baccou and Jouvencel, 2002]. De nombreuses approches ont été proposées pour résoudre ces problèmes. Il y a les méthodes probabilistes [Thrun et al., 2005] [Clérentin et al., 2008]. Dans ce manuscrit, les méthodes probabilistes ne sont pas présentées. En effet, ce manuscrit se focalise sur les améliorations apportées aux méthodes ensemblistes et en particulier sur les méthodes utilisant le calcul par intervalles.

Le problème de la localisation du robot en utilisant les méthodes ensemblistes ont été abordés par de nombreux auteurs [Meizel et al., 1996] [Halbwachs and Meizel, 1996] [Gning, 2006] [Kieffer, 1999] dans le cas où le problème est linéaire ou non et aussi [Caiti et al., 2002] lorsque le robot est sous l'eau. Le calcul par intervalle se montre bien utile pour résoudre des problèmes fortement non-linéaires (voir, par exemple, [Meizel et al., 2002], où la première localisation d'un robot réel a été résolu avec des méthodes d'intervalle). Le principe est que les données sur l'environnement (issus des capteurs) ainsi que l'état du robot (position, orientation) sont représentés par des domaines d'appartenance. Les contraintes (équations) entre ces différentes variables permettent de réduire la taille du domaine d'appartenance l'état du robot ainsi augmentant la précision de la localisation. Un autre point fort des méthodes ensemblistes c'est la prise en compte des données aberrantes [Jaulin and Walter, 2002] [Jaulin et al., 1996]. À titre d'exemple, cette propriété a été utilisée pour améliorer la localisation en ville à l'aide d'un capteur GPS dont les données sont généralement corrompus par les trajets multiples [Drevelle and Bonnifait, 2009]. Ces méthodes ont également été utilisés pour la localisation robustes par rapport aux données aberrantes de robots sous-marins [Jaulin, 2009].

Dans cette thèse, nous nous sommes focalisé sur le traitement des données aberrantes lorsque leur nombre est inconnu et peut varier avec le temps. Nous présentons également d'autres contributions aux méthodes ensemblistes et particulièrement la théorie des contracteurs. L'utilité de la plupart de ces contributions est montré à travers de différents exemples de problèmes de localisation.

1.2 L'approche utilisée pour résoudre le problème de localisation

Tout problème de localisation implique quatre concepts étant l'environnement, la carte de l'environnement, la pose (position et orientation) et les mesures de l'environnement. Pour chaque mesure de l'environnement est associée une équation (ou un ensemble d'équations) reliant la carte, la pose et la mesure. Ainsi, le problème de la localisation peut être formulé

comme un problème de satisfaction de contraintes ou CSP. Un CSP peut être vu comme un ensemble d'équations (ou contraintes) impliquant des variables à déterminer (dans le cas de la localisation, la position du robot est une telle variable). Chacune de ces variables est connue pour appartenir à un ensemble connu appelé domaine ou l'espace de recherche. Dans notre cas, chaque contrainte peut être considérée comme une représentation de l'information sur la position du robot. Cette information est une compilation des mesures que l'on fait de l'environnement et la connaissance que l'on a de cet environnement (La carte). Plus il y a de contraintes, plus il y a d'informations sur la pose du robot et meilleur est son estimation. La solution d'un CSP est l'ensemble des points (poses) qui satisfont toutes les contraintes. Ceci implique que toutes les mesures sont consistantes (non aberrantes). Une mesure de l'environnement réel est consistante si celle ci correspond à la mesure théorique d'un capteur modèle dans l'environnement (modèle) représenté par la carte.

Le capteur principal utilisé pour la localisation de notre sous-marin est un sonar sectoriel. Ce capteur donne souvent des mesures aberrantes. Une telle mesure peut être dû à une défaillance électrique du capteur ou d'un phénomène non pris en compte lors de la modélisation de l'environnement (objets non prévus qui sont détectées par le sonar, des phénomènes physiques tels que des échos multiples ...).

Un problème de localisation avec données aberrantes peut être mis sous la forme de ce que nous appelons un CSP relaxé c'est à dire un CSP où pas toutes les contraintes doivent être satisfaites. Le nombre de données aberrantes dans un jeu de mesures utilisés pour la localisation est a priori inconnu et peut varier avec le temps. Une meilleure caractérisation de la solution d'un CSP relaxé permet une meilleur prise en compte de ces valeurs aberrantes. La caractérisation et le calcul de la solution d'un CSP relaxé est l'une des contributions majeures à cette thèse.

Un des obstacles à surmonter pour résoudre les problèmes de localisation est la représentation de la carte. En cas d'environnements structurés, il est possible de représenter la carte par un ensemble d'objets paramétrés tels que des segments, polygones, courbes. En cas d'environnements non structurées où en partie structurées tels que des cartes marines ou des cartes routières, l'idée est de représenter la carte (qui est en fait un ensemble de points) sous la forme d'une image binaire où les pixels d'intérêt (noir par exemple) représentent l'ensemble des points de la carte. Un des contributions majeurs de la thèse était d'incorporer une telle représentation de la carte dans le formalisme d'un CSP ou d'un CSP relaxé.

Dans ce manuscrit, nous proposons donc une approche permettant la localisation du robot robuste par rapport aux données aberrantes et dans le cas ou la carte peut être non structurée.

1.3 Contributions

Cette section résume les contributions de la thèse.

1.3.1 Représentation d'une solution d'un CSP relaxé

Durant la thèse, nous avons travaillé sur la façon de caractériser la solution d'un problème de satisfaction de contraintes relaxé à savoir un CSP où certaines des contraintes peuvent être laissées insatisfaites (Voir la sous-section 1.3.1 pour la définition).

Intervalles et boites

Intervalle : Un intervalle réel est un sous-ensemble connexe et fermé de $\mathbb{R}$. Si x est un réel, on notera par $[x]$ l'intervalle qui contient cette variable. $[x]$ est appelé domaine de x. Un intervalle $[x]$ a des bornes inférieure et supérieures notées respectivement par x^- et x^+. $\mathbb{IR}$ est l'ensemble de tous les intervalles réels. $\mathbb{IN}$ est l'ensemble des intervalles de nombres entiers. $w([x]) = x^+ - x^-$ est appelé longueur de $[x]$.

Exemple 1 $\emptyset, \{-1\}, [-1, 1], [-1, \infty], \mathbb{R}$ *sont des intervalles de* $\mathbb{R}$.

Intervalle vectoriel (ou boite) : une boite de $\mathbb{R}^n$ est un produit cartésien d'intervalles. Une boite peut être aussi considérée comme un vecteur d'intervalles ou aussi d'un intervalle de vecteurs. Si $\mathbf{x} = (x_1, .., x_n) \in \mathbb{R}^n$ est un vecteur de réels on note par $[\mathbf{x}] = ([x_1], .., [x_n])$ la boite qui contient cette variable.

Exemple 2 $[1, 3] \times [2, 4]$ *est une boite de* $\mathbb{R}^2$.

Il existe une arithmétique similaire à l'arithmétique des réels mais s'appliquant aux intervalles appelée arithmétique intervalle. Grace à cette arithmétique, il est possible d'effectuer des opérations binaires (sommes, produits,...) sur des intervalles ainsi que calculer l'image par une fonction d'un ou plusieurs intervalles.

L'arithmétique par intervalle est bien adapté à des calculs sur ordinateur. En effet, celle ci a été développée dans les années 50 pour faire des calculs en prenant en compte les incertitudes sur les variables utilisés. Pour cette thèse, l'arithmétique par intervalle est utilisée pour implémenter des algorithmes de résolution de CSP relaxés.

Il est possible d'étendre la notion d'intervalles à des ensembles plus complexes comme les intervalle de fonctions [Le Bars et al., 2011] ou les intervalles d'ensembles ou de sous-pavages [Jaulin, 2011].

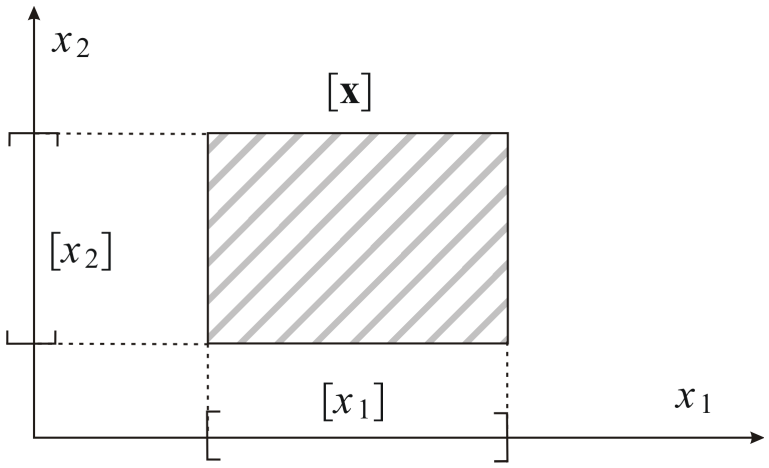

Figure 1.1 – Exemple d'une boite

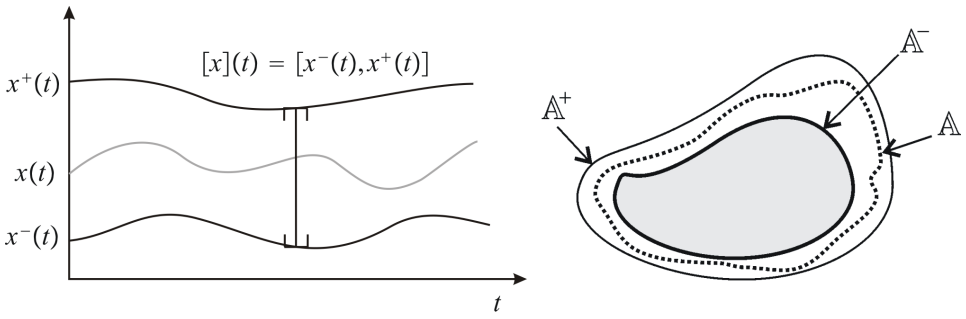

Figure 1.2 – Autres types d'intervalles

CSPs et CSPs relaxés

Un problème de satisfaction de contraintes (ou CSP) est défini par un ensemble de contraintes $C_1, .., C_n$, un vecteur de variables $\mathbf{x} = (x_1, .., x_m)$ et un domaine (ou espace de recherche) $\mathbf{D} = D_1 \times .. \times D_m$ de toutes les valeurs possibles de $\mathbf{x}$. A l'origine, les CSPs ont été étudiés pour les domaines discrets [Clowes, 1971] [Waltz, 1975]. Plus tard, l'étude des CSPs a été étendue pour les domaines continus [Cleary, 1987] [Davis, 1987] [Hyvönen, 1992] [Sam-Haroud, 1995]. Pendant cette thèse, nous avons considéré que les CSPs continus ou $\mathbf{D}$ est une boite de $\mathbb{R}^m$. Pour cela nous changeons la notations de $\mathbf{D}$ en $[\mathbf{x}] = ([x_1], .., [x_n])$.

Les contraintes sont des equations linéaires ou non linéaires

$$\begin{array}{l} g_i : \mathbb{R}^m \rightarrow \mathbb{R}, h_i : \mathbb{R}^m \rightarrow \mathbb{R} \\ C_i : g_i(\mathbf{x}) \leq 0, i = 1, .., k \\ C_i : h_i(\mathbf{x}) = 0, i = k+1, .., n. \\ \mathbf{x} \in [\mathbf{x}], \end{array} \tag{1.1}$$

Une notation plus générales peut être utilisée pour représenter un tel CSP

$$\begin{array}{l} \mathbf{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{p_i} \\ C_i : \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], \\ \mathbf{x} \in [\mathbf{x}], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \end{array} \tag{1.2}$$

ou $[y_i]$ sont des intervalles réels connus.

Le CSP peut être noté sous un forme encore plus compacte

$$\begin{array}{l} \mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n \\ \mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}], [\mathbf{y}] \in \mathbb{IR}^n. \end{array} \tag{1.3}$$

ou $[\mathbf{y}]$ est une boite connue de $\mathbb{R}^n$.

Toutes ces notations sont équivalentes et sont utilisées dans le manuscrit.selon la situation. Chercher la solution d'un CSP est de chercher l'ensemble d'éléments $\mathbf{x} \in [\mathbf{x}]$ qui satisfont toutes les contraintes. Parfois, un tel élément n'existe pas. Un CSP ou toutes les contraintes ne sont pas nécessairement satisfaites est appelé un *problème de satisfaction de contraintes relaxé* ou *CSP relaxé*.

Relaxer un nombre fixe de contraintes

Une première méthode de résolution d'un CSP relaxé a été introduite dans [Jaulin, 2009] et consiste à chercher l'ensemble $\mathbb{S}_q$ d'éléments qui satisfont toutes les contraintes $\{C_1, .., C_n\}$

sauf (en relaxant) $q \in \{1, .., n\}$ d'entre elles. Cela signifie que jusqu'à q contraintes peuvent ne pas être satisfaites. Cela signifie également que $\mathbb{S}_q$ est aussi l'ensemble des points qui satisfont au moins $n - q$ contraintes. Notez qu'il ya une équivalence entre les formulations "relaxer q contraintes" et "satisfaire $n - q$ contraintes». La première formulation est généralement utilisée lorsque le nombre de contraintes à relaxer (ou des contraintes inconsistentes) est faible. Tel est le cas du CSP relaxé correspondant au problème de localisation. Dans ce cas, les contraintes inconsistentes sont causées par des valeurs aberrantes dans les mesures du capteur qui ont un ratio inférieur à 30%. L'ensemble $\mathbb{S}_q$ est exprimé par la formule suivante

$$\mathbb{S}_q = \{\mathbf{x} \in [\mathbf{x}], \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}. \quad (1.4)$$

On note $\mathbb{X}_i$ l'ensemble des éléments qui satisfont la contrainte C_i

$$\mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i]\}, [\mathbf{y}_i] \in \mathbb{IR}^{p_i}, i \in \{1, .., n\}. \quad (1.5)$$

Les ensembles $\mathbb{X}_i$ jouent un rôle important dans la caractérisation des différentes solutions du CSP relaxé. Par exemple, l'ensemble des points satisfaisant toutes les contraintes (la solution du CSP) est l'ensemble des points qui appartiennent à tous les ensembles $\mathbb{X}_i$ et donc l'intersection des ensembles $\mathbb{X}_i$. De la même manière, l'ensemble $\mathbb{S}_q$ est défini comme étant une intersection spéciale appelée intersection relaxée des ensembles $\mathbb{X}_i$. L'intersection relaxée des ensembles $\mathbb{X}_i, i \in \{1, .., n\}$ notée $\bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i$ est l'ensemble des points qui appartiennent à tous les ensembles sauf q d'entre eux. Ainsi

$$\mathbb{S}_q = \bigcap_{i \in \{1,..,n\}}^{\{q\}} \mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{K} \subset \{1, .., n\}, card(\mathbb{K}) = n - q, \forall i \in \mathbb{K}, \mathbf{x} \in \mathbb{X}_i\}. \quad (1.6)$$

Exemple 3 *La Figure 1.3 illustre la q-intersection de 5 ensembles* $\mathbb{X}_1, .., \mathbb{X}_5$. *on a*

$$\begin{aligned} &\bigcap_{i \in \{0,..,5\}}^{\{0\}} \mathbb{X}_i = \bigcap_{i \in \{1,..,5\}} \mathbb{X}_i = \emptyset \\ &\bigcap_{i \in \{0,..,5\}}^{\{2\}} \mathbb{X}_i = \mathbb{X}_3 \cap \mathbb{X}_4 \cap \mathbb{X}_5 \ \textit{(la partie hachurée dans Fig. 1.3)} \\ &\bigcap_{i \in \{0,..,5\}}^{\{5\}} \mathbb{X}_i = \bigcup_{i \in \{1,..,5\}} \mathbb{X}_i. \end{aligned} \quad (1.7)$$

Dans le cas de la localisation d'un robot, le nombre réel des contraintes inconsistantes q_{real}, du a des données aberrantes , est généralement inconnu et peut varier avec le temps.

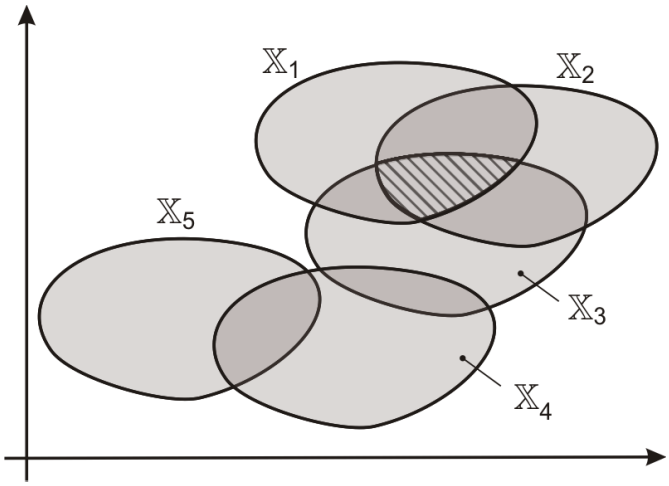

Figure 1.3 – Illustration de l'intersection q-relaxée de 5 ensembles. L'ensemble hachuré correspond à l'intersection 2-relaxée

Il est parfois possible de supposer un nombre maximum de contraintes inconsistantes $q_{\max}$. Nous considérons $\mathbb{S}_{q_{\max}}$ la solution du problème en supposant que le nombre de contraintes non satisfaites ne dépasse pas $q_{\max}$. L'ensemble $\mathbb{S}_{q_{\max}}$ est une solution garantie à savoir la position réelle du robot se trouve certainement dans $\mathbb{S}_{q_{\max}}$ tant que le nombre réel des valeurs aberrantes q_{real} est inférieur $q_{\max}$. Notez que, puisque $q_{\text{real}} < q_{\max}$, nous avons $\mathbb{S}_{q_{\text{real}}} \subset \mathbb{S}_{q_{\max}}$ ce qui signifie que l'ensemble solution $\mathbb{S}_{q_{\max}}$ est surestimé. Les formulations de la solution d'un CSP relaxé exliqué dans les sous-sous sections suivantes font abstraction de ce problème.

Le tri d'ensembles et la notation polynomiale

L'une des contributions à la thèse de doctorat a été d'essayer de trouver une nouvelle représentation de la solution d'un CSP relaxé. L'approche consiste à considérer tous les ensembles de la solution $\mathbb{S}_q$ pour toutes les valeurs possibles de q. La première représentation mathématique de la solution d'un CSP relaxé est donc un vecteur d'ensembles $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$. Pour caractériser ce vecteur nous avons considéré une transformée notée $\mathcal{T}$ que nous avons appelé le tri d'ensembles tel que le vecteur $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ soit la transformée du vecteur d'ensembles $(\mathbb{X}_1, .., \mathbb{X}_n)$ définis dans (1.5). On a

$$(\mathbb{S}_{n-1}, .., \mathbb{S}_0) = \mathcal{T}\ (\mathbb{X}_1, .., \mathbb{X}_n). \tag{1.8}$$

Exemple 4 *Considérons un vecteur d'ensembles* $(\mathbb{X}_1, .., \mathbb{X}_6) \subset \mathbb{R}^2 \times ... \times \mathbb{R}^2$ *representé sur la Figure 15.a. On note par by* $(\mathbb{Y}_1, ..\mathbb{Y}_6) \subset \mathbb{R}^2 \times ... \times \mathbb{R}^2$ *le résultat du tri ensembliste du vecteur* $(\mathbb{X}_1, ..\mathbb{X}_6)$. *Le vecteur trié est représenter sur la Figure 1.4.b. On remarque que*

$\mathbb{Y}_5 = \emptyset$ *et* $\mathbb{Y}_6 = \bigcap_{i \in \{1,..,6\}} \mathbb{X}_i = \emptyset$.

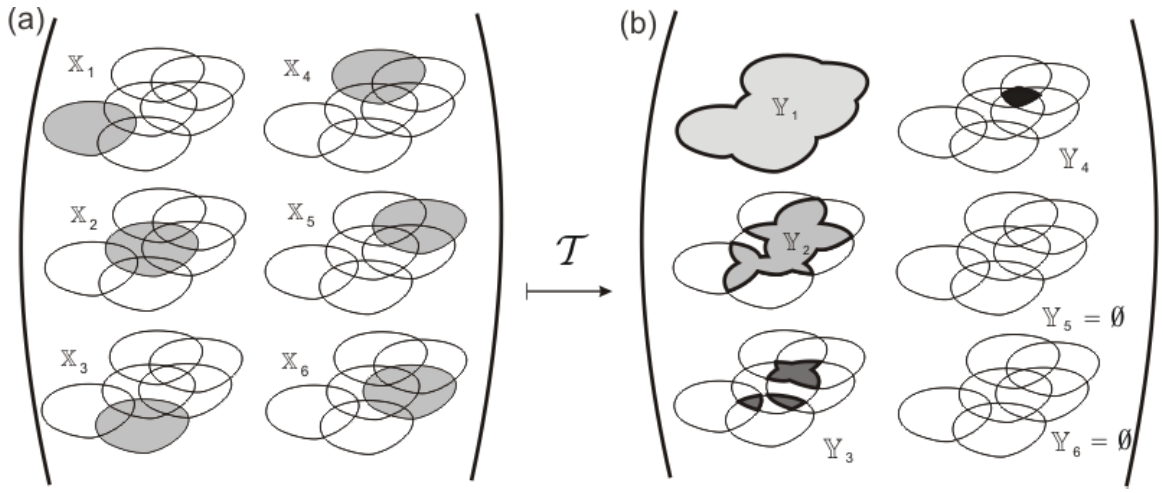

Figure 1.4 – Illustration du tri de 6 ensembles

L'appellation tri ensembliste de la transformée $\mathcal{T}$ vient du fait que le vecteur de sortie est trié par ordre décroissant par rapport à l'ordre d'inclusion $\subset$ c'est à dire $\mathbb{S}_0 \subset ... \subset \mathbb{S}_{n-1}$. Afin d'obtenir le vecteur trié, il existe un formule pour chaque ensemble de solutions $\mathbb{S}_q, q \in \{1, .., n\}$ qui comme nous l'avons déja dit correspond à l'intersection q-relaxée des ensembles $\mathbb{X}_i, i \in \{1, ., n\}$.

Dans ce manuscrit, nous montrons qu'il y a une manière plus compacte de représenter le tri ensembliste ainsi que la solution dans une même formule. La solution que nous proposons est d'utiliser une représentation polynomiale. L'idée est de considérer les polynômes avec des coefficients sous forme d'ensembles qu'on appelle les polynômes ensemblistes. La solution du CSP relaxé sera représenté sous la forme d'un polynôme ensembliste dont les coefficients sont les éléments du vecteur $(\mathbb{S}_{n-1}, .., \mathbb{S}_0)$ tels que le degré du coefficient correspond au nombre de contraintes qui sont satisfaits par les éléments de chaque coefficients. Notons $X^*(s)$ ce polynôme.

$$X^*(s) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{1.9}$$

Le but de l'utilisation de la représentation polynomiale est de profiter de l'arithmétique des polynômes (somme, produit) pour représenter le tri ensembliste avec une formulation unique. Le produit et la somme de polynômes ensemblistes est similaire au produit et la somme de polynômes réels à la différence que le produit " $*$ " de deux coefficients est remplacé par leur intersection " $\cap$ "et la somme " $+$ " de deux coefficients est remplacée

par leur union " $\cup$ ". On considère le polynôme suivant

$$Y^*(s) = \prod_{k=1}^{n} (\mathbb{X}_k s + \mathbb{R}^m) \tag{1.10}$$

où $\mathbb{X}_i, i \in \{1, .., n\}$ sont les ensembles définis dans (1.5). En développant ce polyôme on peut prouver que

$$\prod_{k=1}^{n} (\mathbb{X}_k s + \mathbb{R}^m) = \sum_{k=0}^{n} \mathbb{S}_{n-k} s^k. \tag{1.11}$$

Ainsi, le tri ensembliste $\mathcal{T}$ a une formulation polynomial equivalente sous la forme d'un produit de monômes $(\mathbb{X}_k s + \mathbb{R}^m), k \in \{1, .., n\}$ qu'on appelera le tri ensembliste polynomial du vecteur $(\mathbb{X}_1, .., \mathbb{X}_n)$.

Remarque 1 *Une interprétation possible des monômes* $\mathbb{X}_i s + \mathbb{R}^m, i \in \{1, .., n\}$ *est de représenter les informations binaires "*{appartient à $\mathbb{X}_i$, appartient à $\mathbb{R}^m$ (aucune information)}*". La variable* s *du polynôme est associée à l'information "*appartient à $\mathbb{X}_i$*". Cela signifie que, sachant que* $X^*(s)$ *est un produit de monômes (équation (1.11)), plus grand est le nombre d'ensembles* $\mathbb{X}_i$ *auxquels un élément* $\mathbf{x}$ *appartient, plus élevé est le degré du coefficient du polynôme* $X^*(s)$ *auquel* $\mathbf{x}$ *peut appartenir.*

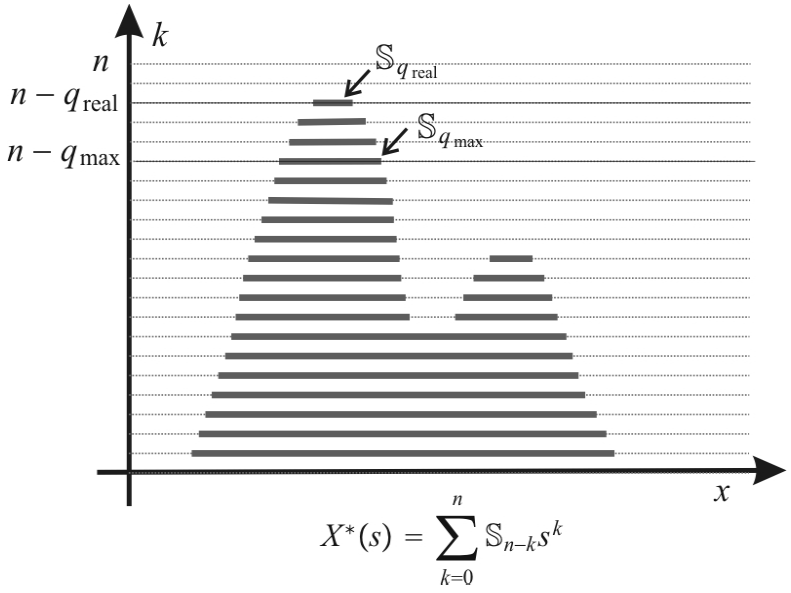

Figure 1.5 – Exemple de représentation d'un polynôme ensembliste solution d'un CSP à une dimension

la Figure 1.5, représente le polynôme ensembliste solution d'un CSP relaxé dont l'inconnue x est à une dimension. La solution est représentée sous forme d'intervalles "empilés" les

uns sur les autres. Plus un coefficient intervall est haut, plus son degré dans le polynôme est grand. L'axe k correspond donc au nombre de contraintes qui sont satisfaites par les elements du coefficient placé à ce niveau. l'axe x designe les valeurs possibles que peut prendre la variable x. On note par q_{real} le nombre réel de données aberrantes et $q_{\max}$ l'estimation qu'on fait de ce nombre. On remarque que $\mathbb{S}_{q_{\text{real}}} \subset \mathbb{S}_{q_{\max}}$.

L'ensemble des sous-ensembles de $\mathbb{R}^n$ a une structure de treillis [Davey and Priestley, 2002]. Un treillis est un ensemble partiellement ordonné fermée où toute paire d'éléments a une borne supérieure et une borne inférieure. Durant la thèse, l'étude des polynômes ensemblistes a été généralisée par l'étude des polynômes sur un treillis où les coefficients appartiennent à un treillis.

Pour conclure, une représentation de solution sous forme de polynôme ensembliste nous permet donner une solution d'un CSP relaxé indépendamment du nombre de contraintes non satisfaites ce qui était le résultat voulu. L'implémentation des polynômes ensemblistes est détaillée dans le manuscrit de thèse.

Les accumulateurs

Il existe un représentation alternative de la solution d'un CSP relaxé sous la forme d'une fonction $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$ appelée *accumulateur* qui pour chaque élément $\mathbf{x} \in [\mathbf{x}]$ associe le nombre de contraintes qu'il satisfait

$$\mathcal{A}(\mathbf{x}) = Card(\{i \in \{1, .., n\}, \mathbf{f}_i(\mathbf{x}) \in [\mathbf{y}_i], [\mathbf{y}_i] \in \mathbb{IR}^{p_i}\}). \tag{1.12}$$

L'accumulateur peut aussi être défini en utilisant la fonction caractéristique χ d'une contrainte $C : \mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{f} : \mathbb{R}^m \to \mathbb{R}^p, [\mathbf{y}] \subset \mathbb{R}^p$ qui est définie par

$$\begin{cases} \chi(C)(\mathbf{x}) = 1 \text{ si } \mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \\ \chi(C)(\mathbf{x}) = 0 \text{ sinon.} \end{cases} \tag{1.13}$$

En fait, l'*accumulateur* $\mathcal{A}$ est la somme des fonctions caractéristiques de toutes les contraintes du CSP relaxé.

$$\mathcal{A}(\mathbf{x}) = \sum_{i \in \{1,..,n\}} \chi(C_i)(\mathbf{x}). \tag{1.14}$$

Cette dernière propriété justifie le nom de "*accumulateur*". La Figure 1.6, représente un accumulateur $\mathcal{A}$ en tant que solution d'un CSP à une dimension avec n contraintes. La variable $x \in \mathbb{R}$ est représentée sur l'axe des x et l'axe k correspond au nombre de contraintes qui sont satisfaits par x. On note par q_{real} le nombre réel de données aberrantes et $q_{\max}$ l'estimation qu'on fait de ce nombre. Les ensembles $\mathbb{S}_{q_{\text{real}}}$et $\mathbb{S}_{q_{\max}}$ sont représentés sur la Figure 1.6. On note que

$$\forall q \in \{1, .., n\}, \mathbb{S}_q = \{x \in \mathbb{R}, \mathcal{A}(x) \geq n - q\}. \tag{1.15}$$

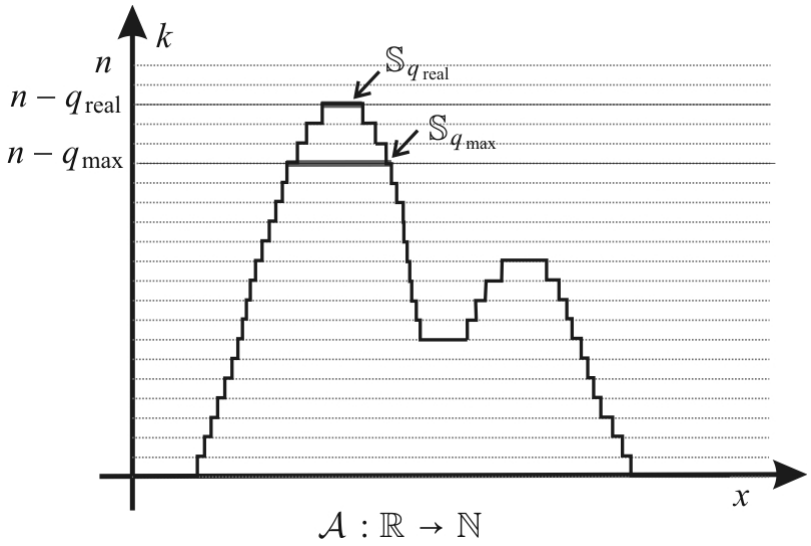

Figure 1.6 – L'utilisation d'un accumulateur pour représenter la solution d'un CSP à une dimension

La théorie des *accumulateurs* est proche de celle de la logique floue ou la transformée de Hough généralisée. Pour conclure, une représentation de solution sous forme d'un *accumulateur* nous permet d'avoir une solution d'un CSP relaxé indépendante du nombre de contraintes non satisfaites ce qui était le résultat voulu. L'implémentation des *accumulateurs* est détaillée dans le manuscrit de thèse.

1.3.2 Les sous-pavages

Tout ensemble $\mathbb{S} \subset \mathbb{R}^n$ peut être approché par une boîte ou un ensemble de boîtes qui ne se chevauchent pas ou *sous-pavage*. Plus il y a de boites dans un *sous-pavage* plus l'approximation de $\mathbb{S}$ peut être précise. Figure 1.7 montre les différentes approximations possibles avec un *sous-pavage* d'un ensemble $\mathbb{S}$ sous forme d'un disque. Dans la sous-figure (a) la boite *englobe* l'ensemble $\mathbb{S}$. Cette boite est aussi notée $[\mathbb{S}]$. Le *sous-pavage* dans la sous-figure (b) est un *sous-pavage régulier* et celui de la sous-figure (c) est appelée une *sous-pavage non régulier*. Les solveurs ensemblistes de CSPs (tels que les algorithmes SIVIA ou un des solveurs basés sur les contracteurs utilisées dans les logiciels QUIMPER [Chabert and Jaulin, 2009]) génèrent un *sous-pavages* qui représente la solution d'un CSP. Il est possible d'adapter solveurs ensemblistes afin de résoudre des CSPs relaxés en créant des contracteurs adaptés. Le paragraphe suivant présente les *contracteurs*.

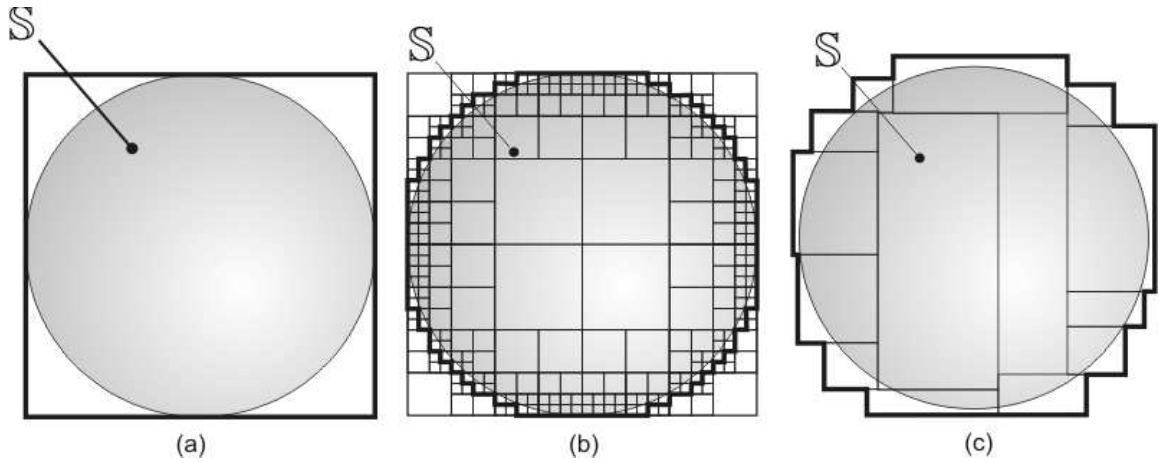

Figure 1.7 – Utilisation des sous-pavages pour approximer les ensembles

1.3.3 Les contracteurs

Si une contrainte est la définition mathématique de l'ensemble, alors le contracteur est l'algorithme qui permet de calculer une approximation de cet ensemble sous la forme d'un sous-pavage.

Exemple 5 *Considérons une contrainte* $\mathbf{x} = (x, y) \in [\mathbf{x}], ax + by + c = 0$ *où* $a, b, c \in \mathbb{R}$ *sont des constantes connues. La Figure 1.8 montre l'action d'un contracteur associé à cette contrainte. Les boîtes initiales sont représentées en gris clair et les boites résultantes des contractions sont représentés en gris foncé.*

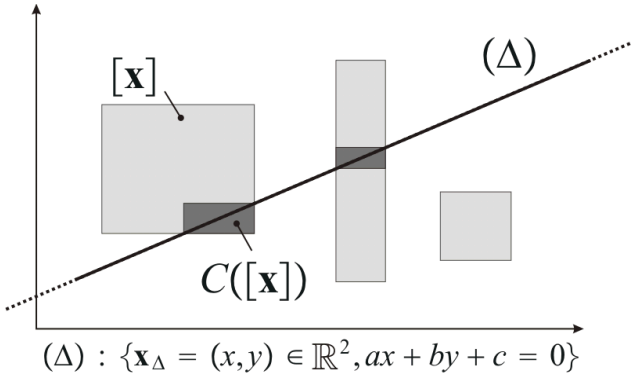

Figure 1.8 – Le contracteur associé à la contrainte "être sur la droite"

Associées à des solveurs approprié, il est possible de générer des sous-pavages représentant un ensemble particulier défini par une contrainte (voir Figure 1.9.(a),(b)) ou un ensemble de contraintes (comme on le voit dans la figure Figure 1.9.(c)).

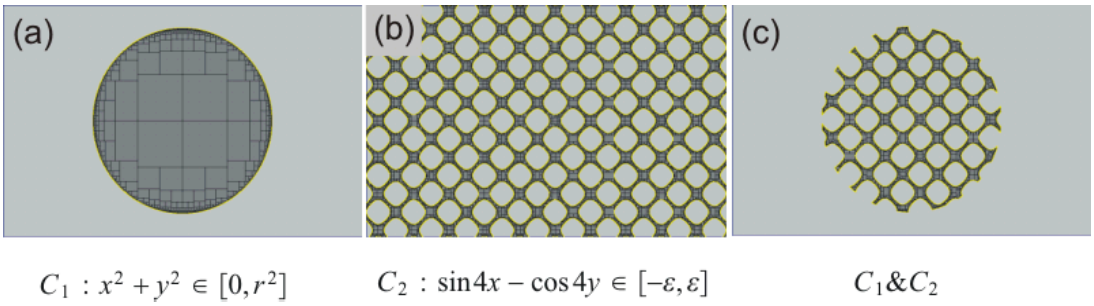

Figure 1.9 – Un exemple de sous-pavage générés par solveurs basés sur les contracteurs

1.3.4 Le contracteur sur l'image

Une majeure contribution à la thèse est la définition d'un contracteur sur l'image. Ce nouveau contracteur est le contracteur associé à l'ensemble défini par les pixels noirs (ou blanc) d'une image binaire. Le contracteur sur l'image permet de représenter des ensembles difficile à paramétrer comme les cartes dans le contexte de la localisation. Le contracteur sur l'image peut également être utilisé pour approcher des contracteurs complexes qui nécessiteraient des calculs lourds. La Figure 1.10 montre un exemple de création ainsi que d'utilisation d'un contracteur sur l'image. Dans le contexte de localisation, nous avons besoin de modéliser la carte de l'environnement. D'abords, nous avons pris une image satellitaire de Google Maps (Voir sous-figure 1.10.(a)). Ensuite, nous avons extrait une image binaire en appliquant un algorithme de traitement d'image pour la détection de contours (Sous-figure 1.10.(b)). L'idée est de calculer un transformée de cette image qui est l'intégrale de cette dernière sous la forme d'une image de niveau de gris représentée dans la figure sous-figure 1.10.(c)). Cette transformée permet de déduire trés facilement une fonction d'inclusion ainsi que le contracteur associé à l'ensemble représenté sur l'image. La sous-figure 1.10.(d) montre un sous-pavage correspondant au résultat de l'algorithme SIVIA [Jaulin and Walter, 1993] associé à la contrainte "appartenir à la carte" qui fait appel à la fonction d'inclusion sur l'image. Nous remarquons que le sous-pavage correspond bien à la carte.

1.3.5 Une application pour la localization d'un robot sous-marin

Pendant la thèse, nous avons comparé notre méthode de localisation avec la méthode de localisation utilisant le filtrage particulaire en collaboration avec l'université Heriott Watt. Les deux méthodes on été testé sur le même jeu de données prises dans une marina abandonnée par un robot sous-marin espagnol de l'université de Girone. La Figure 1.11 montre le résultat des deux méthodes. La sous-figure de gauche trace la trajectoire calculée par les méthodes ensemblistes et la trajectoire de référence. On que celle-ci sont

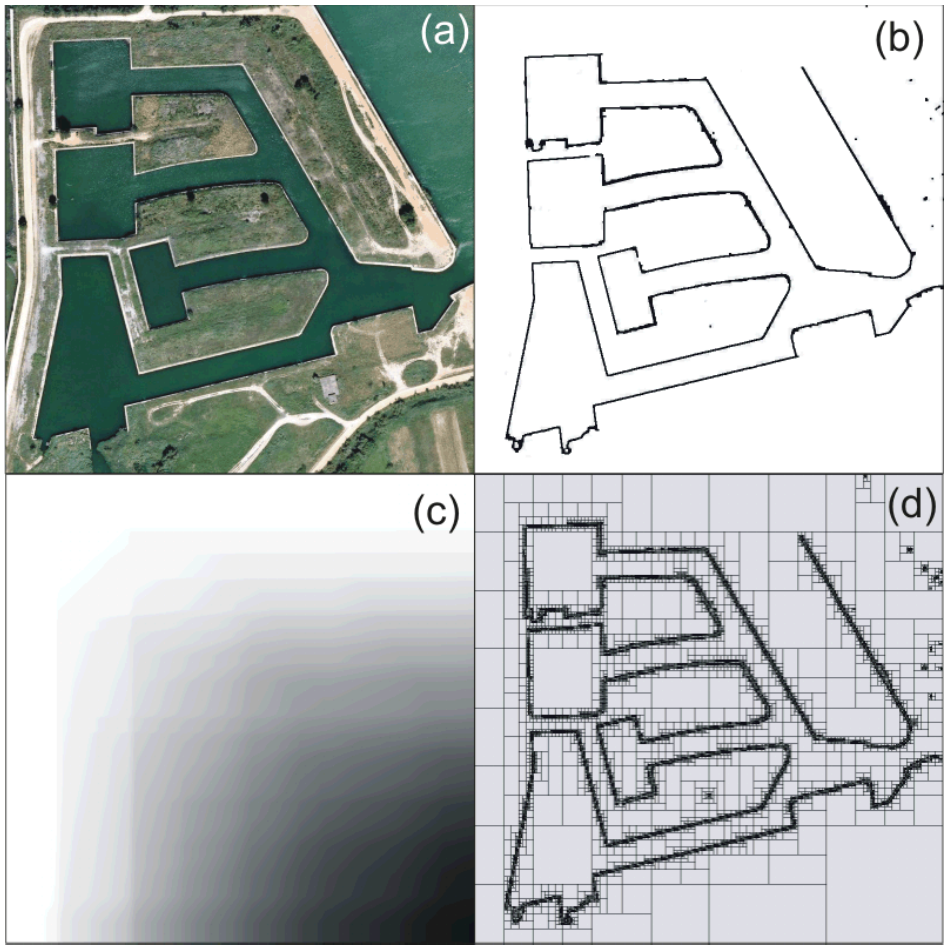

Figure 1.10 – Les différentes étapes de l'implémentation de la contrainte sur l'image

confondues. Dans, la sous-figure de droite, la trajectoire calculée par le filtre particulaire est aussi confondue avec la trajectoire GPS. Cette sous-figure montre en plus la trajectoire calculée par le "dead reckoning" (estimation aveugle) qui diverge est n'est pas exploitable. Les deux méthodes arrivent à localiser le robot et effectuent le calcul en temps réel (c'est à dire à la même allure qu'arrivent les données sur le sous-marin.

1.4 Conclusion

Dans ce résumé de thèse nous avons présenté brièvement les contributions majeures de la thèse. Ces contributions permettent d'effectuer une localisation robustes, grace aux méthodes de résolution des CSPs relaxés, dans des environnements structurés ou pas grâce au contracteur sur l'image. Ce moyen de localisation a été testé sur un jeu de données prise par un robot sous-marin dans des conditions réelles.

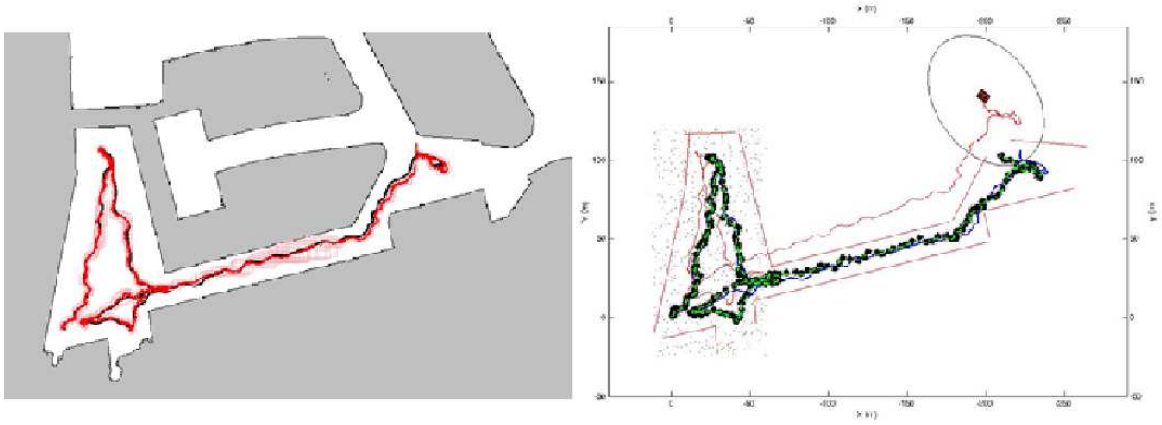

Figure 1.11 – Les résultats des différents algorithmes de localisation. Méthode ensembliste à gauche et filtrage particulaire à droite.

Publications

Journal articles

F. Le Bars, J. Sliwka, and M. Sebela. Autonomous submarine robotic system. *Cybernetic Letters*, http://www.cybletter.com, 2010.

F. Le Bars, J. Sliwka, O. Reynet, and L. Jaulin. Set-membership state estimation with fleeting data. Accepted by *Automatica*, 2011.

J. Sliwka, F. Le Bars, and L. Jaulin. Student autonomous underwater robotics at ENSIETA. submitted to *IJME (International Journal of Maritime Engineering)*, 2011.

Congres with selection committee

J. Sliwka, F. Le Bars, and L. Jaulin. Calcul ensembliste pour la localisation et la cartographie robustes. In *JD-JN-MACS 2009*, Angers, France, 2009.

F. Le Bars, J. Sliwka, and L. Jaulin. Analyse par intervalles pour le lancé de rayon et pour l'analyse de stabilité. In *JD-JN-MACS 2009*, Angers, France, 2009.

J. Sliwka, P. H. Reilhac, R. Leloup, P. Crepier, H. D. Malet, P. Sittaramane, F. L. Bars, K. Roncin, B. Aizier and L. Jaulin. Autonomous robotic boat of ENSIETA. In *IRSC-WRSC 2009*, Matosinhos , Portugal, 2009.

F. Le Bars, A. Bertholom, J. Sliwka, and L. Jaulin. Interval SLAM for underwater robots - a new experiment. In *NOLCOS 2010*, Bologna, Italy, 2010.

J. Sliwka, F. Le Bars, O. Reynet, and L. Jaulin. Using interval methods in the context of robust localization of underwater robots. In *NAFIPS 2011*, El Paso, USA, 2011.

J. Sliwka, J. Nicola, R. Coquelin, F. Becket De Megille, Benoit Clement and Luc Jaulin. Sailing without wind sensor and other hardware and software innovations. In *IRSC-WRSC 2011*, Luebeck, Germany, 2011.

K. Xiao, J. Sliwka, L. Jaulin. A wind-independent control strategy for autonomous sailboats based on voronoi diagram. In *CLAWAR 2011*, Paris, 2011.

J. Sliwka, L. Jaulin, M. Ceberio and V. Kreinovich. Processing Interval Sensor Data in the Presence of Outliers, with Potential Applications to Localizing Underwater Robots. *IEEE SMC'2011*, Anchorage, Alaska, 2011.

Congres without selection committee

J. Sliwka and L. Jaulin. Setting a Robust SLAM problem in a set of equations. In *SWIM09*, Lausanne, Switzerland, 2009.

J. Sliwka, and F. Le Bars. Robotics at ENSIETA. In *AIM'09*, Brno, Czech Republic, 2009.

J. Sliwka. Robotics activities at ENSIETA focusing on the AUV and the set membership methods. *Invited talk at Lübeck University*, Lübeck, Germany, 2009

J.Sliwka, F. Le Bars, and L. Jaulin. Image derived contractor and its application in robot localization, *SWIM 2010*, Nantes,

F. Le Bars and J. Sliwka. Calcul par intervalles et robotique à l'ENSIETA. In *Journées Jeunes Chercheurs en Robotique 2010*, Paris, France, 2010.

F. Le Bars, J. Sliwka, and L. Jaulin. SAUC'ISSE, un robot sous-marin autonome. In *Journées Démonstrateurs 2010*, Angers, France, 2010.

J. Sliwka, F. Le Bars and L. Jaulin. Using polynomials with set valued coefficients to solve relaxed constraint satisfaction problems. In *SWIM 2011*, Bourges, France, 2011.

Other documents

F. Le Bars, J. Sliwka, and L. Jaulin. Réalisation d'un robot sous-marin autonome pour le concours SAUC-E. *Compte-rendu final pour l'année 2007-2008 pour la MRIS*, 2008.

F. Le Bars, J. Sliwka, and L. Jaulin. Réalisation d'un robot sous-marin autonome. *Fourniture 1 pour la MRIS*, 2009.

J. Sliwka, F. Le Bars, and L. Jaulin. Réalisation d'un robot sous-marin autonome. *Fourniture 2 pour la MRIS*, 2009.

F. Le Bars, J. Sliwka, and L. Jaulin. Réalisation d'un robot sous-marin autonome. *Fourniture 3 pour la MRIS*, 2009.

F. Le Bars, J. Sliwka, and L. Jaulin. Réalisation d'un robot sous-marin autonome. *Fourniture 4 pour la MRIS*, 2009.

F. Le Bars, J. Sliwka, and L. Jaulin. SAUC-E 2009 Journal paper ENSIETA. In *SAUC-E 2009*, Gosport, UK, 2009.

J. Sliwka, F. Le Bars, and L. Jaulin. Construction d'un robot sous-marin pour le concours SAUC-E. *Rapport d'avancement T0+6 pour la MRIS*, 2010.

F. Le Bars, J. Sliwka, and L. Jaulin. Construction d'un robot sous-marin pour le concours SAUC-E. *Rapport d'avancement T0+9 pour la MRIS*, 2010.

F. Le Bars, J. Sliwka, and L. Jaulin. Construction d'un robot sous-marin pour le concours SAUC-E. *Rapport d'avancement T0+12 pour la MRIS*, 2010.

F. Le Bars, J. Sliwka, and L. Jaulin. SAUC-E 2010 Journal paper ENSIETA. In *SAUC-E 2010*, La Spezia, Italy, 2010.

J. Sliwka, F. Le Bars, and L. Jaulin. Construction d'un robot sous-marin pour le concours SAUC-E. *Rapport d'avancement T0+6 pour la MRIS*, 2011.

F. Le Bars, J. Sliwka, I. Torres-Tamanaja, E. Campos-Mercado, M. S. Ibn Seddik, C. Aubry, and L. Jaulin. SAUC'ISSE and SARDINE, 2 AUVs for SAUC-E 2011. In *SAUC-E 2011*, La Spezia, Italy, 2011.

Bibliography

[Baccou and Jouvencel, 2002] Baccou, P. and Jouvencel, B. (2002). Homing and navigation using one transponder for auv, post-processing comparisons results with long base-line navigation. *IEEE Int. Conf. Robotics and Automation*, pages 11–15.

[Bazeille, 2008] Bazeille, S. (2008). *Vision sous-marine monoculaire pour la reconnaissance d'objets*. PhD thesis, Université de Bretagne Occidentale.

[Benhamou and Ceberio, 2003] Benhamou, F. and Ceberio, M. (2003). Soft constraints: A unifying framework applied to continuous soft constraints. In *WS ERCIM - CoLogNET*.

[Bovik, 2000] Bovik, A. (2000). *Handbook of image and video processing*. Academic Press.

[Caiti et al., 2002] Caiti, A., Garulli, A., Livide, F., and Prattichizzo, D. (2002). Set-membership acoustic tracking of autonomous underwater vehicles. *Acta Acustica united with Acustica*, 5(88):648–652.

[Chabert and Jaulin, 2009a] Chabert, G. and Jaulin, L. (2009a). Contractor Programming. *Artificial Intelligence*, 173:1079–1100.

[Chabert and Jaulin, 2009b] Chabert, G. and Jaulin, L. (2009b). Hull consistency under monotonicity. In *CP'2009*.

[Cleary, 1987] Cleary, J. G. (1987). Logical arithmetic. *Future Computing Systems*, 2(2):125–149.

[Clowes, 1971] Clowes, M. B. (1971). On seeing things. *Artificial Intelligence*, 2:179–185.

[Clérentin et al., 2008] Clérentin, A., Delafosse, M., Delahoche, L., Marhic, B., and Jolly-Desodt, A. (2008). Uncertainty and imprecision modeling for the mobile robot localization problem. *Autonomous Robots*, 24(3):1573–7527.

[Combastel, 2005] Combastel, C. (2005). A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. In *CDC-ECC '05*.

[Davey and Priestley, 2002] Davey, B. A. and Priestley, H. A. (2002). *Introduction to Lattices and Order*. Cambridge University Press, (ISBN 0521784514).

[Davis, 1987] Davis, E. (1987). Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331.

[Drevelle and Bonnifait, 2009] Drevelle, V. and Bonnifait, P. (2009). High integrity gnss location zone characterization using interval analysis. In *ION GNSS*.

[Durieu et al., 1996] Durieu, C., Polyak, B., and Walter, E. (1996). Ellipsoidal state outer-bounding for MIMO systems via analytical techniques. In *Proceedings of the IMACS—IEEE–SMC CESA'96 Symposium on Modelling and Simulation*, volume 2, pages 843–848, Lille, France.

[Fantoni and Lozano,] Fantoni, I. and Lozano, R. *Non-linear control for underactuated mechanical systems*. Springer.

[Ferson et al., 2003] Ferson, S., Kreinovich, V., Ginzburg, L., Myers, D. S., and Sentz, K. (2003). Constructing probability boxes and dempster-shafer structures. *SAND REPORT*.

[Gning, 2006] Gning, A. (2006). *Localisation garantie d'automobiles. Contribution aux techniques de satisfaction de contraintes sur les intervalles*. PhD dissertation, Université de Technologie de Compiègne, Compiègne, France.

[Gordon et al., 1993] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113.

[Halbwachs and Meizel, 1996] Halbwachs, E. and Meizel, D. (1996). Bounded-error estimation for mobile vehicule localization. *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, pages 1005–1010.

[Hardouin, 2004] Hardouin, L. (2004). Sur la commande linéaire de systèmes à événements discrets dans l'algèbre (max,+). *HDR*.

[Hyvönen, 1992] Hyvönen, E. (1992). Constraint reasoning based on interval arithmetic: The tolerance propagation approach. *Artificial Intelligence*, 58(1-3):71–112.

[Jaulin, 2009] Jaulin, L. (2009). Robust set membership state estimation ; application to underwater robotics. *Automatica*, 45(1):202–206.

[Jaulin, 2011] Jaulin, L. (2011). Solving set-valued constraint satisfaction problems. *Computing*.

[Jaulin and Bazeille, 2009] Jaulin, L. and Bazeille, S. (2009). Image shape extraction using interval methods. In *Sysid 2009*.

[Jaulin et al., 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London.

[Jaulin and Walter, 1993] Jaulin, L. and Walter, E. (1993). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064.

[Jaulin and Walter, 2002] Jaulin, L. and Walter, E. (2002). Guaranteed robust nonlinear minimax estimation. *IEEE Transaction on Automatic Control*, 47(11):1857–1864.

[Jaulin et al., 1996] Jaulin, L., Walter, E., and Didrit, O. (1996). Guaranteed robust nonlinear parameter bounding. In *Proceedings of CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, pages 1156–1161, Lille, France.

[Joly, 2010] Joly, C. (2010). *Contributions aux méthodes de localisation et cartographie simultanées par vision omnidirectionnelle*. PhD dissertation, Ecole Nationale Supérieure des Mines de Paris, Paris, France.

[Kearfott and Kreinovich, 1996] Kearfott, R. B. and Kreinovich, V., editors (1996). *Applications of Interval Computations*. Kluwer, Dordrecht, the Netherlands.

[Kieffer, 1999] Kieffer, M. (1999). *Estimation ensembliste par analyse par intervalles, application à la localisation d'un véhicule*. PhD dissertation, Université Paris-Sud, Orsay, France.

[Klir and Yuan, 1995] Klir, G. J. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall PTR, NJ, USA, 1st edition.

[Le Bars et al., 2010] Le Bars, F., Bertholom, A., Sliwka, J., and Jaulin, L. (2010). Interval slam for underwater robots; a new experiment. In *NOLCOS 2010*, Italy.

[Le Bars et al., 2011] Le Bars, F., Sliwka, J., Reynet, O., and Jaulin, L. (2011). State estimation with fleeting data. *Automatica*.

[Maurelli et al., 2008] Maurelli, F., Krupinski, S., Petillot, Y., and Salvi, J. (2008). A particle filter approach for auv localization. In *OCEANS 2008*, pages 1–7.

[Meizel et al., 2002] Meizel, D., Lévêque, O., Jaulin, L., and Walter, E. (2002). Initial localization by set inversion. *IEEE transactions on robotics and Automation*, 18(6):966–971.

[Meizel et al., 1996] Meizel, D., Preciado-Ruiz, A., and Halbwachs, E. (1996). Estimation of mobile robot localization: geometric approaches. In Milanese, M., Norton, J., Piet-Lahanier, H., and Walter, E., editors, *Bounding Approaches to System Identification*, pages 463–489. Plenum Press, New York, NY.

[Moore, 1959] Moore, R. E. (1959). Automatic error analysis in digital computation. Technical Report LMSD-48421 Lockheed Missiles and Space Co, Palo Alto, CA.

[Nguyen and Walker, 2005] Nguyen, H. T. and Walker, E. A. (2005). *A First Course in Fuzzy Logic, Third Edition*. Chapman & Hall/CRC.

[Normand et al., 2010] Normand, J., Goldsztejn, A., Christie, M., and Benhamou, F. (2010). A branch and bound algorithm for numerical max-csp. *Constraints*.

[Porta, 2005] Porta, J. (2005). Cuikslam: A kinematics-based approach to slam. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2436–2442, Barcelona (Spain).

[Ramdani, 2005] Ramdani, N. (2005). Méthodes ensemblistes pour l'estimation. *HDR*.

[Ribas et al., 2008] Ribas, D., Ridao, P., Tardós, J., and Neira, J. (2008). Underwater SLAM in man made structured environments. *Journal of Field Robotics*, Accepted for publication.

[Russ, 2002] Russ, J. C. (2002). *Image processing handbook (The)*. CRC Press.

[Sam-Haroud, 1995] Sam-Haroud, D. (1995). *Constraint consistency techniques for continuous domains*. PhD dissertation 1423, Swiss Federal Institute of Technology in Lausanne, Switzerland.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *CVPR 94*.

[Sliwka et al., 2011a] Sliwka, J., Jaulin, L., Ceberio, M., and Kreinovich, V. (2011a). Processing interval sensor data in the presence of outliers, with potential applications to localizing underwater robots. In *IEEE SMC'2011*, Anchorage, Alaska.

[Sliwka et al., 2011b] Sliwka, J., Nicola, J., Coquelin, R., Megille, F. B. D., Clement, B., and Jaulin, L. (2011b). Sailing without wind sensor and other hardware and software innovations. In *IRSC-WRSC 2011*, Luebeck, Germany.

[Sliwka et al., 2009] Sliwka, J., Reilhac, P. H., Leloup, R., Crepier, P., Malet, H. D., Sittaramane, P., Le Bars, F., Roncin, K., Aizier, B., and Jaulin, L. (2009). Autonomous robotic boat of ensieta. In *IRSC-WRSC 2009*, Matosinhos (Portugal).

[Tarski, 1955] Tarski, A. (1955). *SYNCHRONIZATION AND LINEARITY: An Algebra for Discrete Event Systems*, volume 8. Pacific Journal of Mathematics.

[Thrun et al., 2005] Thrun, S., Bugard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press, Cambridge, M.A.

[Veres et al., 2008] Veres, S., Tsourdos, A., and Fisher, M. (2008). Low cost disposable autonomous vehicles. *Defence Management Journal*, 42:64–65.

[Waltz, 1975] Waltz, D. (1975). Generating semantic descriptions from drawings of scenes with shadows. In Winston, P. H., editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, NY.

[Warmus, 1956] Warmus, M. (1956). Calculus of approximations. *Bull. Acad. Polon. Sci., Cl. III*.

[Xiao et al., 2011] Xiao, K., Sliwka, J., and Jaulin, L. (2011). A wind-independent control strategy for autonomous sailboats based on voronoi diagram. In *CLAWAR 2011*, Paris.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information Control*, 5:338–353.

Title: Using set membership methods for robust underwater robot localization
Keywords: Robust localization, relaxed CSP, outliers, set polynomials, accumulators, image contractor
Abstract: For an intelligent robot to be able to properly interact with its environment, it has to know in one hand the environment and in the other hand its state in that environment. In particular, a robot must know where it is to know where it has to go. Since the appearance of GPS, the problem of localization has been practically solved on the ground. GPS doesn't work underwater since high frequency electromagnetic waves don't propagate in that environment. However, the number of undersea operations increases significantly every year. In our school, we develop an autonomous underwater vehicle to test the underwater localization systems. The main sensor we use is an imaging sonar. An imaging sonar is an acoustic sensor which detects acoustically reflective objects. For example, the sonar can be used to detect the walls of a port. The measurements from the sonar are often corrupted with outliers. An outlier may be due to an electrical failure of the sensor or a phenomenon not taken into account when modeling the environment. The number of outliers is often unknown and varies with time. The aim of this thesis was to solve the localization problem using such data. The localization problem can be formulated as a constraint satisfaction problem (CSP). A CSP is basically a system of equations (constraints). Here, the unknown is the pose of the robot. For each measurement we obtain a constraint involving the pose, a measurement and the environment (the map). The classical solution of a CSP is the set of points (poses) that satisfy all constraints. However, because of outliers, such points may not exist. The new problem is to find a solution to a CSP when only part of constraints is satisfied. We call this problem a relaxed CSP. A major contribution to the thesis was to find several representations of the solution of a relaxed CSP as well as algorithms to compute these solutions. The first representation is in the form of a polynomial with set valued coefficients also called a set polynomial. Each coefficient is the set of points that satisfy the number of constraints equal to the degree of the coefficient in the polynomial. Such representation allows the use of polynomial arithmetic to calculate the solution polynomial. A second representation is in the form of a function, called accumulator, which for each element of the search space returns the number of constraints it satisfies. One of the hurdles to overcome to solve localization problems is the representation of the map. In case of structured environments, it is possible to represent the map by a set of parameterized objects such as segments, polygons, curves. In case of unstructured maps such as seashore or lake borders, the idea is to represent the map (which actually is a set) in the form of a binary image where pixels of interest (black for example) represent the set of points of the map. Another major contribution to the thesis was to be able to use the binary image representation in CSP or relaxed CSP computer solvers in the form of a contractor called the image contractor. The usefulness of those two contributions is illustrated on a real case example of localization of an underwater robot in an abandoned marina. The thesis contains many other contributions to set membership methods and the contractor theory.

Titre: Méthodes ensemblistes pour une localisation robuste de robots sous-marins
Mots Clés: Localisation robuste, CSP relaxé, données aberrantes, polynômes ensemblistes, accumulateurs, contracteur sur l'image
Résumé: Pour qu'un robot autonome puisse interagir proprement avec son milieu, ce dernier doit connaitre d'une part l'environnement dans lequel il évolue et d'autre part son état dans cet environnement. En particulier, un robot doit savoir où il est pour savoir où il doit aller. Depuis l'apparition du GPS, le problème de la localisation a été pratiquement résolu pour les robots terrestres. Le GPS ne fonctionne pas sous l'eau. Toutefois, le nombre d'opérations sous-marines augmente de manière significative chaque année. Dans notre école, nous développons un robot sous-marin pour tester des systèmes de localisation sous-marins. Le capteur principal que nous utilisons est un sonar sectoriel. Un sonar est un capteur acoustique qui positionne les objets acoustiquement réfléchissant. Par exemple, le sonar peut être utilisé pour détecter les parois d'un port. Ce capteur donne souvent des mesures aberrantes. Une telle mesure peut être due à une défaillance électrique du capteur ou d'un phénomène non pris en compte lors de la modélisation de l'environnement. Le nombre de mesures aberrantes est souvent inconnu et varie avec le temps. Le but de la thèse est de résoudre le problème de localisation avec de telles données. Un problème de localisation peut être formulé en tant que problème de satisfaction de contraintes (CSP en anglais). Un CSP est en gros un système d'équations (contraintes). Ici, l'inconnu est la pose du robot. Pour chaque mesure on obtient une contrainte reliant la pose, la mesure et l'environnement. La solution classique d'un CSP est l'ensemble des points (poses) qui satisfont toutes les contraintes. Toutefois, a cause des données aberrantes de tels points peuvent ne pas exister. Le nouveau problème consiste à trouver une solution d'un CSP lorsque une partie seulement de contraintes est satisfaite. Nous appelons ce problème un CSP relaxé. Une des contributions majeures à la thèse était de trouver plusieurs représentations de la solution d'un CSP relaxé ainsi que les algorithmes qui permettent de calculer ces solutions. La première représentation est sous la forme d'un polynôme dont les coefficients sont des ensembles que nous appelons polynômes ensemblistes. Chaque coefficient correspond à l'ensemble des points qui satisfont le nombre de contraintes égal au degré du coefficient dans le polynôme. Une telle représentation permet d'utiliser l'arithmétique des polynômes pour calculer le polynôme solution. Une deuxième représentation est sous la forme d'une fonction, qu'on appelle accumulateur, qui pour chaque élément de l'espace de recherche retourne le nombre de contraintes satisfaites. Un des obstacles à surmonter pour résoudre les problèmes de localisation est la représentation de la carte. En cas d'environnements structurés, il est possible de représenter la carte par un ensemble d'objets paramétrés tels que des segments, polygones ou des courbes. En cas d'environnements non structurées où en partie structurées tels que des cartes marines ou des cartes routières, l'idée est de représenter la carte (qui est en fait un ensemble de points) sous la forme d'une image binaire où les pixels d'intérêt (noir par exemple) représentent l'ensemble des points de la carte. Une des contributions majeures de la thèse était d'incorporer une telle représentation de la carte dans le formalisme d'un CSP ou d'un CSP relaxé sous la forme d'un contracteur appelé le contracteur sur l'image. L'utilité de ces deux contributions est montrée par un exemple de localisation d'un vrai robot dans une marina abandonnée. La thèse contient plusieurs autres contributions aux méthodes ensemblistes et la théorie des contracteurs.