THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Informatique

École doctorale I2S (Information, Structures, Systèmes)

Unité de recherche COCONUT

Méthodes ensemblistes pour le problème de recalage dans un contexte de robotique d'exploration

Présentée par Verlein RADWAN
Le 11 Décembre 2025

Sous la direction de Gilles TROMBETTONI et Simon ROHOU

Devant le jury composé de

David DANEY, DR, INRIA de l'Université de Bordeaux	Rapporteur
Luc JAULIN, PR, Lab-STICC, Université de Bretagne Occidentale, ENSTA	Rapporteur
Christophe FIORIO, PR, LIRMM, Université de Montpellier	Président du jury
Alexandre GOLDSZTEJN, CR, LS2N, CNRS	Examinateur
Carine JAUBERTHIE, PR, LAAS-CNRS, Université de Toulouse	Examinateur
Simon ROHOU, MCF, Lab-STICC, Université de Bretagne Occidentale, ENSTA	co-Encadrant de thèse
Gilles TROMBETTONI, PR, LIRMM, Université de Montpellier	Directeur de thèse

UNIVERSITÉ DE MONTPELLIER

Résumé

Le problème qui nous intéresse dans cette thèse est celui du recalage d'ensembles en dimension 2 : soient $\mathbb{A}$ et $\mathbb{B}$ deux sous-ensembles de $\mathbb{R}^2$, trouver la transformation qui permet de recaler $\mathbb{A}$ et $\mathbb{B}$. Les outils mathématiques proposés, basés sur l'arithmétique d'intervalles, permettent une résolution ensembliste offrant des propriétés de garantie et d'exhaustivité.

Une première contribution répond à un problème de robotique d'exploration, une variante difficile du problème de localisation et cartographie simultanées (SLAM) dans un environnement non structuré. Un robot se déplace dans un environnement d'obstacles et effectue n observations partielles d'un ensemble à cartographier grâce à des capteurs distance et angle, et cherche à s'y localiser en cherchant à approximer finement la transformation rigide (translation et rotation) cohérente avec les observations et la carte. D'abord, nous montrons comment décomposer ce problème de recalage en une conjonction de sous-problèmes que nous pouvons traiter. Le problème manipule des variables ensemblistes dont les valeurs sont des sous-ensembles de $\mathbb{R}^2$, bornées par un intervalle d'ensembles. Un ensemble épais, ou intervalle d'ensembles, possède une borne inférieure incluse dans tous les ensembles de l'intervalle et une borne supérieure qui en est un sur-ensemble. Un ensemble épais est représenté par un pavage épais, c'est-à-dire un ensemble de boîtes (vecteurs d'intervalles, ici des rectangles) pavant l'espace. Nous proposons d'étiqueter les boîtes du pavage avec trois intervalles de booléens. Avec ce cadre plus complet que les approches de la littérature, nous avons pu développer une algèbre de ces étiquettes particulières afin d'étendre les opérations ensemblistes classiques aux pavages épais réguliers et de garantir ces opérations. Un algorithme de pavage dédié, basée sur cette approche, permet de résoudre ce problème de SLAM.

Une seconde contribution s'intéresse au cas du recalage d'ensembles 2D sous transformation de similarité. Soit deux ensembles $\mathbb{A}$ et $\mathbb{B}$, le problème consiste à retrouver la transformation de similarité (translation, rotation et changement d'échelle) qui permet de recaler $\mathbb{A}$ sur $\mathbb{B}$. Ce problème s'applique par exemple à l'alignement de prises de vues aériennes d'un environnement plat à des positions, orientations et altitudes différentes. Nous proposons une version intervalle d'une approche de Procuste qui consiste à recentrer et mettre à l'échelle les ensembles avec leur cercle minimal, avant de résoudre le problème d'orientation résultant plus simple à traiter. Un arsenal intervalle sophistiqué met en œuvre ce principe de Procuste en utilisant des opérateurs intervalles appelés séparateurs, une généralisation des contracteurs permettant de caractériser l'intérieur et l'extérieur de l'espace-solution d'une contrainte. Cette approche calcule l'ensemble des solutions possibles (en cas de symétries) tout en prenant en compte les incertitudes.

Une implémentation dans la bibliothèque CODAC permet de valider les contributions proposées sur des problèmes simulés difficiles.

Abstract

The thesis handles the 2-dimensional set registration problem : let $\mathbb{A}$ and $\mathbb{B}$ be two subsets of $\mathbb{R}^2$, find the transformation that allows $\mathbb{A}$ and $\mathbb{B}$ to be registered. The mathematical tools proposed, based on interval arithmetic, enable a set-theoretic solution offering properties of guarantee and completeness.

A first contribution addresses a problem in exploration robotics, a difficult variant of the simultaneous localization and mapping (SLAM) problem in an unstructured environment. A robot moves through an environment containing obstacles and makes n partial observations of a set to be mapped using range and bearing sensors. It then attempts to locate itself within the environment by seeking to accurately approximate the rigid transformation (translation and rotation) consistent with the observations and the map. First, we show how to decompose this registration problem into a conjunction of subproblems that we can handle. The problem manipulates set variables whose values are subsets of $\mathbb{R}^2$, bounded by an interval of sets. A thick set, or set interval, has a lower bound included in all sets of the interval and an upper bound that is a superset of it. A thick set is represented by a thick paving, i.e. a set of boxes (interval vectors, here rectangles) paving the space. We propose to label the boxes of the paving with three Boolean intervals. With this framework more comprehensive than the approaches in the literature, we could develop an algebra of these particular labels in order to extend classical set operations to regular thick pavings and to guarantee these operations. A dedicated paving algorithm, based on this approach, allows us to solve this SLAM problem.

A second contribution focuses on the 2D set registration problem under similarity transformation. Given two sets $\mathbb{A}$ and $\mathbb{B}$, the problem consists in finding the similarity transformation (translation, rotation, and scaling) that allows $\mathbb{A}$ to be registered onto $\mathbb{B}$. This problem applies, for example, to the alignment of aerial views of a flat environment at different positions, orientations, and altitudes. We propose an interval version of a Procrustes approach that consists of re-centering and scaling the sets with their minimum circle, before solving the resulting orientation problem that is easier to handle. A sophisticated interval arsenal implements this Procrustes principle using interval operators called separators, a generalization of contractors that characterize the inner and outer regions of the solution space of a constraint. This approach calculates the set of possible solutions (in the case of symmetries) while taking uncertainties into account.

An implementation in the CODAC library allows the proposed contributions to be validated on difficult simulated problems.

Table des matières

Chapitre 1

Introduction

La robotique est un domaine vaste, avec des applications variées telles que des missions d'exploration, de cartographie allant même jusqu'aux véhicules autonomes.

Dans ce contexte particulier, la capacité d'un robot à percevoir, modéliser et se localiser dans son environnement est un problème classique de robotique, celui de Localisation et Cartographie Simultanées (*Simultaneous Localisation And Mapping* ou **SLAM**).

Une manière de s'attaquer au problème de SLAM est d'utiliser le recalage comme une sous-routine. On estime la *pose* du robot, c'est-à-dire sa position et orientation, à chaque instant en comparant ses observations de l'environnement entre elles, ou bien avec une carte partiellement construite. Le recalage consiste alors, étant donné deux ensembles, à déterminer la transformation qui les aligne au mieux.

Nous verrons dans le chapitre 3, où nous faisons un état de l'art du SLAM, qu'il peut être abordé sous plusieurs formalismes, notamment probabiliste et ensembliste. Dans sa forme classique probabiliste, sa résolution repose sur des méthodes d'optimisation, visant à maximiser la ressemblance entre observations. Cependant, ces méthodes sont susceptibles d'aboutir à des non-solutions dues aux minima locaux, phénomène particulièrement fréquent dans les environnements symétriques ou redondants, dans lesquels différentes localisations produisent des observations identiques. De tels environnements ne sont pas rares en réalité : différentes pièces d'un même immeuble de bureaux peuvent être virtuellement identiques, pièces dans lesquelles des robots sont amenés à circuler.

À l'inverse, une approche ensembliste cherche à raisonner de manière conservative sur des ensembles d'état admissibles. Elle consiste alors à ne rejeter que des non-solutions, et donc garantir que les solutions conservées incluent la pose et l'environnement réels.

Dans ce manuscrit, nous mettons en place des outils ensemblistes permettant de répondre à différents problèmes exprimés dans un formalisme ensembliste tels que celui du recalage, et de pouvoir aller jusqu'à une implémentation d'un SLAM ensembliste.

1.1 Problème du SLAM en robotique

Dans la littérature, plusieurs propositions permettent de résoudre le SLAM de manière probabiliste, c'est-à-dire en trouvant la solution la plus probable selon les données à disposition. Les propositions plus récentes s'emploient à coller de plus en plus à la réalité avec des environnements dynamiques plutôt que statiques, ou encore à y intégrer d'autres technologies de capteurs et les nouvelles avancées dans les méthodes et outils de traitement.

1.2 Problème du recalage pour le SLAM

L'un des sous-problèmes du SLAM est celui de l'association de données : il s'agit de faire correspondre des données perçues depuis des endroits ou instants différents. En effet, le robot se déplace selon une transformation rigide – c'est-à-dire une translation et une rotation. Ainsi de son point de vue, l'environnement effectue la transformation inverse. Par un alignement entre les mesures, il est possible de retrouver cette transformation, et donc le déplacement du robot. Ce problème particulier est classique dans de nombreux domaines et possède plusieurs dénominations, à savoir problème de recalage dans le domaine d'image (*registration* en anglais) [1], ou encore problème de Procuste [2] en analyse factorielle.

Le problème de recalage qui nous intéresse dans cette thèse est celui du recalage d'ensembles en dimension n : soient $\mathbb{A}$ et $\mathbb{B}$ deux sous-ensembles de $\mathbb{R}^n$. Nous cherchons à trouver une fonction bijective $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^n$ telle que :

$$\mathbf{h}(\mathbb{A}) = \mathbb{B}. \tag{1.1}$$

Pour nos applications, cette fonction représente un changement de repère. Plus généralement, elle pourra être une rotation, une réflexion, une translation, un changement d'échelle, ou bien une composition de plusieurs de celles-ci. Ainsi, cette fonction sera notée $\mathbf{h}(\mathbf{p}, \cdot)$ ou encore $\mathbf{h}_\mathbf{p}(\cdot)$, où $\mathbf{p} \in \mathbb{R}^d$ représente les d paramètres de la fonction permettant de complètement la caractériser, à savoir les vecteurs de translation, angles de rotations, facteurs de changement d'échelle, et s'il y a ou non une réflexion.

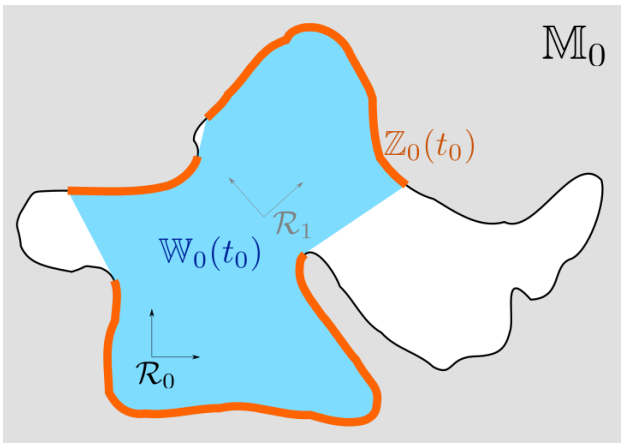
1.3 Problèmes de recalage considérés dans cette thèse

Pour les deux contributions principales de cette thèse, nous allons considérer deux transformations différentes dans un espace en dimension 2, avec des objets manipulés de différentes natures.

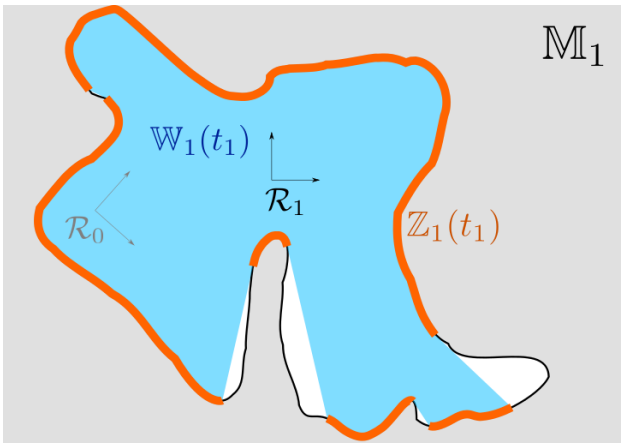
1.3.1 Recalage ensembliste sur ensembles partiels

Pour la première contribution de cette thèse (cf. partie II), nous allons considérer le scénario illustré sur la figure 1.1 : un robot se déplace dans un environnement $\mathbb{M} \in \mathcal{P}(\mathbb{R}^2)$ d'obstacles et effectue n observations. Le sous-ensemble $\mathbb{M}$ est une zone non libre, tandis que $\overline{\mathbb{M}}$ est une zone où peut évoluer le robot et les signaux capteurs. Les mesures permettent de déduire une zone sans obstacles $\mathbb{W}_i(t_i)$, ainsi qu'une zone d'impact $\mathbb{Z}_i(t_i)$ des obstacles visibles depuis le robot. Ces deux ensembles sont définis plus formellement dans la suite. Le déplacement du robot entre deux instants t_i, t_j équivaut à une

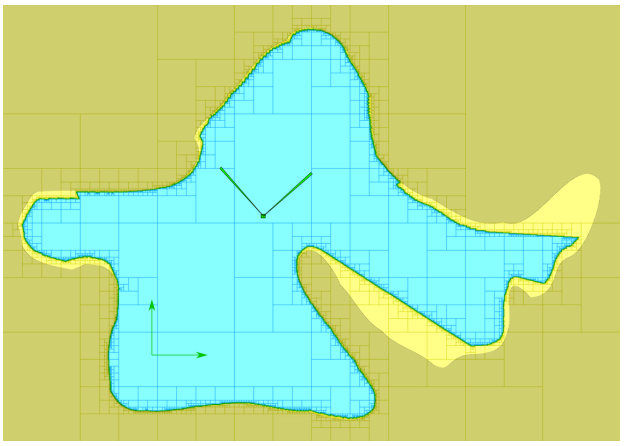
transformation rigide dans un environnement à deux dimensions, soit une translation de dimension 2 et une rotation de dimension 1, nécessitant donc l'identification de trois paramètres. À chaque instant t_i, la pose du robot définit un repère local noté $\mathcal{R}_i$. Sans perte de généralité, on définira $\mathcal{R}_0$ comme étant le repère global, et en particulier $\mathbf{x}(t_0) = \mathbf{0}$ comme dans tout problème de SLAM.



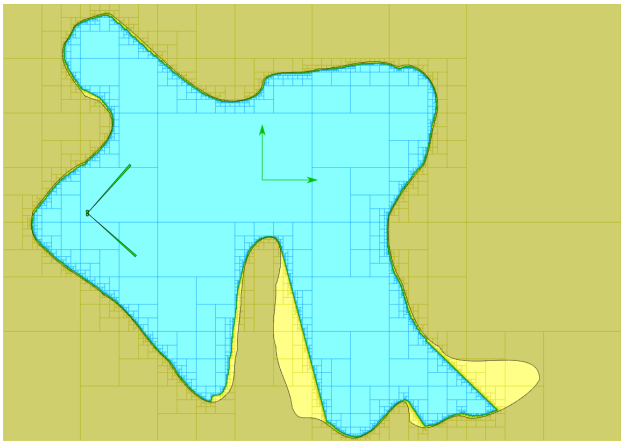
(a) Représentation des ensembles $\mathbb{W}_0(t_0)$ et $\mathbb{Z}_0(t_0)$, obtenus par la fonction d'observation.



(b) Représentation des ensembles $\mathbb{W}_1(t_1)$ et $\mathbb{Z}_1(t_1)$, obtenus par la fonction d'observation.



(c) Résultat de notre approche pour les ensembles exprimés dans $\mathcal{R}_0$, et estimation du repère $\mathcal{R}_1$.



(d) Résultat de notre approche pour les ensembles exprimés dans $\mathcal{R}_1$, et estimation du repère $\mathcal{R}_0$.

FIGURE 1.1 – À l'instant t_0 (resp. t_1), un robot effectue un ensemble de mesures avec des capteurs *range and bearing* (distance et angle). La visualisation des résultats obtenus par les méthodes ensemblistes développées dans ce manuscrit apparaît en bas : en bleu est représenté l'ensemble $\mathbb{W}_0(t_0) \cup \mathbb{W}_0(t_1)$ (resp. $\mathbb{W}_1(t_0) \cup \mathbb{W}_1(t_1)$), garanti d'être sans obstacle, en vert est représentée l'enveloppe des estimations du repère $\mathcal{R}_1$ (resp. $\mathcal{R}_0$), obtenue par l'estimation de l'état $\mathbf{x}(t_1)$ (resp. $\mathbf{h}^{-1}(\mathbf{x}(t_1), \mathbf{0})$, puisque exprimé dans le repère $\mathcal{R}_1$ ici).

Remarque 1. *Dans la suite, lorsqu'un ensemble ou un vecteur est exprimé dans un repère donné, nous l'indiquons par un indice correspondant. Ainsi, $\mathbb{M}_i$ désignera l'ensemble $\mathbb{M}$ projeté dans le repère $\mathcal{R}_i$.*

Remarque 2. *Ici, les changements de repères peuvent être déduits à partir de l'état du robot : entre deux instants t_i et t_j, le robot s'est déplacé – en position et orientation – de $\mathbf{x}(t_j) - \mathbf{x}(t_i)$. Il est plus*

adapté d'exprimer ce déplacement dans l'un des repères concernés, $\mathcal{R}_i$ ou $\mathcal{R}_j$, dans lesquels nous avons :

$$\mathbf{x}_{\mathcal{R}_i}(t_j) - \mathbf{x}_{\mathcal{R}_i}(t_i) = \mathbf{x}_{\mathcal{R}_i}(t_j) \tag{1.2}$$
$$\mathbf{x}_{\mathcal{R}_j}(t_j) - \mathbf{x}_{\mathcal{R}_j}(t_i) = -\mathbf{x}_{\mathcal{R}_j}(t_i) \tag{1.3}$$

Du point de vue du robot, l'environnement a suivi la transformation inverse. Nous en déduisons donc :

$$\mathbf{h}_{\mathcal{R}_i \to \mathcal{R}_j}(\cdot) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \cdot). \tag{1.4}$$

Pour percevoir son environnement, celui-ci dispose d'un ou plusieurs capteurs « distance et angle » (*range and bearing*), mesurant des distances à différents angles, comme un LiDAR (pour *Light Detection And Ranging*) qui utilise des lasers, ou un sonar avec des ondes acoustiques. Par cette technologie de capteur, il est possible d'obtenir seulement des couvertures partielles des ensembles puisque le LiDAR ou sonar ne perçoivent que le premier obstacle rencontré. Nous utilisons les notations suivantes pour la définition du problème, aussi illustrées dans la figure 1.1 :

- $\mathbf{g}(\cdot)$ est une fonction d'observation qui modélise la donnée du capteur en fonction de l'état et de l'environnement,
- l'ensemble $\mathbb{Z}_i(t_i)$ est l'ensemble des obstacles présents sur la frontière $\partial\mathbb{M}_i$ de l'environnement associés à une mesure capteur à l'instant t_i, $i \in \{0, \dots, m-1\}$,
- l'ensemble $\mathbb{W}_i(t_i)$ est un ensemble vide de tout obstacle situé entre la position du robot $\mathbf{x}(t_i)$ et les points de mesures $\mathbb{Z}_i(t_i)$.

Pour comparer ces ensembles avec la connaissance de l'environnement, il est nécessaire de les exprimer dans le même repère, choisi comme étant $\mathcal{R}_0$ ici, via un changement de repère.

Il est alors possible d'en tirer plusieurs ensembles de relations.

- Définitions et propriétés des ensembles :

$$\forall i \in \{0, \dots, m-1\}, \quad \begin{cases} \mathbb{W}_i(t_i), \mathbb{Z}_i(t_i) = \mathbf{g}(\mathbf{x}(t_i), \mathbb{M}_0) \\ \mathbb{W}_i(t_i) \subset \overline{\mathbb{M}_i} \\ \mathbb{Z}_i(t_i) \subset \partial\mathbb{M}_i \end{cases} \tag{1.5}$$

- Recalage de chaque paire d'ensembles :

$$\forall (i,j) \in \{0, \dots, m-1\}^2, \quad \begin{cases} \mathbb{W}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{W}_i(t_i)) \\ \mathbb{Z}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{Z}_i(t_i)) \end{cases} \tag{1.6}$$

Une autre manière d'envisager ce problème est de considérer l'environnement comme la donnée de trois sous-ensembles, formant une tripartition de l'espace : l'intérieur, noté habituellement $\mathring{\mathbb{M}}$, qui ne contient que des obstacles, l'extérieur $\overline{\mathbb{M}}$ représentant la zone libre, et la frontière $\partial\mathbb{M}$ représentant l'interface entre les deux, c'est-à-dire les seuls points de l'espace susceptibles de déclencher des mesures LiDAR ou sonar. La difficulté dans la première contribution par rapport au problème de recalage classique réside dans l'information incomplète des ensembles comme montré par (1.5), et en particulier qui ne concernent que l'extérieur et la frontière. Cela demandera l'utilisation de méthodes ensemblistes adaptées à cette modélisation.

Les inconnues de ce problème sont alors les poses $\mathbf{x}(t_i)$ et les environnements $\mathbb{M}_j$, qui seront approchés par l'extérieur et la frontière :

$$\bigcup_{0\leq i\leq m-1} \mathbb{W}_j(t_i) \subset \overline{\mathbb{M}_j}, \tag{1.7}$$

$$\bigcup_{0\leq i\leq m-1} \mathbb{Z}_j(t_i) \subset \partial\mathbb{M}_j. \tag{1.8}$$

Application au SLAM

Pour son application au SLAM, il n'est pas nécessaire de mettre en correspondance toutes les paires d'instants. Dans la première contribution de ce manuscrit, nous allons uniquement considérer les ensembles exprimés dans le repère $\mathcal{R}_0$, c'est-à-dire les cas où $j = 0$, des équations (1.6) à (1.8). La figure 1.2 montre alors un exemple de ce qu'il est possible d'obtenir sur ce même exemple avec $m = 4$ instants d'observation.

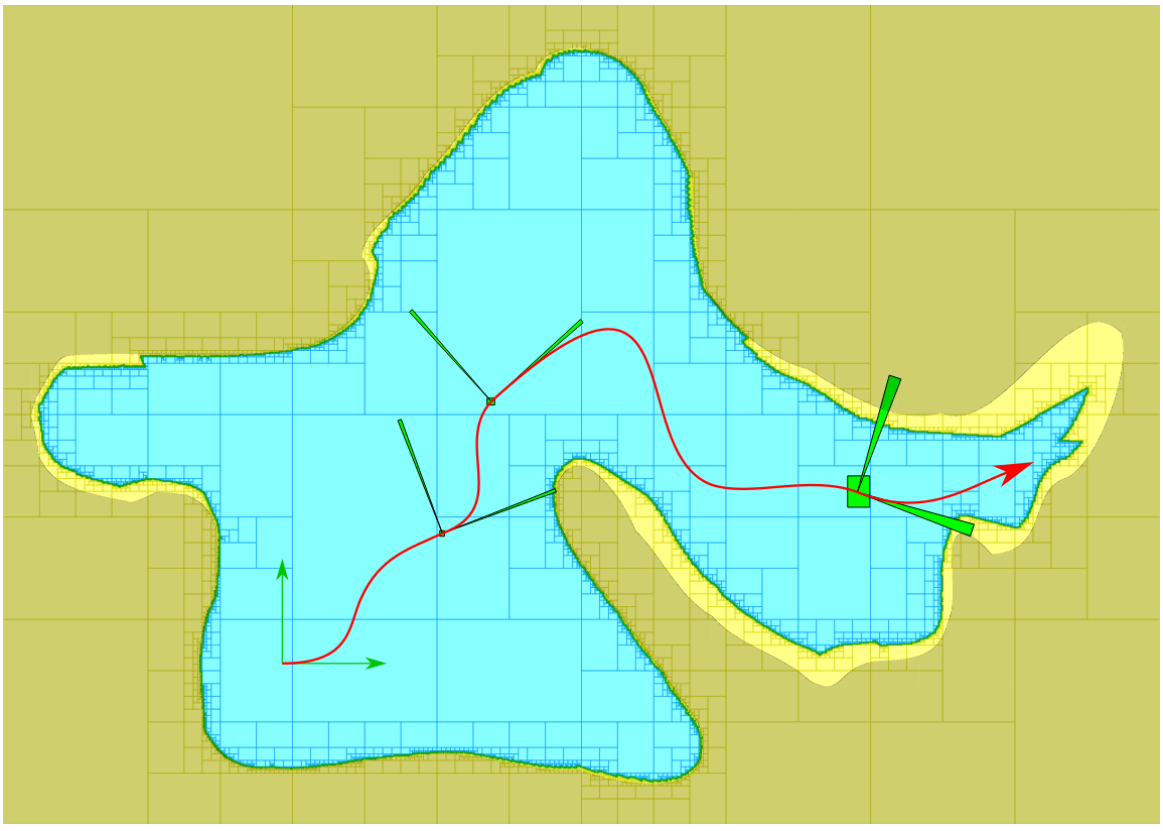


FIGURE 1.2 – Application du recalage avec 4 prises d'observations le long de la trajectoire inconnue (rouge). Une fois ramené dans le repère $\mathcal{R}_0$, on construit incrémentalement une zone de l'espace $\bigcup_{0\leq i\leq 3} \mathbb{W}_0(t_i)$ dépourvue d'obstacle (bleue), ainsi que des ensembles de position et orientation (vert) par lesquels la trajectoire est assurée de passer. La zone assombrie correspondant à l'ensemble réel, le manque à gagner sur la cartographie apparaît en jaune clair.

1.3.2 Recalage ensembliste sur ensembles complets

La seconde contribution de cette thèse envisage un autre scénario dans lequel tout l'ensemble est parfaitement connu, que se soit son intérieur, extérieur ou frontière. Cependant, la transformation considérée est plus compliquée avec une dimension réelle supplémentaire pour le facteur de mise à l'échelle uniforme.

Un aéronef d'assiette et de roulis nuls réalise deux prises de vues aériennes d'un environnement plat à des positions, orientations et altitudes différentes aux instants t_0 et t_1. Les environnements $\mathbb{M}_0$ et $\mathbb{M}_1$ sont extraits à partir des photos aériennes par binarisation (segmentation des images). Cela nous permet d'avoir une connaissance de tout l'environnement à la fois.

Une fois les déformations de projection compensées le cas échéant, la transformation liant deux prises de vues est une transformation de similarité 2D, à savoir une composition de rotation, translation et mise à l'échelle uniforme.

En réutilisant les notations précédentes, le problème se modélise comme un problème de recalage classique :

$$\mathbb{M}_i = \mathbf{h}(\mathbf{x}(t_i), \mathbb{M}_0) \tag{1.9}$$

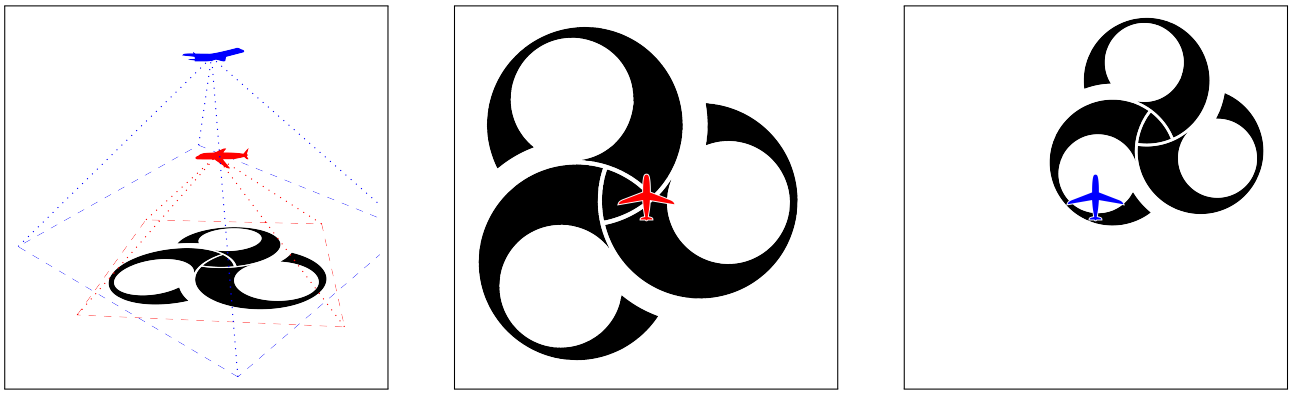


FIGURE 1.3 – Un aéronef passe deux fois au dessus d'une scène particulière et prend une photo à deux instants différents, avec respectivement des positions, orientations et altitudes différentes. Ces deux photos diffèrent alors d'une transformation dite de similarité, alliant translation, rotation et mise à l'échelle, chacun des paramètres étant respectivement liés à la différence de position, orientation et altitude de l'aéronef entre les deux instants.

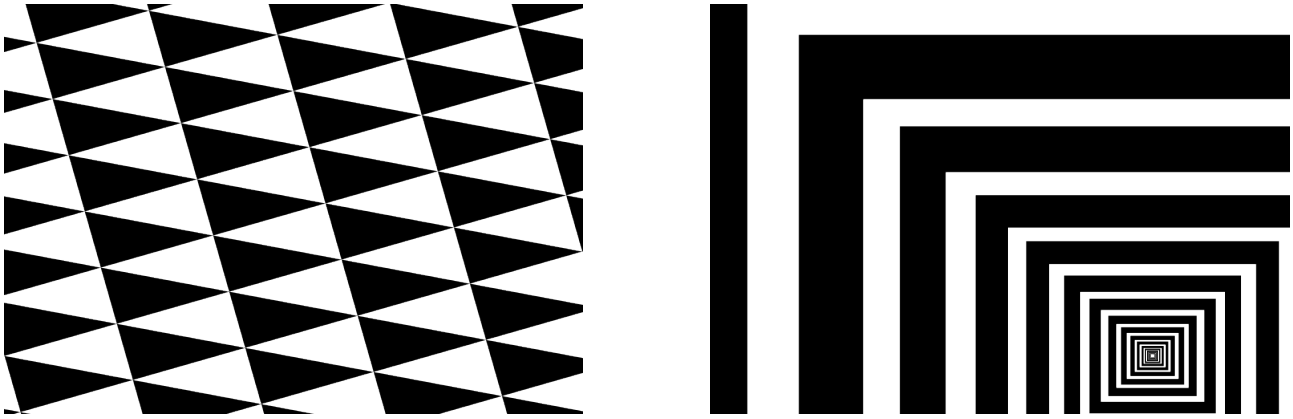
1.3.3 Remarques sur le problème de recalage ensembliste

Comme nous le verrons dans ce manuscrit, le problème de recalage est un problème commun à beaucoup de domaines, et dispose par conséquent de nombreuses méthodes de résolution. Celles-ci se concentrent sur des problèmes d'alignement de nuages de points, adaptés aux méthodes d'optimisation, et sont alors capables de retourner une solution quoiqu'il arrive, ce qui reste pratique dans des contextes réels où les nuages de points ne correspondent pas entièrement.

En contrepartie, le défaut de ne proposer qu'une seule solution met de côté une spécificité du problème : celui-ci peut présenter des solutions multiples, cf. figure 1.3 qui présente 3 solutions du fait de la symétrie évidente du triskel.

Les méthodes ensemblistes développées dans cette thèse proposent une résolution exhaustive du problème, même lorsque des solutions multiples par rotation apparaissent : l'ensemble des solutions possibles est caractérisé.

D'autres symétries, telles que celles présentées dans la figure 1.4, mènent à d'autres solutions multiples. Cependant, elles n'existent que dans des contextes d'ensembles non-bornés, qui ne seront pas étudiés dans ce manuscrit.



(a) Ensemble symétrique par translation. (b) Ensemble symétrique par mise à l'échelle.

FIGURE 1.4 – Exemples d'autres symétries menant à des solutions multiples, non traitées dans cette thèse.

1.4 Plan de la thèse

La partie I présente une revue de la littérature sur les problèmes de la thèse, avec d'une part la représentation et manipulation d'ensembles par les méthodes intervalles au chapitre 2, et d'autre part une présentation du problème de SLAM en robotique au chapitre 3.

La partie II est dédiée à la première contribution de cette thèse. Comme vu ci-dessus, celle-ci s'adresse au problème de SLAM dans un contexte où l'environnement peut être représenté en deux dimensions et le robot muni de capteurs *range and bearing* (chapitre 4). Les pavages épais réguliers, une nouvelle version de pavage issue des pavages intervalles, est définie au chapitre 5 afin de répondre à cette problématique.

La partie III présente la seconde contribution de la thèse, à savoir un problème de recalage avec une transformation de similarité en dimension 2. Dans cette partie, nous utilisons la méthode du cercle minimal afin de calculer les paramètres permettant une normalisation des ensembles (chapitre 7) avant de procéder à un recalage avec des outils ensemblistes appelés séparateurs (chapitres 8 et 9).

Première partie

État de l'art et analyse

Chapitre 2

CSP à domaines continus sur variables réelles, trajectoires, sous-ensembles de $\mathbb{R}^n$

Plan du chapitre

Comme nous l'avons vu en introduction, les problèmes étudiés dans cette thèse feront intervenir des variables à domaines continus variés :

- des vecteurs à valeurs réelles pour les déplacements $\mathbf{x}_{\mathcal{R}_|}(t_i)$,
- des trajectoires (fonctions de $\mathbb{R}$ dans $\mathbb{R}^n$) pour la trajectoire du robot $\mathbf{x}(\cdot)$,
- des sous-ensembles de $\mathbb{R}^n$ pour l'environnement $\mathbb{M}$ et ses différentes composantes et projections $\mathbb{M}_i, \mathbb{W}_i(t_j), \mathbb{Z}_i(t_j)$.

Afin de rester proche de la déclaration des problèmes vus en introduction, nous allons nous appuyer sur le cadre des problèmes de satisfaction de contraintes (*Constraint Satisfaction Problems* ou **CSPs** [3]). En particulier, comme ces problèmes font intervenir plusieurs types de variables, ce type de CSP est désigné comme un *CSP hybride* (cf. section III de [4]).

Dans ce chapitre, nous allons aborder cette notion de CSP hybride, comment le résoudre, et étudier chacun des domaines continus apparaissant dans les problèmes de cette thèse.

2.1 CSP à variables dans $\mathbb{R}^n$, $\mathcal{F}(\mathbb{R},\mathbb{R}^n)$ et $\mathcal{P}(\mathbb{R}^n)$

Un CSP est un cadre utilisé pour décrire des problèmes de recherche dans lesquels il s'agit de trouver une assignation de valeurs aux variables qui soit cohérente avec un ensemble donné de contraintes sur ces variables.

Nous donnons les définitions de CSP et CSP hybride, montrons comment ils peuvent se résoudre et soulignons comment ils peuvent répondre à nos objectifs.

Définition 1. *Un problème de satisfaction de contraintes (**CSP**) est un triplet d'ensembles :*

- $\mathcal{V} = (v_1, \ldots, v_n)$ *: ensemble de* ***variables****,*
- $\mathcal{D} = (\mathbb{D}_1, \ldots, \mathbb{D}_n)$ *: ensemble de* ***domaines*** *où $\mathbb{D}_i$ est l'ensemble des valeurs que peut prendre v_i,*
- $\mathcal{K} = (\mathcal{L}_1, \ldots, \mathcal{L}_m)$ *: ensemble de* ***contraintes*** *où $c_j : \mathcal{D} \to \{\bot, \top\}$. Lorsque, pour une affectation $d \in \mathcal{D}$, $c(d) = \top$, alors d satisfait c.*

Une solution est une affectation qui vérifie toutes les contraintes.

Dans la littérature, les variables les plus communes sont booléennes, discrètes ou continues, avec pour domaines respectifs des ensembles finis pour les deux premiers, et des intervalles réels pour le dernier. De plus, tous ces domaines sont cohérents, puisque ce sont tous des sous-ensembles de $\mathbb{R}$. Dans notre application cependant, nous considérons à la fois des variables de type trajectoire et des sous-ensembles réels, impliquant des domaines différents. Un tel CSP est dit hybride [4].

La résolution d'un CSP passe par l'élimination de valeurs dans les domaines, une opération dite de *filtrage*, ou encore de *contraction* pour les domaines continus. Ainsi, les seules valeurs éliminées sont des valeurs non-cohérentes, c'est-à-dire ne faisant partie d'aucune solution.

Exemple 1. *Exemple de CSP continu :*

$$\begin{cases} \textbf{\textit{Variables :}}\ x_1,\ x_2 \\ \textbf{\textit{Domaines :}}\ \mathbb{D}_1 = \mathbb{D}_2 = [-\infty, +\infty] \\ \textbf{\textit{Contrainte :}} \\ \qquad x_1^2 + x_2^2 \in [9, 25] \end{cases} \tag{2.1}$$

Dans cet exemple, on recherche tous les points du plan $\mathbb{R}^2$ dont la distance au carré à l'origine est entre 9 et 25, c'est-à-dire un anneau centré en l'origine, de rayon intérieur 3 et de rayon extérieur 5. Cet exemple servira ensuite à illustrer les différents outils de contraction introduits.

Dans les trois sections suivantes, nous allons passer en revue chacun des trois types de variables, à savoir les variables réelles dans $\mathbb{R}^n$ en section 2.2, les trajectoires dans $\mathcal{F}(\mathbb{R},\mathbb{R}^n)$ en section 2.3, et les sous-ensembles réels dans $\mathcal{P}(\mathbb{R}^n)$ à la section 2.5. Pour ces trois types, nous allons voir quels domaines nous pouvons leur associer ainsi que le fonctionnement d'opérateurs de filtrage sur ces domaines.

En particulier, le domaine privilégié pour les domaines continus, dans la littérature et dans ce manuscrit, est celui d'*intervalle réel*. La manipulation des intervalles fait l'objet de l'*analyse par intervalles*,

développée dans la prochaine section. Nous verrons alors comment utiliser cette dernière pour proposer des opérateurs de filtrage pour les intervalles, appelés **contracteurs**, puis comment généraliser les intervalles réels aux autres domaines.

2.2 Intervalles de réels et analyse par intervalles

2.2.1 Définitions : intervalles et boîtes

L'analyse par intervalles [5, 6] est une branche des mathématiques numériques dédiée à la gestion de l'incertitude et des erreurs de calculs. Pour cela, elle s'appuie sur l'utilisation d'intervalles réels afin d'encadrer les valeurs réelles que l'on cherche à calculer.

Définition 2. *Dans cette thèse, un **intervalle** est un sous-ensemble connexe réel. Soient $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ avec $a \leq b$. L'intervalle $[a, b]$ est défini comme :*

$$[a, b] := \{x \in \mathbb{R} \mid a \leq x \leq b\}. \tag{2.2}$$

Tous les intervalles seront considérés fermés, même lorsque les bornes sont infinies. L'ensemble des intervalles réels est noté $\mathbb{IR}$.

Un intervalle sera noté entre crochets. De plus, pour un réel x, on notera $[x] \in \mathbb{IR}$ un intervalle contenant $x \in [x]$. Les bornes d'un tel intervalle seront notées ainsi : $[x] = [x^-, x^+]$.

Définition 3. *Une **boîte** est un produit cartésien d'intervalles. Par conséquent, l'ensemble des boîtes réelles est noté $\mathbb{IR}^n$, où n est la dimension de la boîte. Dans ce rapport, les vecteurs et boîtes seront notés en gras $[\mathbf{x}] \in \mathbb{IR}^n$, $[\mathbf{x}] = [\mathbf{x}^-, \mathbf{x}^+]$. En dimension 2, une boîte correspond à un rectangle aligné avec les axes, et plus généralement à un parallélotope de dimension n aligné avec les axes.*

2.2.2 Algèbre ensembliste appliquée aux intervalles

Les intervalles étant des sous-ensembles réels, nous verrons dans la suite que ceux-ci peuvent être utilisés dans les méthodes ensemblistes. Pour ce faire, nous nous intéressons tout d'abord à l'algèbre ensembliste.

2.2.2.1 Algèbre des parties d'un ensemble

Les sous-ensembles réels, de n'importe quelle dimension, seront notés avec des caractères en doubles barres : $\mathbb{X}, \mathbb{Y}$.

Nous allons considérer la relation d'ordre d'inclusion :

$$\mathbb{X} \subset \mathbb{Y} \iff \forall \mathbf{x} \in \mathbb{X}, \mathbf{x} \in \mathbb{Y}. \tag{2.3}$$

Il en découle l'égalité :

$$\mathbb{X} = \mathbb{Y} \iff (\mathbb{X} \subset \mathbb{Y}) \wedge (\mathbb{Y} \subset \mathbb{X}). \tag{2.4}$$

Les opérations classiques de l'algèbre des parties d'un ensemble en dimension n comprennent :

— l'*union* :

$$\mathbb{X} \cup \mathbb{Y} = \{\mathbf{z} \in \mathbb{R}^n \mid (\mathbf{z} \in \mathbb{X}) \vee (\mathbf{z} \in \mathbb{Y})\}, \tag{2.5}$$

— l'*intersection* :

$$\mathbb{X} \cap \mathbb{Y} = \{\mathbf{z} \in \mathbb{R}^n \mid (\mathbf{z} \in \mathbb{X}) \wedge (\mathbf{z} \in \mathbb{Y})\}, \tag{2.6}$$

— le *complémentaire* :

$$\overline{\mathbb{X}} = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbf{z} \notin \mathbb{X}\}. \tag{2.7}$$

Remarque 3. *Les autres opérations ensemblistes classiques se déduisent de ces trois opérations, notamment la différence ensembliste :*

$$\mathbb{X} \setminus \mathbb{Y} = \mathbb{X} \cap \overline{\mathbb{Y}}. \tag{2.8}$$

Dans le cadre de ce manuscrit, nous allons aussi considérer des opérations sur des sous-ensembles de dimensions différentes, respectivement m pour $\mathbb{X}$ et n pour $\mathbb{Y}$:

— le *produit cartésien* :

$$\mathbb{X} \times \mathbb{Y} = \left\{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{m+n} \mid (\mathbf{x} \in \mathbb{X}) \wedge (\mathbf{y} \in \mathbb{Y})\right\}, \tag{2.9}$$

— la *projection* sur un sous-ensemble I de ses dimensions :

$$\mathrm{proj}_I \mathbb{X} = \left\{\mathbf{z} \in \mathbb{R}^{|I|} \mid \exists \mathbf{x} \in \mathbb{X},\, \mathbf{z} = (x_i)_{i \in I}\right\}, \tag{2.10}$$

qui est une projection sur les coordonnées notées $\mathbf{z}$.

2.2.2.2 Algèbre des sous-parties d'un ensemble réel étendue aux intervalles réels

Une partie des opérations de l'algèbre des parties d'un ensemble s'applique bien aux intervalles de réels :

— l'intersection, l'inclusion et l'égalité sont stables sur les intervalles et les boîtes,

— les opérations nécessitant des dimensions multiples, à avoir le produit cartésien et la projection sur certaines de ses dimensions, sont stables sur les boîtes.

Cependant, les intervalles de réels ne sont pas stables par union et complémentaire. Pour ce faire, nous utilisons classiquement un opérateur permettant d'obtenir un intervalle à partir d'un sous-ensemble de $\mathbb{R}$ quelconque.

Définition 4. *L'**enveloppe intervalle** d'un sous-ensemble $\mathbb{X} \subset \mathbb{R}$ est le plus petit intervalle, au sens de l'inclusion, contenant $\mathbb{X}$. L'opérateur associé est noté $\square$, et se définit plus formellement comme :*

$$\begin{aligned} \square : \mathcal{P}(\mathbb{R}) &\to \mathbb{IR} \\ \mathbb{X} &\mapsto \bigcap_{[x] \supset \mathbb{X}} [x]. \end{aligned} \tag{2.11}$$

Exemple 2. *Soit $\mathbb{X} = \{2, 3\} \cup [0, 1]$. Alors, $\square(\mathbb{X}) = [0, 3]$.*

Avec cet opérateur, on peut alors définir des nouveaux opérateurs pour l'union et le complémentaire qui soient stables pour les intervalles :

— l'union intervalle notée $\sqcup$:

$$[x] \sqcup [y] = \square([x] \cup [y]), \tag{2.12}$$

— le complémentaire intervalle $\square\left(\overline{[x]}\right)$, noté simplement $\overline{[x]}$ par abus de notation.

Remarque 4. *Il faut cependant noter que la distributivité et la propriété du complémentaire sont perdues sur les intervalles, comme illustré dans l'exemple 3.*

Exemple 3. *Dans cet exemple, les trois intervalles ne vérifient pas la propriété de distributivité avec l'opérateur $\sqcup$:*

$$[0,1] \sqcup ([2,3] \cap [4,5]) = [0,1]$$
$$([0,1] \sqcup [2,3]) \cap ([0,1] \sqcup [4,5]) = [0,2]$$

Le complémentaire de l'intervalle suivant, au sens intervalle, n'est pas le complémentaire au sens ensembliste :

$$[0,1] \cap \overline{[0,1]} = [0,1] \cap [-\infty, +\infty] = [0,1] \neq \emptyset.$$

2.2.3 Arithmétique réelle étendue aux ensembles et intervalles

Nous avons évoqué que les intervalles de réels étaient utilisés afin de gérer les incertitudes et les erreurs de calcul. L'arithmétique intervalle est une technique mathématique qui manipule non pas des réels, mais des intervalles qui les encadrent. Elle a pour but de tenir compte des incertitudes sur les réels manipulés, tout en garantissant que le vrai résultat se trouve dans l'intervalle obtenu.

Nous commençons par généraliser l'arithmétique aux ensembles de réels, avant de nous intéresser aux intervalles.

2.2.3.1 Arithmétique réelle sur les sous-ensembles réels

Nous nous intéressons à généraliser l'arithmétique réelle sur des ensembles de réels au moyen de deux opérations supplémentaires. Soient $\mathbb{X} \subset \mathbb{R}^m$ et $\mathbb{Y} \subset \mathbb{R}^n$ deux sous-ensembles réels et soit une fonction arithmétique $\mathbf{f} : \mathbb{X} \to \mathbb{Y}$.

L'*image directe* de $\mathbb{X}' \subset \mathbb{X}$ par $\mathbf{f}(\cdot)$ est :

$$\mathbf{f}(\mathbb{X}') := \left\{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathbb{X}'\right\}. \tag{2.13}$$

L'*image réciproque* de $\mathbb{Y}' \subset \mathbb{Y}$ par $\mathbf{f}(\cdot)$ est :

$$\mathbf{f}^{-1}(\mathbb{Y}') := \left\{\mathbf{x} \in \mathbb{X} \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}'\right\}. \tag{2.14}$$

Tout particulièrement, l'image directe définit une extension d'une fonction réelle aux ensembles, qui a pour propriété :

$$\mathbf{f}(\mathbb{X}') = \mathbb{Y}' \implies \forall \mathbf{x} \in \mathbb{X}', \mathbf{f}(\mathbf{x}) \in \mathbb{Y}'. \tag{2.15}$$

Ainsi, calculer l'image directe d'un ensemble $\mathbb{X}'$ donne un ensemble qui est garanti de contenir n'importe quelle image $\mathbf{f}(\mathbf{x})$, et ce quel que soit le $\mathbf{x}$ choisi dans $\mathbb{X}'$. Ainsi, si l'ensemble $\mathbb{X}'$ représente le champ des

possibles pour $\mathbf{x}$, c'est-à-dire son incertitude, alors cette arithmétique ensembliste permet de donner un résultat garanti de contenir la réalité.

Les ensembles n'étant pas des objets faciles à manipuler, l'arithmétique intervalle fait le choix de se restreindre aux intervalles qui sont bien plus faciles à représenter et manipuler.

2.2.3.2 Arithmétique intervalle élémentaire

Dans le cadre des ensembles de réels, la définition donnée n'est pas forcément implémentable en machine, ce que l'arithmétique intervalle cherche à être. Nous commençons par les opérations arithmétiques classiques. Si $\diamond \in \{+, -, *, /\}$ est un opérateur binaire arithmétique classique, alors :

$$[x] \diamond [y] = \square(\{x \diamond y \mid x \in [x] \wedge y \in [y]\}). \tag{2.16}$$

Pour permettre une implémentation, il est possible de raisonner sur les bornes uniquement. Ainsi, on a :

— l'addition :
$$[x] + [y] = [x^- + y^-, x^+ + y^+], \tag{2.17}$$

— la soustraction :
$$[x] - [y] = [x^- - y^+, x^+ - y^-], \tag{2.18}$$

— la multiplication :
$$[x] * [y] = [\min(x^-y^-, x^-y^+, x^+y^-, x^+y^+), \max(x^-y^-, x^-y^+, x^+y^-, x^+y^+)], \tag{2.19}$$

— l'inverse :
$$\frac{1}{[x]} = \begin{cases} \emptyset & \textbf{si } [x] = [0,0] \\ [1/x^+, 1/x^-] & \textbf{si } 0 \notin [x] \\ [1/x^+, +\infty] & \textbf{si } x^- = 0 \text{ et } x^+ > 0\,, \\ [-\infty, 1/x^-] & \textbf{si } x^- < 0 \text{ et } x^+ = 0 \\ [-\infty, +\infty] & \textbf{si } x^- < 0 \text{ et } x^+ > 0 \end{cases} \tag{2.20}$$

— la division se déduit de la composition entre l'inverse et la multiplication :
$$\frac{[x]}{[y]} = [x] * \frac{1}{[y]}. \tag{2.21}$$

2.2.3.3 Fonctions arithmétiques élémentaires

Les fonctions arithmétiques élémentaires, telles que exp, cos, sin, sont aussi facilement étendues aux intervalles en étudiant leur monotonie ou monotonie par morceaux. Ainsi, sur des fonctions monotones, telles que l'exponentielle :

$$\exp([x]) = [\exp(x^-), \exp(x^+)].$$

Sur les fonctions non-monotones, comme le sinus, cette propriété ne s'applique pas directement, mais il est possible de décomposer en parties monotones, par exemple :

$$\sin([0,4]) = \sin([0,\pi/2]) \sqcup \sin([\pi/2,4]) = [0,1] \sqcup [\sin(4),1] = [\sin(4),1].$$

Avec toutes ces fonctions élémentaires, il est possible de s'intéresser à l'extension des fonctions arithmétiques réelles aux intervalles qui sont une composition de ces fonctions élémentaires.

2.2.3.4 Fonctions d'inclusion

Afin d'étendre l'arithmétique réelle aux intervalles, nous définissons ce qu'est une fonction d'inclusion.

Définition 5. *Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction réelle. Soit $f([x])$ son image directe (voir équation (2.13)) Une fonction intervalle $[f]$ de $\mathbb{IR}$ dans $\mathbb{IR}$ est une **fonction d'inclusion**, ou **extension**, de f si :*

$$\forall [x] \in \mathbb{IR},\ f([x]) \subset [f]([x]). \tag{2.22}$$

*Une fonction d'inclusion est dite **minimale** si dans (2.22) l'inclusion est remplacée par une égalité.*

Une fonction d'inclusion permet donc de calculer une surestimation de l'*image directe* d'une fonction pour les intervalles. Calculer une surestimation n'est pas un problème en pratique pour les algorithmes basés sur les méthodes à intervalles, qui recherchent l'exhaustivité. La précision vient avec la convergence rapide de ces méthodes, qui nécessitent alors des fonctions d'inclusion rapides à calculer.

Toutes les extensions aux intervalles des opérations binaires et fonctions élémentaires présentées ci-dessus sont donc des fonctions d'inclusion minimales pour leur opération ou fonction arithmétique correspondante.

Il existe différentes manières de calculer des fonctions d'inclusion $[f]$ pour une fonction réelle f donnée. Lorsque l'on transpose directement l'expression de la fonction en intervalle, et en remplaçant chaque occurrence de la variable par l'intervalle dont on cherche à calculer l'image, nous obtenons la fonction d'inclusion dite *naturelle*.

Cependant, cette fonction d'inclusion n'est en général pas minimale, comme illustré dans l'exemple ci-dessous :

Exemple 4. *Soit $f(x) = 2x^3 - 5x^2 + 3x - 7$ avec $x \in [0, 2]$ une fonction polynomiale. Sa fonction d'inclusion naturelle $[f]_N$ est donnée par :*

$$[f]_N([x]) = 2[x]^3 - 5[x]^2 + 3[x] - 7.$$

Ainsi :

$$\begin{aligned}[f]_N([0,2]) &= 2[0,2]^3 - 5[0,2]^2 + 3[0,2] - 7 \\ &= 2[0,8] - 5[0,4] + [0,6] - 7 = [-20, 22].\end{aligned}$$

Afin d'atténuer ces effets, d'autres approches sont possibles. En utilisant un développement de Taylor [7, 6], on obtient une autre fonction d'inclusion. Celle-ci donne généralement de meilleurs résultats que la fonction d'inclusion naturelle lorsque les intervalles sont petits.

Définition 6. *Pour un intervalle $[x]$, notons $x_c = \frac{1}{2}(x^+ - x^-)$ son centre. Pour une fonction f, son extension de Taylor à l'ordre deux est définie de la manière suivante :*

$$[f]_{T,2}([x]) = f(x_c) + f'(x_c) \cdot ([x] - x_c) + f''([x]) \frac{([x] - x_c)^2}{2} \tag{2.23}$$

Exemple 5. *En gardant les mêmes fonction f et intervalle $[0,2]$ en entrée que l'exemple précédent, le développement de Taylor au second ordre donne :*

$$[f]_{T,2}([0,2]) = -7 + [-1,1] + (12 \cdot [0,2] - 10)\frac{[0,1]}{2} = [-13,1].$$

Il est aussi possible d'étudier la monotonie de la fonction comme mentionné dans la section 2.2.3.3. Cela permet de donner une fonction d'inclusion minimale lorsque la fonction est dérivable à dérivée continue.

Exemple 6. *Toujours en utilisant la même fonction et intervalle que précédemment, nous commençons par étudier la monotonie de la fonction :*

$$f'(x) = 6x^2 - 10x + 3$$

On trouve les deux racines $(r_1, r_2) = \left(\frac{1}{6}(5-\sqrt{7}), \frac{1}{6}(5+\sqrt{7})\right)$, où r_1 est un maximum local et r_2 un minimum local. On en déduit une fonction d'inclusion :

$$[f]_M([0,2]) = [\min(f(0), f(r_2)), \max(f(r_1), f(2))] \subset [-7.1578, -5].$$

L'ensemble de ces outils permettent d'obtenir une arithmétique intervalle et de garantir les résultats des calculs numériques.

2.2.3.5 Généralisation en dimension supérieure

Toutes les opérations précédentes s'étendent naturellement aux boîtes :

— l'intersection :

$$[\mathbf{x}] \cap [\mathbf{y}] = ([x_1] \cap [y_1]) \times \cdots \times ([x_n] \cap [y_n]), \tag{2.24}$$

— l'union intervalle :

$$[\mathbf{x}] \sqcup [\mathbf{y}] = ([x_1] \sqcup [y_1]) \times \cdots \times ([x_n] \sqcup [y_n]), \tag{2.25}$$

— l'enveloppe intervalle est étendue à la plus petite boîte contenant un sous-ensemble de $\mathbb{R}^n$,

— les équations binaires $\diamond$ entre deux vecteurs comme la somme, la différence, le produit scalaire :

$$[\mathbf{x}] \diamond [\mathbf{y}] = \square(\{\mathbf{x} \diamond \mathbf{y} \mid (\mathbf{x} \in [\mathbf{x}]) \wedge (\mathbf{y} \in [\mathbf{y}])\}). \tag{2.26}$$

Remarque 5. *Il est possible de définir une boîte autrement : considérons la relation d'ordre $\leq$ entre vecteurs réels de dimension n définie de la manière suivante :*

$$\mathbf{x} \leq \mathbf{y} \iff \forall i \in \{1, \ldots, n\},\ x_i \leq y_i. \tag{2.27}$$

Dans ce cas là, une boîte est un intervalle pour les vecteurs de réels :

$$[\mathbf{x}] = [\mathbf{x}^-, \mathbf{x}^+] = \left\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}^- \leq \mathbf{x} \leq \mathbf{x}^+\right\}. \tag{2.28}$$

Il est aussi possible d'étendre les intervalles aux matrices à coefficients réels. Une matrice intervalle est une matrice à coefficients intervalles réels.

2.2.3.6 Causes du pessimisme de l'arithmétique intervalle

Nous avons vu que les fonctions d'inclusion génèrent du pessimisme dans les différents exemples ci-dessus. Ses causes principales sont :

- les domaines de définition et discontinuités : nous avons $1/[-1,1] = [-\infty,+\infty]$ mais pourtant pour tout x de $[-1,1]$, $1/x \neq 0$.
- les **occurrences multiples** de variable (ou dépendances entre variables) entraînent aussi des surestimations puisque chaque occurrence est traitée séparément. Par exemple, l'équation $x - x$ évaluée de façon intervalle sur $[x] = [0,1]$ donne :

$$[x] - [x] = [0,1] - [0,1] = [-1,1].$$

- l'**effet d'enveloppe** (*wrapping effect*) survient lorsque l'image d'un intervalle ou d'une boîte par une fonction n'est pas un intervalle. L'enveloppe intervalle de l'image utilisée par les méthodes à intervalles introduit des surestimations. En répétant successivement une opération, les surestimations peuvent augmenter exponentiellement, comme montré dans la figure 2.1.

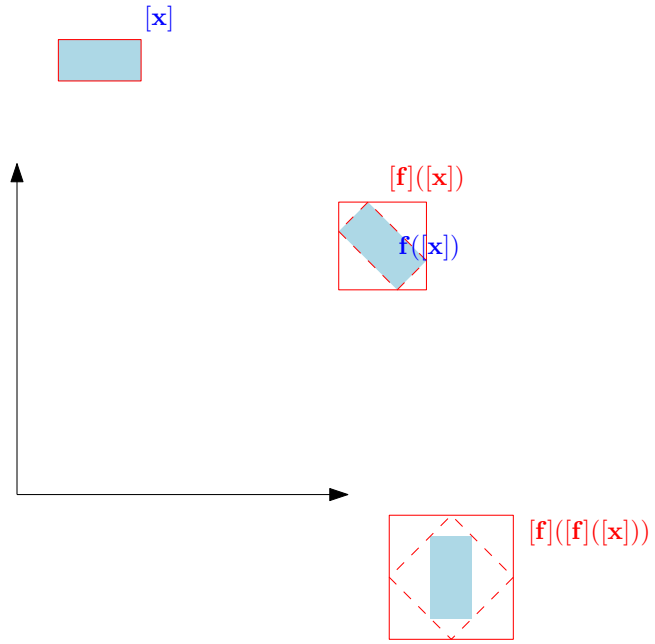


FIGURE 2.1 – Phénomène de *wrapping effect* apparaissant lors d'une application d'une rotation **f** de 45° en sens horaire à une boîte [**x**]. La boîte après deux rotations n'est pas minimale en dépit de la minimalité des opérations individuelles.

2.2.4 Contraction de boîtes

L'opération qui consiste à filtrer des domaines continus réels est appelée **contraction**, et l'opérateur est un **contracteur**. Historiquement, on construit un contracteur associé à une contrainte, et le contracteur permet alors de *contracter* les domaines en agissant sur les bornes de la boîte afin d'éliminer des valeurs qui ne sont présentes dans aucune solution de la contrainte.

Définition 7. *L'opérateur $\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$ est un **contracteur** associé à une contrainte $\mathcal{L}$ s'il vérifie deux propriétés :*

$$\begin{array}{ll} \forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & \text{contractance} \\ (\mathbf{x} \in [\mathbf{x}] \wedge \mathcal{L}(\mathbf{x})) \implies \mathbf{x} \in \mathcal{C}([\mathbf{x}]) & \text{complétude} \end{array}$$

Exemple 7. *Reprenons l'exemple de l'anneau précédent :*

$$\begin{cases} \textbf{\textit{Variables :}}\ x_1,\ x_2 \\ \textbf{\textit{Domaines :}}\ \mathbb{D}_1 = \mathbb{D}_2 = [-\infty, +\infty] \\ \textbf{\textit{Contrainte :}} \\ \qquad x_1^2 + x_2^2 \in [9, 25] \end{cases} \tag{2.29}$$

La figure 2.2 illustre un contracteur potentiel sur cet exemple. Les détails du fonctionnement de ce contracteur sont évoqués dans la suite.

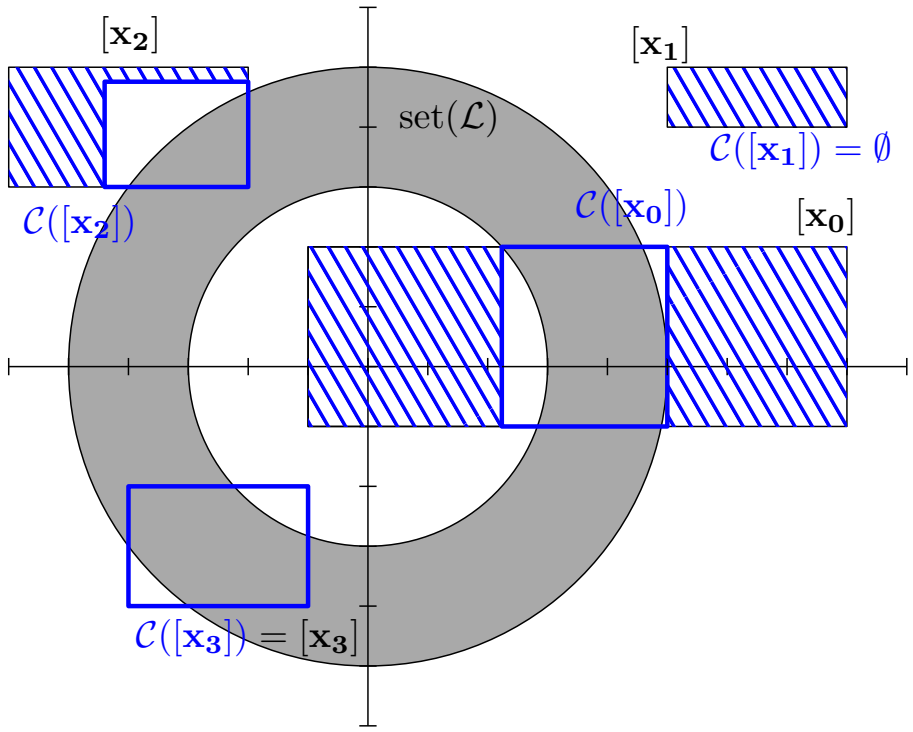


FIGURE 2.2 – Exemple de contracteur $\mathcal{C}$ sur la contrainte $\mathcal{L}$: $x_1^2 + x_2^2 \in [9, 25]$. Les boîtes initiales sont contractées en boîtes bleues sans perte de solution. Les parties hachurées sont celles éliminées par le contracteur.

Il est possible d'associer un ensemble à chaque contrainte et chaque contracteur.

Définition 8. *Ensembles associés aux contraintes et contracteurs :*

— *ensemble solution d'une contrainte* $\mathcal{L}$ *:*

$$\text{set}(\mathcal{L}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathcal{L}(\mathbf{x})\}, \tag{2.30}$$

— *ensemble associé à un contracteur* $\mathcal{C}$ *:*

$$\text{set}(\mathcal{C}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathcal{C}(\{\mathbf{x}\}) = \{\mathbf{x}\}\}. \tag{2.31}$$

Ainsi par définition, un contracteur $\mathcal{C}$ est associé à une contrainte $\mathcal{L}$ si :

$$\text{set}(\mathcal{C}) \supset \text{set}(\mathcal{L}). \tag{2.32}$$

Définition 9. *Un contracteur associé à une contrainte est dit* ***minimal*** *si pour chaque boîte* $[\mathbf{x}]$*, le contracteur retourne la plus petite boîte contenant l'ensemble solution, à savoir* $\square([\mathbf{x}] \cap \text{set}(\mathcal{L}))$*. Il est* ***asymptotiquement minimal*** *si* $\text{set}(\mathcal{C}) = \text{set}(\mathcal{L})$*. Notons qu'un contracteur minimal est asymptotiquement minimal.*

Nous avons vu ce que sont les contracteurs, notamment leur capacité à réduire la taille de boîte par rapport à l'ensemble solution d'une contrainte associée. La définition d'un contracteur à partir de la déclaration d'une contrainte est alors un enjeu important pour la résolution de CSP. C'est pourquoi nous définissons maintenant une algèbre des contracteurs, basée sur l'arithmétique intervalle, qui permet de proposer des contracteurs associés aux contraintes lorsque celles-ci sont exprimées comme des expressions arithmétiques réelles ou intervalles.

2.2.4.1 Algèbre des contracteurs

Intéressons-nous maintenant à la manipulation des contracteurs en nous appuyant sur le paradigme introduit dans *contractor programming* [8]. Dans celui-ci, le raisonnement est de considérer les contracteurs comme des sous-ensembles réels, en l'occurrence leur ensemble associé. Ce raisonnement se traduit par une algèbre des contracteurs autorisant les opérations ensemblistes :

— l'intersection :
$$(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]), \tag{2.33}$$

— l'union :
$$(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]), \tag{2.34}$$

— la composition :
$$(\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1(\mathcal{C}_2([\mathbf{x}])). \tag{2.35}$$

Ces opérations entre contracteurs sont cohérentes avec les ensembles : une intersection entre contracteurs produira un contracteur cohérent avec l'intersection de leurs ensembles associés. La composition permet une séquence de contractions, voire même une itération du même contracteur jusqu'à stabilisation : le **point fixe**. En pratique, cette algèbre permet de fabriquer des contracteurs en décomposant les contraintes : une conjonction de contraintes s'obtient avec une intersection de contracteurs, une disjonction de contraintes s'obtient avec une union.

Le sujet de ce manuscrit reposant sur la manipulation de sous-ensembles réels, nous avons là une première manière de procéder : en considérant ces ensembles comme des ensembles solutions de contraintes, il devient possible d'effectuer les opérations ensemblistes présentées ci-dessus.

Par exemple, cherchons à caractériser une intersection $\mathbb{X} \cap \mathbb{Y}$. Pour cela, nous voulons un contracteur $\mathcal{C}$ tel que $\text{set}(\mathcal{C}) = \mathbb{X} \cap \mathbb{Y}$. Par l'algèbre des contracteurs, en posant $\mathbb{X} = \text{set}(\mathcal{C}_1)$ et $\mathbb{Y} = \text{set}(\mathcal{C}_2)$, nous avons :

$$\begin{aligned} \text{set}(\mathcal{C}) &= \mathbb{X} \cap \mathbb{Y} \\ &= \text{set}(\mathcal{C}_1) \cap \text{set}(\mathcal{C}_2) \\ &= \text{set}(\mathcal{C}_1 \cap \mathcal{C}_2) \end{aligned}$$

Le contracteur $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$ vérifie bien cette propriété. Pour résumer, *contractor programming* donne un cadre permettant de traduire directement une expression entre ensembles en une expression entre contracteurs, et le contracteur résultant est cohérent avec l'expression initiale.

L'algèbre des contracteurs contitue le cœur des outils mobilisés dans les deux contributions de cette thèse. Nous y mettrons en œuvre cette algèbre afin de définir les nouveaux opérateurs menant à une résolution ensembliste des problématiques de ces contributions.

2.2.4.2 Contracteur d'une contrainte arithmétique

Maintenant que nous avons vu comment appliquer des opérations ensemblistes aux contracteurs, regardons comment appliquer des opérations d'arithmétique réelle aux contracteurs, et comment les utiliser afin de construire un contracteur adapté à une contrainte arithmétique.

L'algorithme `HC4-Revise` [9] permet de construire un contracteur à partir de l'arbre d'expression d'une contrainte arithmétique, où chaque nœud correspond à une opération élémentaire et chaque feuille est une variable ou une constante. Un intervalle est associé à chaque nœud. Ensuite, `HC4-Revise` contracte ces intervalles par une approche en deux phases, illustrées dans l'exemple 8 :

- la phase d'évaluation (*forward*) (voir la figure 2.3 gauche) est une phase ascendante (*bottom-up*). Elle part des feuilles et évalue l'opération de chaque nœud en appliquant l'arithmétique intervalle,
- la phase de contraction (*narrowing phase* ou *backward*) (voir la figure 2.3 droite) est une phase descendante (*top-down*). Elle contracte l'intervalle de chaque nœud en appliquant cette fois les opérateurs inverses (*backward, cancel* ou *reverse*). Cet algorithme nécessite donc une définition d'opérateur inverse pour chaque opération élémentaire.

Exemple 8. *Le résultat de l'algorithme sur la boîte* $[\mathbf{x_0}] = [-1, 8] \times [-1, 2]$ *dans l'exemple de l'anneau est présenté sur la figure 2.2, et l'algorithme est explicité dans la figure 2.3.*

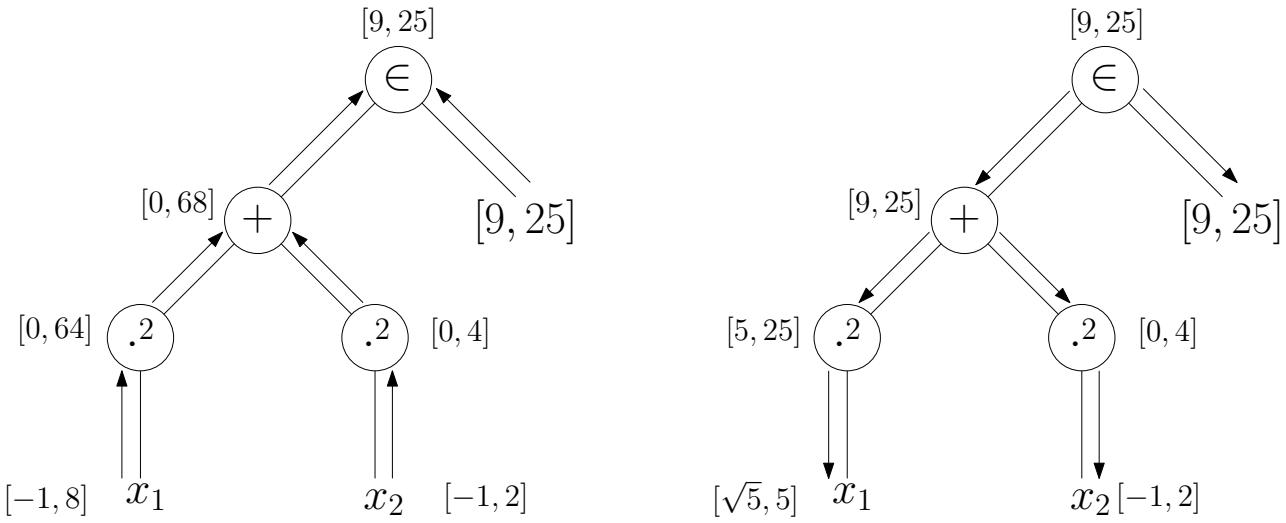


FIGURE 2.3 – Schéma du fonctionnement de l'algorithme `HC4-Revise` sur l'exemple de l'anneau. En partant d'une boîte initiale $[-1, 8] \times [-1, 2]$, l'algorithme évalue (gauche) l'intervalle de chaque nœud de manière ascendante, jusqu'à la racine. Ensuite dans une phase descendante (droite), chaque intervalle est contracté en retirant les valeurs incohérentes, selon chaque opérateur. Au final, la boîte est contractée pour obtenir $[\sqrt{5}, 5] \times [-1, 2]$.

Remarque 6. *Suivant comment* `HC4-Revise` *est implémenté, les contractions peuvent différer. Dans notre exemple 8, regardons de plus près la dernière contraction de l'intervalle de* x_1*. L'inverse de l'opération précédente – la mise au carré – donne le résultat* $[-5, -\sqrt{5}] \cup [\sqrt{5}, 5]$*. Une implémentation naïve utilisera l'union intervalle, soit le résultat* $[-5, 5]$*, qui ne permettra pas de contracter la borne inférieure de* x_1*. Une meilleure implémentation, gérant bien cette disjonction d'intervalles, permet d'obtenir le résultat optimal* $[\sqrt{5}, 5]$ *sur cet exemple. L'algorithme* `HC4-Revise` *utilisé dans ce manuscrit est celui présent dans la bibliothèque C++ IBEX [10]. Celle-ci s'appuie sur plusieurs implémentations possibles et choisit la meilleure selon leurs résultats. Dans notre exemple, l'algorithme renvoie bien la plus petite boîte possible.*

Afin de révéler l'ensemble set($\mathcal{C}$), il est possible d'appliquer le contracteur dans un algorithme de *branch-and-prune*, appelé **paveur**. La figure 2.4 illustre l'utilisation du contracteur précédent dans un paveur.

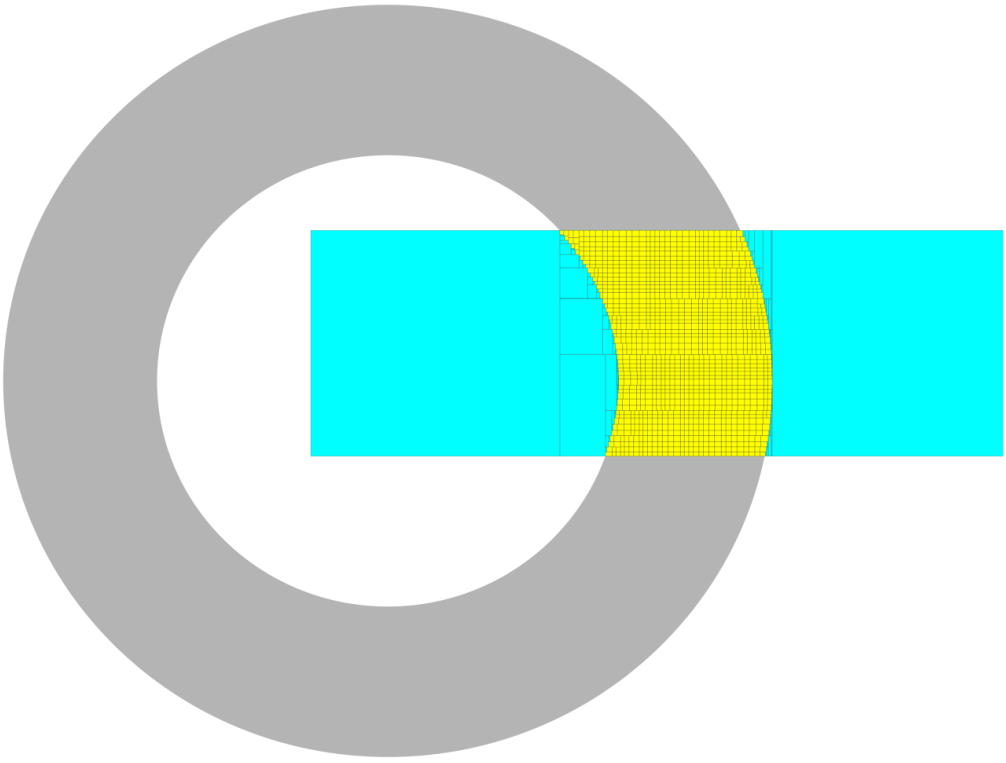


FIGURE 2.4 – Application du contracteur obtenu par `HC4-Revise` sur la boîte $[\mathbf{x_0}]$ dans un paveur. Les boîtes bleues sont les boîtes ne contenant aucune solution. Les boîtes contractées sont bissectées et le processus répété jusqu'à une taille limite. Comme ce contracteur est minimal, il est possible d'approcher l'ensemble solution de la contrainte.

Remarque 7. *La figure 2.4 montre la limite de ce type d'approche pour décrire un ensemble. En effet, un contracteur ne caractérise que l'extérieur de l'ensemble, et est donc inefficace pour caractériser son intérieur. La manière de traiter ce type de problème est d'utiliser un second contracteur [8], associé avec la négation de la contrainte pour caractériser l'intérieur de l'ensemble. Cette paire de contracteurs forme un nouvel opérateur appelé séparateur, et il sera abordé en détail dans le chapitre 8.*

Nous avons montré comment construire un contracteur à partir d'équations arithmétiques à l'aide de la méthode `HC4Revise`. Dans la littérature, on trouve des contracteurs pour d'autres contraintes communes – par exemple l'union de contracteurs pour représenter une disjonction de contraintes – ou basées sur des méthodes différentes. En pratique, on retrouve ces contracteurs dans un catalogue de contracteurs, comme par exemple la bibliothèque C++ IBEX [10] utilisée dans cette thèse.

2.3 Tubes comme domaines des variables trajectoires

Dans la section précédente, nous avons vu comment traiter des contraintes sur réels, c'est-à-dire des contraintes statiques. Dans cette section, nous allons étendre les contraintes que nous allons pouvoir traiter avec les contraintes dynamiques, faisant intervenir des variables évoluant dans le temps, très présentes dans le domaine de la robotique mobile, comme nous le verrons dans le chapitre 3. Elles sont modélisées comme des fonctions temporelles, de $\mathbb{R}$ dans $\mathbb{R}^n$, et sont appelées **trajectoires**. L'exemple 9 montre une situation type en robotique impliquant des trajectoires.

Exemple 9. *Considérons un robot dont les états $\mathbf{x}(\cdot)$, vus comme une trajectoire, décrivent à chaque instant sa position $(x_1(\cdot), x_2(\cdot))$ dans le plan $\mathbb{R}^2$ ainsi que son orientation $x_3(\cdot)$. Si ce robot évolue à une vitesse constante v connue, et que sa vitesse angulaire $\dot{x_3}(\cdot)$ est contrôlée par une commande $u(\cdot)$ connue, son évolution est déduite des équations différentielles suivantes :*

$$\begin{cases} \dot{x}_1(t) = v\cos(x_3(t)) \\ \dot{x}_2(t) = v\sin(x_3(t)) \\ \dot{x}_3(t) = u(t) \end{cases} \tag{2.36}$$

Ces trajectoires sont impliquées dans des équations (donc des contraintes) différentielles, qui font notamment intervenir l'arithmétique réelle.

Nous montrons comment traiter ces contraintes dans un CSP, en leur attribuant un domaine basé sur les intervalles appelé *tube* [11], et comment faire des contracteurs sur les tubes, notamment en étendant l'arithmétique intervalle aux tubes.

2.3.1 Tubes : intervalles de trajectoires

Définition 10. *Un **tube** [12, 11] est un intervalle de trajectoires selon la relation d'ordre $\leq$ entre trajectoires suivante : $\mathbf{x}(\cdot) \leq \mathbf{y}(\cdot) \iff \forall t \in \mathbb{R}, \mathbf{x}(t) \leq \mathbf{y}(t)$. Ainsi, un tube $[\mathbf{x}](\cdot)$ est défini comme :*

$$[\mathbf{x}](\cdot) = [\mathbf{x}^-, \mathbf{x}^+](\cdot) = \left\{\mathbf{x}(\cdot) \in \mathcal{F}(\mathbb{R}, \mathbb{R}^n) \mid \forall t \in \mathbb{R},\ \mathbf{x}^-(t) \leq \mathbf{x}(t) \leq \mathbf{x}^+(t)\right\} \tag{2.37}$$

Dans la suite, on notera $\mathcal{F}^n$ l'ensemble des fonctions de $\mathbb{R}$ dans $\mathbb{R}^n$ et $\mathbb{I}\mathcal{F}^n$ l'ensemble des tubes de dimension n.

Remarque 8. *Les tubes sont aussi des fonctions de $\mathbb{R}$ dans $\mathbb{IR}^n$, et donc $\mathbb{I}\mathcal{F}^n = \mathcal{F}(\mathbb{R}, \mathbb{IR}^n)$.*

Implémentation des tubes par des tranches L'implémentation des tubes choisie dans cette thèse est celle proposée dans [13], voir la figure 2.5. Cette implémentation discrétise le tube comme une série de **tranches** (*slices*) qui contiennent la trajectoire de manière garantie. L'implémentation par tranches permet d'avoir un encadrement garanti puisque elle s'appuie sur l'arithmétique intervalle. Une autre manière d'envisager cette implémentation est de constater que cela revient à un tube $[\mathbf{x}^-(\cdot), \mathbf{x}^+(\cdot)]$ dans lequel les fonctions $\mathbf{x}^-(\cdot)$ et $\mathbf{x}^+(\cdot)$ sont des fonctions escaliers, numériquement représentables en machine.

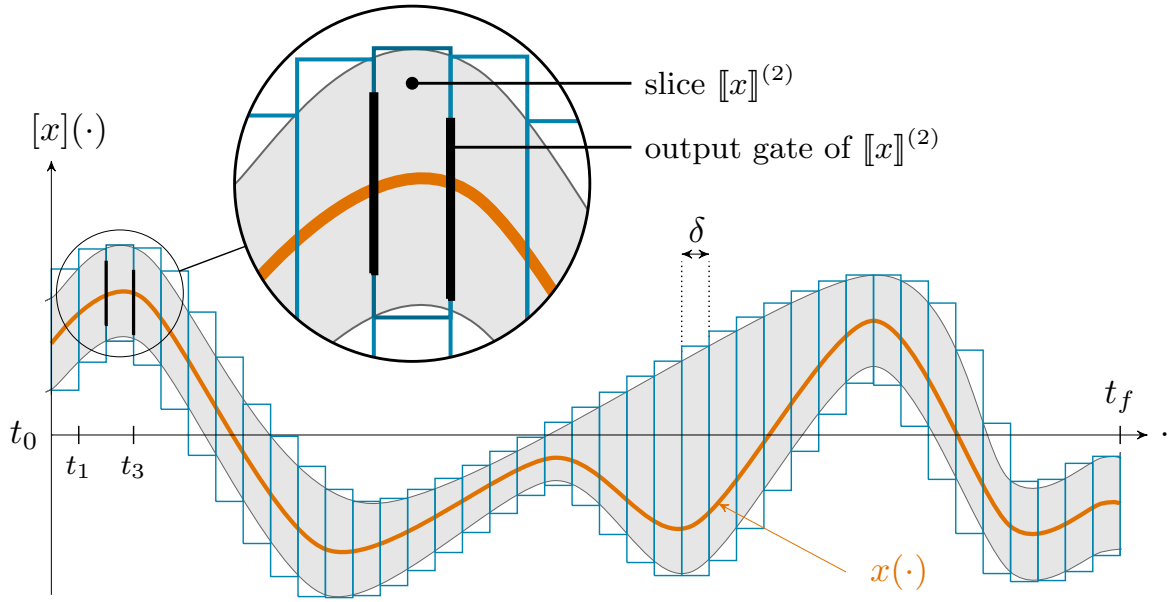


FIGURE 2.5 – Détail de l'implémentation d'un tube. Celui-ci est discrétisé en une suite de tranches, c'est-à-dire des boîtes qui bornent le tube sur un pas de temps δ. À l'interface des tranches sont situées des portes incluses dans les tranches et contenant la trajectoire aux instants correspondants. Les portes permettent d'améliorer les algorithmes de contraction décrits ci-dessous. La figure est extraite de [14], où l'exploitation des portes est explicitée.

2.3.2 Opérations sur tubes

Nous avons vu comment appliquer des opérations arithmétiques et ensemblistes sur des boîtes. Comme à un instant t, $[\mathbf{x}](t)$ est une boîte, ces mêmes opérations s'appliquent aux tubes [11, 13] en les appliquant à chaque instant t, ou à chaque tranche du tube dans son implémentation.

Nous pouvons alors définir les opérations sur tubes suivantes :

- l'inclusion : $[\mathbf{x}](\cdot) \subset [\mathbf{y}](\cdot)$ si pour tout $t \in \mathbb{R}$, $[\mathbf{x}](t) \subset [\mathbf{y}](t)$ ou autrement dit si $\mathbf{y}^-(\cdot) \leq \mathbf{x}^-(\cdot) \leq \mathbf{x}^+(\cdot) \leq \mathbf{y}^+(\cdot)$,
- le produit cartésien : $[\mathbf{x}](\cdot) \times [\mathbf{y}](\cdot)$ est le tube qui à t associe $[\mathbf{x}](t) \times [\mathbf{y}](t)$,
- l'intersection : $[\mathbf{x}](\cdot) \cap [\mathbf{y}](\cdot)$ est le tube qui à t associe $[\mathbf{x}](t) \cap [\mathbf{y}](t)$,
- l'arithmétique réelle pour une fonction arithmétique $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ générale : $\mathbf{f}([\mathbf{x}])(\cdot)$ est le plus petit tube pour l'inclusion contenant les valeurs possibles de $\mathbf{f}(\mathbf{x}(\cdot))$ pour $\mathbf{x}(\cdot) \in [\mathbf{x}](\cdot)$.

À ces opérations, nous pouvons envisager l'intégration entre deux bornes t_1 et t_2 [13] :

$$\int_{t_1}^{t_2} [\mathbf{x}](\tau)\mathrm{d}\tau = \left[\int_{t_1}^{t_2} \mathbf{x}^-(\tau)\mathrm{d}\tau, \int_{t_1}^{t_2} \mathbf{x}^+(\tau)\mathrm{d}\tau\right]. \tag{2.38}$$

Remarque 9. *Il n'est pas possible de dériver un tube : dès lors que pour $t \in \mathbb{R}$, $\mathbf{x}^-(t) \neq \mathbf{x}^+(t)$ alors parmi les trajectoires de $[\mathbf{x}](\cdot)$ à l'instant t, on trouve des trajectoires pouvant avoir n'importe quel taux de variation, ce qui rend la dérivée impossible à borner. Voir la preuve du Lemme 3 de [15] pour une démonstration formelle.*

2.3.3 Contraction de tubes

Tout comme nous avons défini des contracteurs de domaine intervalle pour des contraintes, il est possible d'en définir pour les tubes [11] :

Définition 11. *L'opérateur $\mathcal{C} : \mathbb{IF}^n \to \mathbb{IF}^n$ est un **contracteur de tube** associé à une contrainte $\mathcal{L}$ s'il vérifie deux propriétés :*

$$\begin{array}{ll} \forall [\mathbf{x}](\cdot) \in \mathbb{IF}^n, \mathcal{C}([\mathbf{x}](\cdot)) \subset [\mathbf{x}](\cdot) & \text{contractance} \\ (\mathbf{x}(\cdot) \in [\mathbf{x}](\cdot) \wedge \mathcal{L}(\mathbf{x}(\cdot))) \implies \mathbf{x}(\cdot) \in \mathcal{C}([\mathbf{x}](\cdot)) & \text{complétude} \end{array}$$

Dans la littérature, on trouve des contracteurs pour différents types de contraintes impliquant des tubes. Nous en présentons ici certains importants.

- **Contrainte d'inégalité entre tubes**
  C'est une contrainte du type $\mathbf{x}(\cdot) \leq \mathbf{y}(\cdot)$, pour laquelle il existe le contracteur $\mathcal{C}_{\leq}([\mathbf{x}](\cdot), [\mathbf{y}](\cdot))$ [11].
- **Contrainte de dérivée**
  C'est une contrainte du type $\dot{\mathbf{x}}(\cdot) = \mathbf{y}(\cdot)$, permettant de contracter un tube à partir de son tube dérivé. Nous utilisons le contracteur $\mathcal{C}_{\frac{d}{dt}}([\mathbf{x}](\cdot), [\mathbf{y}](\cdot))$ [13] dans l'ensemble de ce manuscrit pour traîter cette contrainte.
- **Contrainte à retard**
  C'est une contrainte du type $\mathbf{x}(t) = \mathbf{y}(t+\tau)$, où deux tubes sont égaux à un délai incertain près. Il existe deux versions d'un contracteur $\mathcal{C}_{\text{delay}}([\mathbf{x}](\cdot), [\mathbf{y}](\cdot), [\tau])$ [11, 16] associé à cette contrainte.
- **Contrainte d'évaluation d'un tube**
  C'est une contrainte du type $\mathbf{x}(\tau) = \mathbf{a}$, qui évalue un tube à un instant incertain, et est traitée par le contracteur $\mathcal{C}_{\text{eval}}([\mathbf{x}](\cdot), [\tau], [\mathbf{a}])$ [15].
- **Contrainte arithmétique entre tubes**
  C'est une contrainte du type $\mathbf{f}(\mathbf{x}(\cdot)) = \mathbf{y}(\cdot)$, qui s'applique à chaque instant du tube. Comme pour la contrainte arithmétique entre réels, il est possible d'utiliser un contracteur noté $\mathcal{C}_{\mathbf{f}}([\mathbf{x}](\cdot), [\mathbf{y}](\cdot))$ construit à l'aide de l'algorithme `HC4-Revise` (voir section 2.2.4.2 page 22). Ce contracteur est paramétré par la fonction arithmétique $\mathbf{f}$.

Comme pour les contractions sur boîtes, on retrouve ces contracteurs dans des catalogues de contracteurs, notamment la bibliothèque Codac [17].

2.4 Application robotique modélisée comme un CSP hybride

Considérons un problème de robotique pouvant être formalisé comme un CSP hybride, en utilisant des variables réelles et des variables trajectoires : ce problème d'estimation d'état est obtenu grâce à une association de données sur des points de repère (*landmarks*) indistinguables, problème traité dans [18].

Le lexique utilisé, bien que repris dans le chapitre 3 sur la robotique, est condensé ici : l'état $\mathbf{x}(\cdot)$ du robot contient toutes les variables nécessaires pour décrire son comportement, à savoir ici sa position et son orientation. Le point de départ des techniques d'estimation d'état consiste à mesurer les accélérations du robot afin de prédire l'évolution de son état, mais ce n'est pas suffisant en raison des dérives. Ce robot dispose de capteurs capables de percevoir des points de repères $\mathbf{m}^i$ de son environnement à différents instants $(t_i)_{i\in I}$, par exemple via un sonar, mais sans pouvoir les distinguer les uns des autres, et par la même occasion mesure leur position $\mathbf{y}^i$ relative au robot. La figure 2.6 montre un exemple de mesure de point de repère et illustre ces quantités.

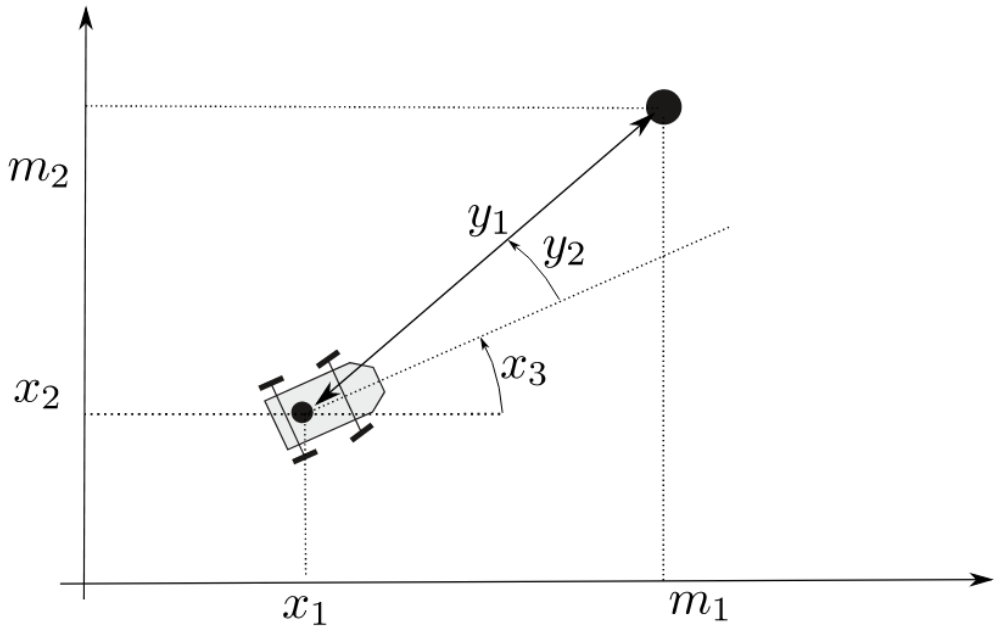


FIGURE 2.6 – Mesure $\mathbf{y}$, en coordonnées polaires, du point de repère $\mathbf{m}$ par le robot.

Enfin, le robot dispose d'une carte $\mathbb{M}$ des points de repère de la zone. L'association des données cherche ici à faire la correspondance entre les mesures et la carte, ce qui n'est pas un problème simple dans le cas de points de repère indistinguables.

$$\left\{\begin{array}{l} \textbf{Variables :}\ \mathbf{x}(\cdot),\ \mathbf{y}^i,\ \mathbf{m}^i \\ \textbf{Domaines :}\ [\mathbf{x}](\cdot),\ [\mathbf{y}^i],\ [\mathbf{m}^i] \\ \textbf{Contraintes :} \\ \quad 1.\quad \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\ \quad 2.\quad \mathbf{g}(\mathbf{x}(t_i), \mathbf{y}^i, \mathbf{m}^i) = \mathbf{0} \qquad \forall i \in I \\ \quad 3.\quad \mathbf{m}^i \in \mathbb{M} \qquad\qquad\quad \forall i \in I \end{array}\right. \tag{2.39}$$

Description des contraintes et contracteurs associés

Contrainte d'évolution. La contrainte 1. est la contrainte dite d'*évolution*, et n'implique que des variables trajectoires. Elle décrit l'évolution du robot en fonction de son état et de sa commande $\mathbf{u}(\cdot)$. Nous avons montré, en exemple 9 et plus particulièrement dans le système (2.36), la forme que peut prendre la fonction $\mathbf{f}$. Nous pouvons simplement décomposer cette contrainte en ajoutant une trajectoire $\mathbf{v}(\cdot)$ supplémentaire :

$$1.1.\quad \mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \qquad\qquad 1.2.\quad \dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$$

Nous avons maintenant une contrainte arithmétique et une contrainte différentielle, pour laquelle nous avons des contracteurs associés dans notre catalogue, respectivement $\mathcal{C}_{\mathbf{f}}$ et $\mathcal{C}_{\frac{d}{dt}}$.

Contraintes d'observation. Les contraintes 2. sont les contraintes d'*observation* déduites de la figure 2.6. Celles-ci portent sur des réels et une trajectoire, et peuvent être décomposées en ajoutant des variables auxiliaires réelles $\mathbf{a}^i$:

$$\begin{aligned} &2.1.\quad \mathbf{x}(t_i) = \mathbf{a}^i && \forall i \in I \\ &2.2.\quad \mathbf{g}(\mathbf{a}^i, \mathbf{y}^i, \mathbf{m}^i) = \mathbf{0} && \forall i \in I \end{aligned}$$

Ces contraintes sont traitées respectivement avec $\mathcal{C}_{\text{eval}}$ et un contracteur arithmétique sur réels $\mathcal{C}_{\mathbf{g}}$, obtenu par exemple avec l'algorithme `HC4-Revise`, voir section 2.2.4.2.

Contraintes d'association des données. Les contraintes 3. codent l'association des données dans le cadre de ce problème. Celles-ci ne correspondent pas aux contraintes présentes dans le catalogue, ni ne se décomposent plus simplement. Pour ce type de contraintes, il faut alors proposer un nouveau contracteur basé sur les intervalles. Dans cette application, la contrainte est traitée au moyen d'un nouveau contracteur, dit de constellation, appelé $\mathcal{C}_{\text{constell}}$. Il permet d'associer la position d'un point inconnu et encore incertain à une carte de *landmarks* donnée.

Résolution du CSP hybride

À chaque contrainte du problème, nous avons un contracteur associé. Pour résoudre le problème, le principe est de réduire la taille des domaines en appliquant les contracteurs, dans n'importe quel ordre, jusqu'à un point fixe. La liste de ces opérations est alors la suivante :

$$\left\{\begin{array}{ll} 1.1. & \mathcal{C}_{\mathbf{f}}([\mathbf{v}](\cdot), [\mathbf{x}](\cdot), [\mathbf{u}](\cdot)) \\ 1.2. & \mathcal{C}_{\frac{d}{dt}}([\mathbf{x}](\cdot), [\mathbf{v}](\cdot)) \\ 2.1. & \mathcal{C}_{\text{eval}}([t_i], [\mathbf{a}^i], [\mathbf{x}](\cdot)) \\ 2.2. & \mathcal{C}_{\mathbf{g}}([\mathbf{a}^i], [\mathbf{y}^i], [\mathbf{m}^i]) \\ 3. & \mathcal{C}_{\text{constell}}([\mathbf{m}^i]) \end{array}\right.$$

La méthode est testée en conditions réelles, avec un robot placé au départ à la position $(0, 0)^\intercal$, dont on ne connaît pas les coordonnées géographiques.

Dans une zone préalablement cartographiée, le robot perçoit différents *amers* avec des sonars latéraux au cours de sa mission. La figure 2.7 montre le résultat de la méthode sur ce problème, et comment il est possible d'estimer la trajectoire du robot même sans connaître sa position initiale. Rappelons que sans effectuer de mesures, le tube de la trajectoire devient de plus en plus large en raison de l'accumulation des incertitudes, et c'est pourquoi l'incertitude sur la trajectoire s'accroît en s'éloignant des instants des mesures, ce qui est notamment visible dans la zone située tout en bas à droite. Au contraire, à chaque mesure il est possible d'estimer les trajectoires cohérentes avec cette mesure, qui possède elle-même des incertitudes. En croisant plusieurs mesures, seules les trajectoires communes à toutes ces mesures sont gardées, ce qui diminue l'incertitude de la trajectoire. Cela est illustré sur la diagonale qui marque la fin de la trajectoire du robot, et où l'incertitude sur la trajectoire est pourtant très faible.

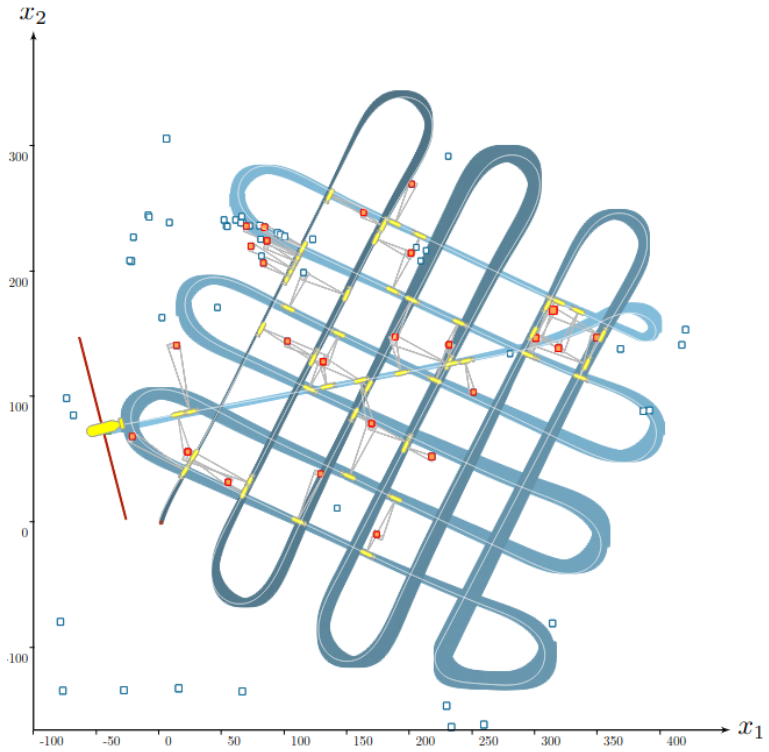


FIGURE 2.7 – Un robot se déplace selon une trajectoire $\mathbf{x}(\cdot)$ (trait blanc) quadrillant l'espace, où la position finale du robot est représentée par le robot le plus à gauche de la figure et les lignes rouges, représentant ici la portée des sonars. Les composantes de position du tube $[\mathbf{x}](\cdot)$ englobant cette trajectoire sont représentées en bleu. Dans son environnement sont situés des *amers*, qui sont des *landmarks* que le robot peut détecter. Lorsque celui-ci mesure un des amers et l'associe sans ambiguïté à un point de repère de la carte, cet amer est représenté par une boîte en rouge.

Discussion

Dans cette section, nous avons vu comment mettre en œuvre les outils de ce chapitre afin de créer des contracteurs susceptibles de répondre à un problème particulier de CSP hybride.

La méthode générale utilisée pour traiter ce problème revient à le décomposer jusqu'à obtenir des

contraintes pour lesquelles il est possible de construire un contracteur, la décomposition étant permise par l'algèbre des contracteurs. Cette approche sera notamment reprise dans notre première contribution, au chapitre 4.

Lorsque après décomposition, une ou plusieurs contraintes ne possèdent pas de contracteurs dans le catalogue de contracteurs, il devient nécessaire de l'enrichir. Dans l'application précédente, il n'existait pas de contracteur pour traiter la contrainte d'association des données, d'où la création du contracteur $\mathcal{C}_{\text{constell}}$. Parmi les contributions de cette thèse, nous construisons plusieurs nouveaux contracteurs :

- dans la première contribution, nous utilisons une nouvelle implémentation de domaine (voir chapitre 5) pour laquelle nous construisons de nouveaux contracteurs spécifiques, développés dans le chapitre 6,
- dans la seconde contribution en section 8.5 (page 112), nous décrivons une paire de contracteurs pour changer les coordonnées cartésiennes d'un sous-ensemble réel en coordonnées polaires.

Enfin, l'application présentée dans ce chapitre montre que la méthode fonctionne avec des données réelles, et permet ici d'estimer la trajectoire du robot à partir d'une position initiale complètement inconnue.

2.5 Ensembles épais comme domaines des variables ensemblistes

Nous avons désormais vu deux types d'intervalles : des intervalles de réels (*boîtes*) et des intervalles de trajectoires (*tubes*). Ces intervalles sont un ensemble compris entre deux bornes selon une relation d'ordre. Ce concept peut alors être étendu aux sous-ensembles réels, où la relation d'ordre privilégiée est l'inclusion $\subset$.

2.5.1 Ensembles épais : intervalles de sous-ensembles réels

Définition 12. *Un **intervalle d'ensembles**, ou **ensemble épais** (thick set [19, 20]), en dimension n est défini et noté de la manière suivante :*

$$[\mathbb{X}] = [\mathbb{X}^-, \mathbb{X}^+] = \left\{ \mathbb{X} \in \mathcal{P}(\mathbb{R}^n) \mid \mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+ \right\}. \tag{2.40}$$

L'ensemble des intervalles d'ensembles est noté $\mathbb{IP}(\mathbb{R}^n)$.

Visualisation et notions connexes aux intervalles d'ensembles

Nous en présentons ici deux notions différentes et proches des intervalles d'ensembles afin de mieux appréhender ces derniers.

Tripartition de l'espace. Naturellement, les intervalles d'ensembles partitionnent l'espace en trois :

- $\mathbb{X}^-$ désigne l'intérieur, à savoir une partie commune à tous les ensembles compris dans l'intervalle,
- $\overline{\mathbb{X}^+}$ désigne l'extérieur, toute une partie de l'espace qui n'appartient à aucun ensemble de l'intervalle,
- $\mathbb{X}^+ \setminus \mathbb{X}^-$ est la région frontière, ou incertaine.

Notamment, cette tripartition induit une représentation visuelle intuitive de ces ensembles à l'aide d'images tricolores, illustrée dans la figure 2.8.

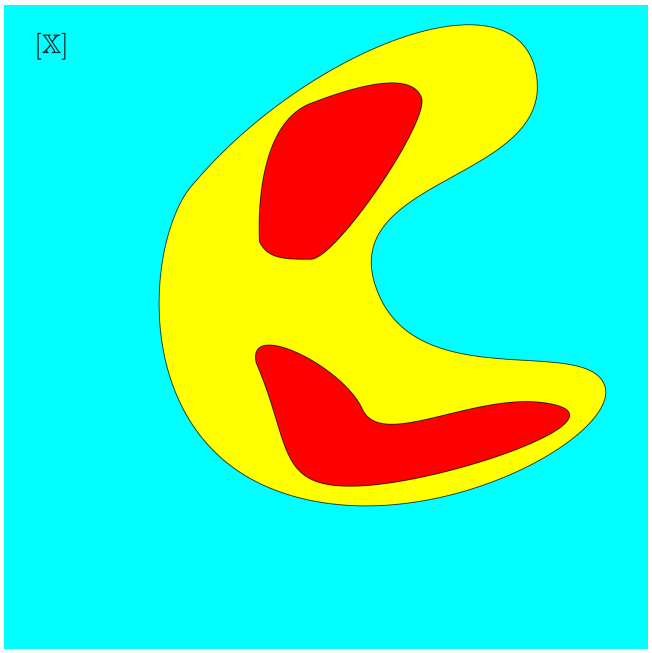


FIGURE 2.8 – Représentation visuelle d'un ensemble épais $[\mathbb{X}]$ comme une tripartition, avec en rouge l'intérieur $\mathbb{X}^-$, en bleu l'extérieur $\overline{\mathbb{X}^+}$ et en jaune l'ensemble frontière ou incertitude $\mathbb{X}^+ \setminus \mathbb{X}^-$.

Fuzzy sets. Les **ensembles flous**, ou **fuzzy sets** [21, 22, 23] forment une représentation ensembliste autorisant des incertitudes. Cette incertitude est représentée par une fonction caractéristique $m(\cdot)$, qui associe un degré d'appartenance dans l'intervalle $[0, 1]$ à chaque point de l'espace. En ce sens, il existe un lien entre les ensembles épais et les ensembles flous. En effet, pour un point $\mathbf{x}$:

— si $m(\mathbf{x}) = 0$ alors $\mathbf{x}$ est extérieur,
— si $m(\mathbf{x}) = 1$ alors $\mathbf{x}$ est intérieur,
— si $0 < m(\mathbf{x}) < 1$ alors $\mathbf{x}$ est frontière.

2.5.2 Opérations sur ensembles épais

Afin de faire des opérations sur intervalles d'ensembles, nous allons procéder de la même manière que pour les boîtes et tubes. En premier lieu, nous définissons l'opération d'enveloppe intervalle, mais cette fois sur des ensembles de sous-ensembles réels afin de les rendre intervalles.

Définition 13. *L'enveloppe intervalle pour un ensemble de sous-ensembles réels $\mathcal{X}$ est le plus petit intervalle d'ensembles contenant $\mathcal{X}$ au sens de l'inclusion. Il est noté $\square$ et se définit comme :*

$$\begin{aligned} \square : \mathcal{P}(\mathcal{P}(\mathbb{R}^n)) &\to \mathbb{I}\mathcal{P}(\mathbb{R}^n) \\ \mathcal{X} &\mapsto \left[\bigcap_{\mathbb{X}\in\mathcal{X}} \mathbb{X}, \bigcup_{\mathbb{X}\in\mathcal{X}} \mathbb{X}\right] \end{aligned} \tag{2.41}$$

En ce qui concerne les intervalles précédents, boîtes et tubes, nous avons pu définir deux types d'opérations :

— des opérations ensemblistes, qui s'appliquent sur ces intervalles considérés comme des ensembles,

— des opérations d'arithmétique réelle, qui cherchent à capturer dans un intervalle toutes les solutions possibles à l'opération. De manière informelle, une somme de boîtes revient à calculer la boîte de toutes les sommes possibles.

Ces deux types d'opérations possèdent certes des sémantiques différentes, mais ne posent pas d'ambiguïtés.

Cependant pour les intervalles d'ensembles, et comme indiqué dans [19], une ambiguïté se dessine : en effet, les opérations ensemblistes, telles que l'intersection, peuvent s'appliquer sur les intervalles d'ensembles, considérés comme des ensembles, mais peuvent aussi étendre les opérations ensemblistes de $\mathcal{P}(\mathbb{R}^n)$, comme pour les opérations arithmétiques précédentes étendues des réels à $\mathbb{IR}^n$.

Il faut donc pouvoir distinguer le fait de vouloir faire une intersection dans $\mathbb{I}\mathcal{P}(\mathbb{R}^n)$ entre intervalles, qui permet d'assurer la contraction de l'incertitude comme nous allons le voir dans la suite, du fait de vouloir étendre l'intersection de $\mathcal{P}(\mathbb{R}^n)$ entre deux ensembles à $\mathbb{I}\mathcal{P}(\mathbb{R}^n)$, afin de pouvoir calculer une enveloppe garantie à l'intersection de deux ensembles incertains.

Comme illustrées sur la figure 2.9 et telles que définies dans l'équation (9) de [20], ces deux opérations sont bien différentes :

— Notée $\sqcap$, il s'agit de l'intersection canonique sur $\mathbb{I}\mathcal{P}(\mathbb{R}^n)$, où le résultat est l'intervalle de tous les ensembles $\mathbb{Z}$ communs aux deux intervalles d'ensembles :

$$[\mathbb{X}] \sqcap [\mathbb{Y}] = \square \left(\{ \mathbb{Z} \in \mathcal{P}(\mathbb{R}^n) \mid \mathbb{Z} \in [\mathbb{X}] \wedge \mathbb{Z} \in [\mathbb{Y}] \} \right), \tag{2.42}$$

— Notée $\cap$, il s'agit de l'extension de l'intersection de l'intersection sur $\mathcal{P}(\mathbb{R}^n)$ à $\mathbb{I}\mathcal{P}(\mathbb{R}^n)$, où l'on cherche à encapsuler dans un intervalle tous les résultats possibles :

$$[\mathbb{X}] \cap [\mathbb{Y}] = \square \left(\{ \mathbb{X} \cap \mathbb{Y} \mid \mathbb{X} \in [\mathbb{X}] \wedge \mathbb{Y} \in [\mathbb{Y}] \} \right). \tag{2.43}$$

Remarque 10. *Dans ces définitions, l'opérateur d'enveloppe $\square$ est utilisé afin de passer de $\mathcal{P}(\mathcal{P}(\mathbb{R}^n))$ à $\mathbb{I}\mathcal{P}(\mathbb{R}^n)$, autorisant la stabilité des opérations.*

En particulier, nous avons les propriétés suivantes :

$$\mathbb{X} \in [\mathbb{X}], \mathbb{Y} \in [\mathbb{Y}] \implies \mathbb{X} \cap \mathbb{Y} \in ([\mathbb{X}] \cap [\mathbb{Y}]), \tag{2.44}$$

$$\mathbb{Z} \in [\mathbb{X}], \mathbb{Z} \in [\mathbb{Y}] \implies \mathbb{Z} \in ([\mathbb{X}] \sqcap [\mathbb{Y}]). \tag{2.45}$$

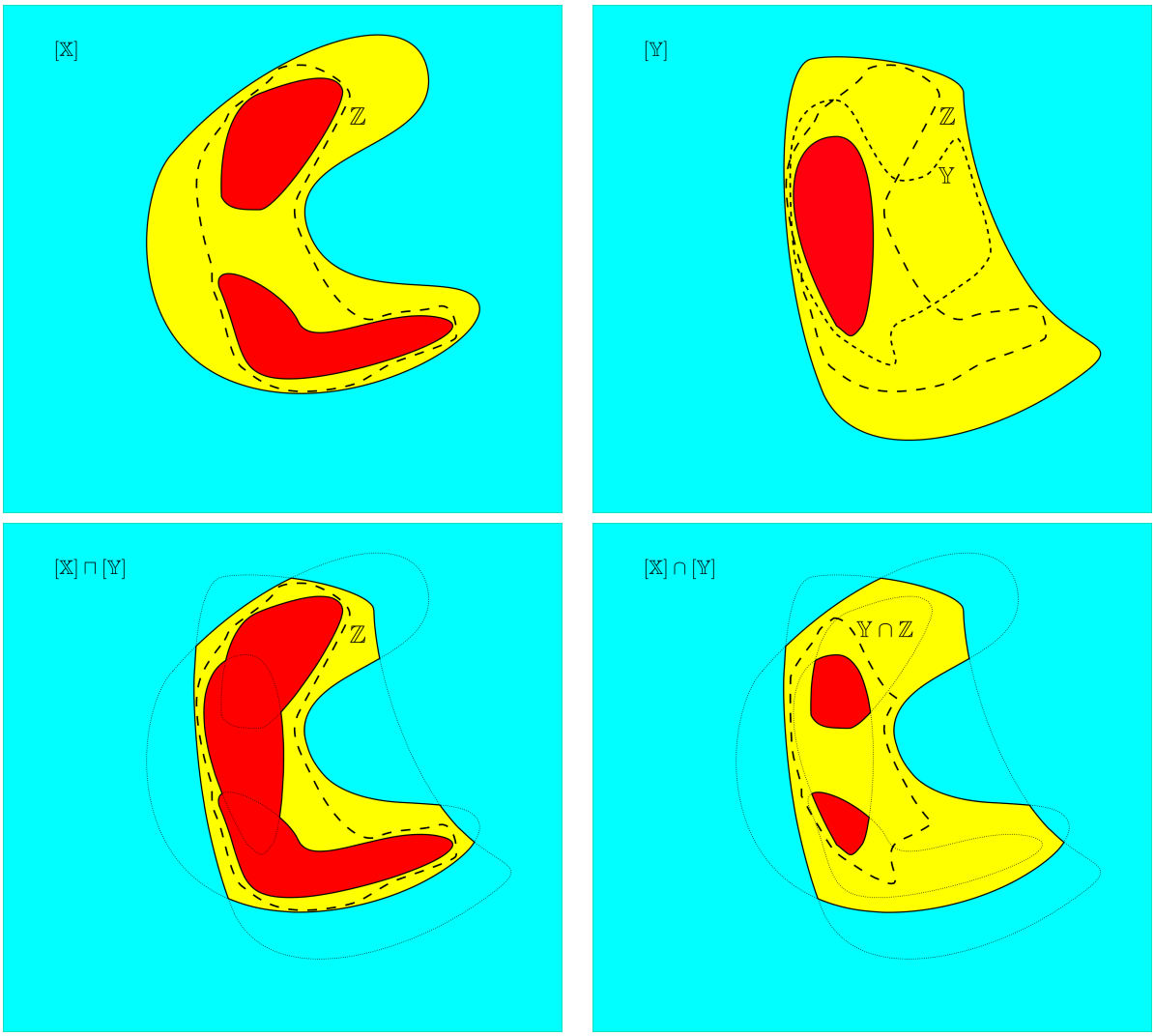


FIGURE 2.9 – Différence entre les deux opérations d'intersection : les ensembles épais $[\mathbb{X}]$ et $[\mathbb{Y}]$ en entrée sont situés en haut. L'ensemble $\mathbb{Z}$, commun aux deux ensembles, se retrouve aussi en pointillés dans $[\mathbb{X}] \sqcap [\mathbb{Y}]$ (en bas à gauche) : les parties intérieure (rouge) et extérieure (bleue) sont cohérentes avec $\mathbb{Z}$. L'ensemble $\mathbb{Z} \cap \mathbb{Y}$, avec $\mathbb{Y} \in [\mathbb{Y}]$, se retrouve bien dans $[\mathbb{X}] \cap [\mathbb{Y}]$ en pointillés (en bas à droite). En bas à droite, on a fait l'intersection $\cap$, à savoir l'intervalle de toutes les intersections, qui contient en particulier l'ensemble $\mathbb{Y} \cap \mathbb{Z}$.

Dans la suite, deux opérations sur $\mathbb{IP}(\mathbb{R}^n)$ nous intéressent :

— l'intersection notée $\sqcap$ comme dans la littérature,

— l'inclusion notée $\sqsubset$.

Les notations classiques seront réservées à l'extension des opérations de $\mathcal{P}(\mathbb{R}^n)$ à $\mathbb{IP}(\mathbb{R}^n)$.

Les autres opérations seront donc les opérations de l'algèbre des sous-ensembles réels détaillée dans la section 2.2.2.1, ainsi que les opérations d'arithmétique réelle, et sont étendues aux intervalles d'ensembles.

La plupart de ces opérations peuvent être facilement calculées avec les opérations algébriques classiques

sur les bornes. Ainsi, en utilisant ces simplifications lorsque possible, nous pouvons définir et calculer les opérations suivantes :

— l'intersection $\sqcap$:

$$[\mathbb{X}] \sqcap [\mathbb{Y}] = [\mathbb{X}^- \cup \mathbb{Y}^-, \mathbb{X}^+ \cap \mathbb{Y}^+]. \tag{2.46}$$

— l'intersection $\cap$:

$$[\mathbb{X}] \cap [\mathbb{Y}] = [\mathbb{X}^- \cap \mathbb{Y}^-, \mathbb{X}^+ \cap \mathbb{Y}^+], \tag{2.47}$$

— l'union :

$$[\mathbb{X}] \cup [\mathbb{Y}] = [\mathbb{X}^- \cup \mathbb{Y}^-, \mathbb{X}^+ \cup \mathbb{Y}^+], \tag{2.48}$$

— le produit cartésien :

$$[\mathbb{X}] \times [\mathbb{Y}] = [\mathbb{X}^- \times \mathbb{Y}^-, \mathbb{X}^+ \times \mathbb{Y}^+], \tag{2.49}$$

— la négation :

$$\neg[\mathbb{X}] = [\overline{\mathbb{X}^+}, \overline{\mathbb{X}^-}], \tag{2.50}$$

— la projection sur un sous-ensemble I de dimensions :

$$\mathrm{proj}_I[\mathbb{X}] = \square\left(\{\mathrm{proj}_I \mathbb{X} \mid \mathbb{X} \in [\mathbb{X}]\}\right), \tag{2.51}$$

— pour une fonction arithmétique $\mathbf{f} : \mathbb{X} \to \mathbb{Y}$, on définit

$$\mathbf{f}([\mathbb{Z}]) = \square\left(\{\mathbf{f}(\mathbb{Z}) \mid \mathbb{Z} \in [\mathbb{Z}]\}\right). \tag{2.52}$$

Nous définissons aussi deux nouvelles opérations spécifiques aux ensembles épais :

— l'intersection des bornes supérieures $\cap^+$:

$$[\mathbb{X}] \cap^+ [\mathbb{Y}] = [\mathbb{X}^-, \mathbb{X}^+ \cap \mathbb{Y}^+], \tag{2.53}$$

— l'union des bornes inférieures $\cup^-$:

$$[\mathbb{X}] \cup^- [\mathbb{Y}] = [\mathbb{X}^- \cup \mathbb{Y}^-, \mathbb{X}^+]. \tag{2.54}$$

Remarque 11. *Ces deux opérations sont importantes puisqu'elles sont cohérentes avec l'inclusion. Si* $\mathbb{X} \in [\mathbb{X}]$, $\mathbb{Y} \in [\mathbb{Y}]$ *et* $\mathbb{X} \subset \mathbb{Y}$ *alors :*

$$\mathbb{X} \in [\mathbb{X}] \cap^+ [\mathbb{Y}], \tag{2.55}$$
$$\mathbb{Y} \in [\mathbb{Y}] \cup^- [\mathbb{X}]. \tag{2.56}$$

Ces équations permettent donc de définir directement un contracteur pour une contrainte d'inclusion.

Enfin, la relation d'ordre sur $\mathbb{IP}(\mathbb{R}^n)$ que nous allons considérer dans ce manuscrit est l'inclusion $\sqsubset$ définie par :

$$[\mathbb{X}] \sqsubset [\mathbb{Y}] \iff (\mathbb{Y}^- \subset \mathbb{X}^-) \wedge (\overline{\mathbb{Y}^+} \subset \overline{\mathbb{X}^+}). \tag{2.57}$$

Remarque 12. *Le calcul de l'équation* (2.52) *peut s'appuyer sur les outils vus précédemment, à savoir l'arithmétique intervalle ainsi que les différentes fonctions d'inclusion abordées dans la section 2.2.3 (page 17).*

2.5.3 Contraction d'ensembles épais

Comme pour les boîtes et tubes, nous pouvons définir un contracteur sur des intervalles d'ensembles :

Définition 14. *L'opérateur $\mathcal{C}\colon \mathbb{IP}(\mathbb{R}^n) \to \mathbb{IP}(\mathbb{R}^n)$ est un **contracteur d'ensemble épais** associé à une contrainte $\mathcal{L}$ s'il vérifie deux propriétés :*

$$\begin{array}{ll} \forall [\mathbb{X}] \in \mathbb{IP}(\mathbb{R}^n),\, \mathcal{C}([\mathbb{X}]) \sqsubset [\mathbb{X}] & \text{contractance} \\ (\mathbb{X} \in [\mathbb{X}] \wedge \mathcal{L}(\mathbb{X})) \implies \mathbb{X} \in \mathcal{C}([\mathbb{X}]) & \text{complétude} \end{array}$$

Remarque 13. *L'opération $\sqcap$, tout comme l'opération d'intersection entre boîtes ou tubes, permet d'obtenir la contractance.*

Avec les opérations décrites ci-dessus, il devient possible de construire des contracteurs adaptés aux contraintes à partir de leurs définitions, comme nous l'avons illustré dans l'application robotique d'un problème CSP hybride, en section 2.4 (page 27).

Notamment, les contributions de cette thèse iront jusqu'à donner l'implémentation de tels contracteurs, qui peut différer selon l'implémentation des ensembles épais.

2.5.4 Exemple de construction d'un contracteur sur variables ensemblistes

Nous allons montrer comment il est possible de se servir des outils présentés dans cette section pour construire un contracteur à partir d'une contrainte impliquant des variables ensemblistes.

Prenons la contrainte suivante :

$$\left\{ \begin{array}{l} \textbf{Variables :}\ \mathbb{X},\, \mathbb{Y},\, \mathbb{Z} \\ \textbf{Domaines :}\ [\mathbb{X}],\, [\mathbb{Y}],\, [\mathbb{Z}] \\ \textbf{Contrainte :}\ \mathbb{Z} \subset \mathbb{X} \cap \mathbb{Y} \end{array} \right. \tag{2.58}$$

Nous pouvons alors en déduire certaines relations qui viennent contraindre les domaines des variables. En particulier, nous pouvons en déduire que :

$$\begin{array}{l} \quad (\mathbb{Z} \subset \mathbb{X} \cap \mathbb{Y}) \wedge (\mathbb{X} \in [\mathbb{X}]) \wedge (\mathbb{Y} \in [\mathbb{Y}]) \\ \implies \exists \mathbb{Z}' \in [\mathbb{X}] \cap [\mathbb{Y}],\, \mathbb{Z} \subset \mathbb{Z}' \\ \implies \mathbb{Z} \subset \mathbb{X}^+ \cap \mathbb{Y}^+ \\ \implies \mathbb{Z} \in [\emptyset, \mathbb{X}^+ \cap \mathbb{Y}^+]. \end{array} \tag{2.59}$$

D'autre part, nous avons aussi :

$$\begin{array}{l} \quad (\mathbb{Z} \subset \mathbb{X} \cap \mathbb{Y}) \wedge (\mathbb{Z} \in [\mathbb{Z}]) \\ \implies \mathbb{Z}^- \subset \mathbb{X} \cap \mathbb{Y} \\ \implies (\mathbb{X} \in [\mathbb{Z}^-, \mathbb{R}^n]) \wedge (\mathbb{Y} \in [\mathbb{Z}^-, \mathbb{R}^n]). \end{array} \tag{2.60}$$

Ainsi, nous pouvons construire un contracteur pour cette contrainte :

$$\mathcal{C}\begin{pmatrix}[\mathbb{X}]\\[\mathbb{Y}]\\[\mathbb{Z}]\end{pmatrix} = \begin{pmatrix}[\mathbb{X}] \sqcap [\mathbb{Z}^-, \mathbb{R}^n]\\[\mathbb{Y}] \sqcap [\mathbb{Z}^-, \mathbb{R}^n]\\[\mathbb{Z}] \sqcap [\emptyset, \mathbb{X}^+ \cap \mathbb{Y}^+]\end{pmatrix} \tag{2.61}$$

L'opérateur $\sqcap$ garantit la contractance de ce contracteur, et les relations (2.59) et (2.60) garantissent la complétude de ce contracteur.

Dans ce chapitre, nous avons introduit le paradigme de CSP et plus particulièrement le CSP hybride comprenant des variables réelles et trajectoires, avec une application à un problème de robotique.

Nous avons aussi montré comment traiter les variables de sous-ensembles de réels, même si l'implémentation des ensembles épais n'a pas encore été discutée. Une implémentation possible d'ensemble épais, le **pavage épais régulier**, sera utilisée pour la première contribution de cette thèse, au chapitre 5.

Chapitre 3

Approches probabilistes et ensemblistes du problème de SLAM

Plan du chapitre

Nous avons abordé dans l'introduction le problème de SLAM, ou problème de localisation et cartographie simultanées, et notamment l'un des objectifs de ce manuscrit qui est de pouvoir le résoudre de manière ensembliste.

Nous commencerons par présenter les différents moyens disponibles pour la localisation dans un environnement connu, désigné habituellement par le nom d'**estimation d'état**, avant de présenter le problème de SLAM.

3.1 Estimation d'état en robotique

3.1.1 Capteurs et grandeurs mesurées

Le problème d'estimation d'état est une problématique vaste en robotique, notamment dû à la grande variété de technologies de capteurs permettant d'acquérir des données sur la localisation ou l'environnement, que nous répartissons ici en deux catégories : *extéroceptifs* et *proprioceptifs*.

Capteurs extéroceptifs. Ces capteurs perçoivent l'environnement extérieur au robot. Ils mesurent des grandeurs de l'environnement selon différentes modalités.

— Certains capteurs extéroceptifs reposent sur une source externe, que ce soit en terme de signaux, infrastructure ou autre. Le système le plus commun pour la localisation est le **système de positionnement par satellite** (*Global Navigation Satellite System* ou **GNSS**). Il s'aide de constellation de satellites pour donner des informations sur la position de manière absolue, et permet de faciliter grandement la partie localisation du SLAM. Dans les contextes où le GNSS est disponible et suffisant, ce problème est alors réduit à de la cartographie.

— Les caméras permettent d'obtenir des données riches et exploitables par les méthodes issues du domaine de la vision par ordinateur (*Computer Vision*). En particulier, on appelle *SLAM visuel* les formulations de SLAM exploitant des caméras. Pour les algorithmes permettant d'extraire des données utilisables à partir des images, nous pouvons citer *Scale-Invariant Feature Transform* (**SIFT**) [24], *Speeded up robust features* (**SURF**) [25], ou encore **ORB** [26], notamment utilisée dans l'algorithme de SLAM **ORB-SLAM2** [27].

— Les capteurs les plus classiques pour percevoir l'environnement sont les capteurs *range and bearing*, mesurant des distances à différents angles, tels que les LiDARs et sonars évoqués en introduction. Ces types de données donnent lieu aux méthodes classiques de SLAM, avec des mesures se représentant sous la forme de nuages de points. Ce type de capteur étant le plus classique et disponible dans des cas d'application plus larges, nous allons considérer dans la suite les méthodes de résolution de *range-based SLAM* basées sur ce type de technologie.

Capteurs proprioceptifs. Les capteurs proprioceptifs, au contraire des capteurs extéroceptifs, mesurent des grandeurs internes au robot. De manière embarquée, la centrale à inertie (*Inertial Measurement Unit* ou **IMU**), possède des capteurs proprioceptifs mesurant les accélérations ainsi que la vitesse angulaire, au moyen d'accéléromètres et gyroscopes. Pour un robot à roues, un encodage des roues permet de mesurer les déplacements relatifs des roues, et l'angle de direction donné aux roues peut être enregistré.

Certaines technologies de capteurs sont limitées dans leur disponibilité, par leur coût, poids, dépendance extérieure, ou indisponibilité dans certains milieux. En particulier, les technologies de type GNSS ne sont pas fiables dans des environnements intérieurs, sous-marins, sous-terrains ou encore extraterrestres. Ainsi dans ce chapitre, ces technologies ne seront pas abordées. Il faut cependant noter que les techniques développées dans ce manuscrit n'invalident pas l'utilisation de technologies différentes, qui peuvent être exploitées de manière complémentaire.

3.1.2 Techniques d'estimation d'état

Afin de bien formuler les techniques de résolution du problème d'estimation d'état, nous introduisons ici les notations qui seront utilisées dans la suite. Elles s'appuient sur les notations déjà évoquées en introduction de ce manuscrit. Soient :

- $\mathbf{x}(t)$ l'état du robot au cours du temps, aussi appelé trajectoire.
- $\mathbf{u}(t)$ la commande du robot au cours du temps.
- $\mathbf{z}_k$ le vecteur de mesures à l'instant t_k, représentant la perception de l'environnement par les capteurs du robot, pour k allant de 0 à $n-1$.

En particulier, nous allons décrire deux techniques de localisation classiques, décrites par deux **équations d'état** :

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.1}$$

$$\mathbf{z_k} = \mathbf{g}(\mathbf{x}(t_k), \mathbb{M}). \tag{3.2}$$

D'une part, la fonction d'évolution $\mathbf{f}(\cdot)$ modélise la physique du robot et permet de connaître sa dynamique à partir de son état actuel et de sa commande. D'autre part, la fonction d'observation $\mathbf{g}(\cdot)$ mentionnée en introduction modélise le comportement des capteurs du robot, c'est-à-dire détermine quelle sera la mesure effectuée par le capteur, étant donné sa position et l'environnement.

Étude de la dynamique du robot

La première manière de se localiser est d'utiliser la connaissance des accélérations et vitesses du robot. En les intégrant successivement, il est alors possible d'obtenir l'état du robot $\mathbf{x}(\cdot)$. Ces accélérations et vitesses sont alors toutes inclues dans les composantes de $\dot{\mathbf{x}}(\cdot)$, dérivée de $\mathbf{x}(\cdot)$.

Il y a deux manières d'acquérir ces accélérations et vitesses :

- soit directement mesurées par les capteurs proprioceptifs,
- soit déduites de notre connaissance de la physique du robot, modélisée par la fonction d'évolution $\mathbf{f}(\cdot)$.

Il est alors possible d'utiliser la mesure ou le calcul de $\dot{\mathbf{x}}(\cdot)$ afin de proposer une prédiction $\hat{\mathbf{x}}(t_k)$ de l'état courant en fonction de l'état précédent et de la commande actuelle :

$$\hat{\mathbf{x}}(t_k) = \mathbf{x}(t_{k-1}) + (t_k - t_{k-1})\,\dot{\mathbf{x}}(t_k). \tag{3.3}$$

Remarque 14. *Ce problème de localisation relative [28] est connu sous le nom de navigation à l'estime (dead reckoning). En particulier on peut distinguer l'odométrie, qui se base sur l'encodage et l'orientation des roues, de la navigation inertielle basée sur les gyroscopes et accéléromètres.*

Ce modèle de localisation est connu sous le nom de **modèle de mouvement** (*motion model*).

Cependant avec cette méthode de localisation, les dérives surviennent rapidement : que ce soit à cause de bruit sur les mesures ou un patinage des roues, les erreurs s'accumulent et cette estimation de l'état du robot devient inutilisable au bout d'un moment. Il est alors nécessaire de la coupler avec d'autres méthodes de localisation, issues la perception de l'environnement direct du robot. Ce nouveau modèle est détaillé dans la section suivante.

Étude des observations de l'environnement

Une manière naturelle de se repérer dans un environnement connu est de pouvoir associer une seule mesure à la carte de l'environnement. En effet, le comportement du capteur est modélisé par une équation d'observation, qui retourne un jeu de mesures distances-angles en fonction de la carte et de la position du robot dans celle-ci :

$$\mathbf{z_k} = \mathbf{g}(\mathbf{x}(t_k), \mathbb{M}). \tag{3.4}$$

Dans le cas de capteurs de type LiDAR, nous connaissons $\mathbf{g}$: elle mesure la distance du plus proche obstacle dans plusieurs cônes prédéterminés. Si de plus nous connsaissons les points de $\mathbb{M}$ qui ont déclenché la mesure $\mathbf{z_k}$, nous pouvons en déduire la pose du robot dans $\mathbf{x}(t_k)$.

Ce problème mobilise plusieurs techniques, allant du recalage à de la reconnaissance de caractéristiques (*features*) dans les données. On parle alors d'**association de données** ou *data association [29].*

Effectuer les bonnes correspondances permet d'en déduire une transformation, c'est-à-dire une information de localisation relative à l'environnement, permettant d'affiner la prédictions du modèle de mouvement.

Ce modèle est appelé **modèle d'observation**.

3.2 Problème de localisation et cartographie simultanées

Nous avons vu différents modèles afin de pouvoir localiser un robot dans un environnement connu. Cependant, ce problème devient plus compliqué lorsque l'environnement est inconnu : il devient alors nécessaire de construire une carte de l'environnement au fur et à mesure, afin de se localiser dans celle-ci.

Il s'agit alors de traiter le modèle d'observation avec $\mathbb{M}$ comme variable.

La carte n'étant pas complète, voire complètement inconnue, le problème d'association des données devient plus compliqué. Cependant, il reste possible d'exploiter différentes problématiques qui ne nécessitent pas de connaître la carte de l'environnement :

- en général entre deux instants proches dans le temps, par exemple t_{k-1} et t_k, les jeux de mesures se superposent en grande partie. Un alignement entre ces deux jeux de mesures permet alors, la plupart du temps, d'estimer correctement la dérive entre ces deux instants, ce qui vient notamment renforcer l'estimation $\hat{\mathbf{x}}(t_k)$ précédente. On parle alors de **scan matching** [30].
- entre deux instants éloignés, lorsque le robot a effectué une boucle complète : on se retrouve dans un cas similaire au *scan matching*, et trouver l'alignement permet de corriger la dérive de manière bien pus efficace. Il reste alors à savoir quand il est pertinent d'essayer de faire cet alignement, et de le faire avec plus d'incertitudes entre les positions que le *scan matching*. On parle alors de **fermeture de boucle**, ou *loop closure*, un problème compliqué et abordé dans différentes approches [31, 32].

De plus, le modèle d'observation permet de rassembler les mesures dans un référentiel commun, et donc d'ajouter tous les jeux de mesure en une carte globale, ou autrement dit permet de cartographier l'environnement.

3.2.1 Formulation du problème de SLAM sous deux formalismes

Jusqu'ici, nous avons présenté deux composantes fondamentales du SLAM :

— le **modèle de mouvement**, qui prédit l'évolution de la position du robot,
— le **modèle d'observation**, qui cherche à rendre la position et la connaissance de l'environnement cohérentes avec les mesures de l'environnement.

Les méthodes de résolution du problème de SLAM reposent donc sur l'utilisation de ces modèles et des concepts qu'ils soulèvent.

Cependant, les hypothèses du problème font apparaître des incertitudes à plusieurs niveaux, ce qui le rend difficile en pratique. En effet, les incertitudes proviennent à la fois des mesures bruitées des différents capteurs, proprioceptifs ou extéroceptifs, ainsi que des résultats des problèmes de recalage ou d'association des données, qui mobilisent généralement des méthodes d'optimisation.

Afin de prendre en compte ces incertitudes, il est possible d'énoncer le problème de SLAM sous différents formalismes :

— dans un formalisme probabiliste, où les incertitudes sont modélisées sous la forme de distribution de probabilité,
— dans un formalisme ensembliste, qui encadre les incertitudes par des intervalles, puis raisonne sur des ensembles d'états compatibles.

Problème de SLAM dans un cadre probabiliste

Dans ce formalisme, toutes les variables du problème seront considérées comme des distributions de probabilités. Le problème se résume à estimer la probabilité de la trajectoire du robot $\mathbf{x}(\cdot)$ et de l'environnement $\mathbb{M}$ conditionnée par les différentes mesures des capteurs proprioceptifs, la commande $\mathbf{u}(\cdot)$, les mesures de l'environnement $(\mathbf{z_k})_{0\leq k\leq n-1}$, et sa position initiale $\mathbf{x}(t_0)$.

Autrement dit, le problème de SLAM probabiliste [33] cherche à calculer, pour chaque instant d'observation t_k, les distributions de probabilité conditionnelles suivantes :

$$P(\mathbf{x}(t_k), \mathbb{M} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}(t_0)), \tag{3.5}$$

Les grandeurs $\mathbf{Z}_{0:k}$ et $\mathbf{U}_{0:k}$ désignent tout l'historique des $\mathbf{z}_i, \mathbf{u}(t_i)$ pour i de 0 à k.

Pour rappel, pour deux variables aléatoires $\mathbf{X}$ et $\mathbf{Y}$, la distribution de probabilité conditionnelle est une fonction $P(\mathbf{x} \mid \mathbf{y})$, qui désigne la probabilité que $\mathbf{X} = \mathbf{x}$, sachant que $\mathbf{Y} = \mathbf{y}$. Dans notre cas, la distribution (3.5) désigne alors notre connaissance de la trajectoire et de la carte selon la pose initiale, l'historique des commandes appliquées et les mesures effectuées.

Dans ce formalisme probabiliste, les modèles de mouvement et d'observation donnent respectivement deux séries de probabilités :

$$P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k), \tag{3.6}$$
$$P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbb{M}). \tag{3.7}$$

Nous verrons dans la section suivante comment les méthodes classiques du problème de SLAM exploitent ces probabilités pour calculer la probabilité (3.5).

Problème de SLAM sous la forme d'un CSP

Dans ce formalisme ensembliste, les variables du problème sont comprises dans un domaine intervalle, tel que décrit dans le chapitre précédent, section 2.2.

Nous rappelons que, pour les techniques d'estimaton d'état, nous avons la donnée d'une **fonction d'évolution f** et d'une **fonction d'observation g**, impliquées respectivement dans deux **équations d'état** : l'équation d'évolution (3.1) et l'équation d'observation (3.2).

La modélisation CSP du problème de SLAM découle de ces équations d'état :

$$\begin{cases} \textbf{Variables :} \ \mathbb{M},\ \mathbf{x}(\cdot) \\ \textbf{Domaines :} \ [\mathbb{M}],\ [\mathbf{x}](\cdot) \\ \textbf{Contraintes :} \\ \quad 1. \quad \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\ \quad 2. \quad \mathbf{x}(t_0) = \mathbf{0} \\ \quad 3. \quad \mathbf{z}_k = \mathbf{g}(\mathbf{x}(k), \mathbb{M}) \qquad \forall k \in \{0, 1, \ldots, n-1\} \end{cases} \tag{3.8}$$

3.2.2 Variantes de SLAM

Le problème de SLAM ayant un champ d'application assez large, nous évoquons ici deux particularités du problème qui peuvent orienter le choix d'une méthode appropriée au travers de deux questions :

— voulons nous calculer juste la pose courante ou bien calculer aussi les poses passées ?

— comment représenter la carte ?

SLAM online* ou *Full SLAM On désigne par *Online SLAM* un algorithme qui ne cherche qu'à estimer la pose actuelle du robot et la carte. On désigne au contraire par *Full SLAM*, appelé aussi parfois *Offline SLAM*, un algorithme qui cherche à estimer l'ensemble de toutes les poses ainsi que la carte. Il est utilisé en général une fois la mission du robot terminée. Ce dernier n'exclut cependant pas l'utilisation en temps réel, et fournit de meilleurs résultats au prix d'un plus grand coût numérique.

Représentations de la carte Selon les hypothèses de départ, la manière de modéliser l'environnement et de représenter la carte donnent lieu à des approches différentes. Il est possible d'en distinguer plusieurs :

1. les approches *feature-based* et *landmark-based*, basées sur l'extraction de primitives géométriques ou visuelles (*features*) et autres points d'intérêt distinguables ou indistinguables de l'environnement (*landmarks*), donnent une approximation de l'environnement selon ces sous-ensembles,
2. l'approche *grid-based*, basée sur les grilles d'occupation [34], découpent l'espace en cellules en leur attribuant chacune une probabilité de présence d'obstacle tout en restant cohérent avec la physique des capteurs [35]. Ces cellules peuvent alors êtres mises à jour par des procédures d'estimation bayésiennes.

3.2.3 Différentes représentations numériques des ensembles dans la littérature

Les différentes représentations utilisées dans la littérature pour les cartes dans le SLAM sont étudiées ici et illustrées dans la figure 3.1.

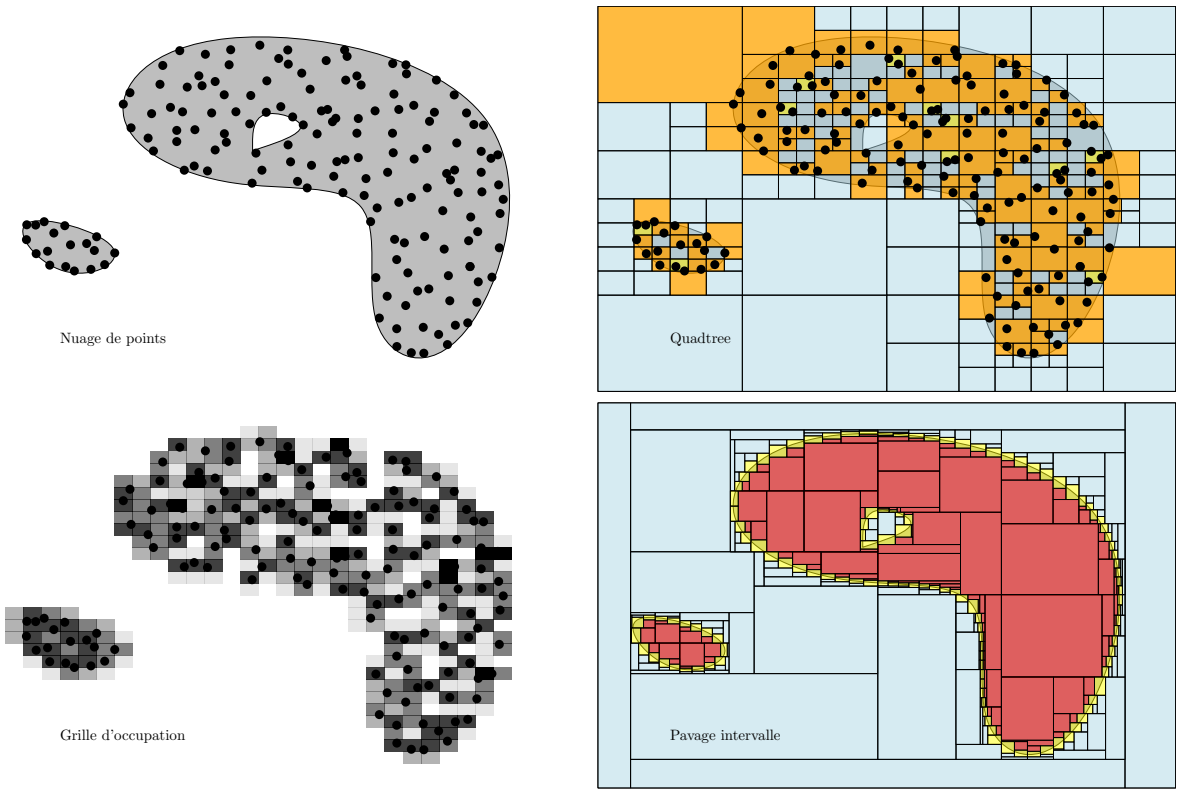


FIGURE 3.1 – Ensemble à représenter et différentes représentations possibles. Le quadtree et la grille d'occupation sont déduits du nuage de points. Pour le quadtree, le résultat présenté est celui d'un arbre quaternaire de profondeur 5. Les couleurs des cellules sont à titre indicatif, avec des cellules bleues vides et des cellules oranges avec un seul point du nuage : les cellules jaunes ont plusieurs points, et seront les seules à être découpées en quatre s'il faut aller plus en profondeur. Pour la grille d'occupation, le niveau de gris indique la probabilité de présence d'obstacle selon le nuage de points. Pour le pavage intervalle, le résultat est celui obtenu après 7 séries de contractions/bissections. Les cellules bleues sont garanties de ne pas intersecter l'ensemble, et les rouges sont garanties d'être incluses dans l'ensemble. Les jaunes seront les seules à être bissectées et contractées s'il est décidé d'aller plus en profondeur.

Nuages de points

Un *nuage de points* est un ensemble de points isolés, qui peut approcher l'ensemble à représenter. Numériquement, les nuages de points sont codés comme des matrices, avec leurs coordonnées et au besoin certains attributs supplémentaires, par exemple une couleur. Sa structure matricielle lui permet de bénéficier d'une multitude d'outils d'analyse bien établis et rapides, et c'est aussi la manière la plus simple d'interpréter des données provenant de certains capteurs, en particulier le LiDAR. Ceci en fait une représentation très utilisée en pratique. Nous pouvons aussi noter que les nuages de points peuvent aussi être issus d'autres méthodes de collecte de données, comme des sondages en analyse

statistique, comme nous le verrons au chapitre 7. Selon sa précision, densité et répartition, un nuage de points peut être une bonne approximation de l'ensemble. En général, cette représentation consiste en un échantillonnage de l'ensemble à représenter comme illustré dans la figure 3.1.

Grilles de pixels

Le deuxième type de représentation provenant directement de capteurs est la grille de pixels, obtenue par caméra. La grille de pixels permet de discrétiser un sous-espace de $\mathbb{R}^2$ borné à l'aide de cellules carrées élémentaires de taille fixe. Comme pour les nuages de points, chaque cellule possède des attributs supplémentaires, en général un niveau de gris ou bien de la couleur à trois canaux. En général, ces méthodes ne sont pas utilisées pour représenter des ensembles : comme nous l'avons expliqué en introduction de ce chapitre, on extrait des données pertinentes (les *features*) à l'aide de la grande variété d'outils issus du traitement et d'analyse d'image. Une utilisation possible d'une grille de pixels comme représentation d'ensemble est d'avoir une image binaire où chaque pixel possède une valeur binaire, ce qui peut être obtenu par segmentation [36] de l'image en deux segments (appelée aussi binarisation), mais cela limite grandement l'intérêt de ce format, et on préférera utiliser les représentations suivantes dans ce cas.

Remarque 15. *La grille de pixels peut se généraliser en 3D (voxels) ou plus (hypervoxel), mais ne sont en général pas extraits directement de capteurs, et sont surtout utilisés à des fins de simulation.*

Grille d'occupation

Comme pour les grilles de pixels, une grille d'occupation [37, 34] est une représentation où l'espace est discrétisé en une grille de cellules. Ici, chacune est associée à une probabilité d'occupation par un obstacle, avec pour valeurs limites 0 si elle estime que la cellule ne contient pas d'obstacle, et 1 si elle estime que la cellule contient un obstacle. Les approches *grid-based* du SLAM utilisent ce type de représentation, qui est plus généralement adaptée à la robotique probabiliste [38].

Pavages

Nous présentons ici un ensemble de représentations hiérarchiques. Les pavages réguliers, avec notamment le **quadtree** (2D) et l'**octree** (3D) [39, 40], et irréguliers avec le ***k*-d tree** [41], sont des structures arborescentes pour les nuages de points. À chaque nœud correspond une région de l'espace en forme de boîte. L'espace initial est donc une boîte intervalle de dimension 2 (quadtree), 3 (octree) ou k (k-d tree), et est le nœud racine. Les nœuds contenant au moins deux points sont divisés récursivement jusqu'à une certaine précision, en quatre sous-boîtes égales (quadtree), huit sous-boîtes égales (octree), ou deux sous-boîtes bissectées selon un hyperplan aligné avec les axes qu'il faut alors préciser (k-d tree). Un quadtree est illustré dans la figure 3.1.

Les **pavages intervalles**, initialement abordés dans [8] puis dans le chapitre 3 de [6], se distinguent des autres représentations présentées ci-dessus. Ils subdivisent l'espace en boîtes, tout en permettant de catégoriser ces boîtes par rapport à l'ensemble :

— les boîtes intérieures, incluses dans l'ensemble, ne contiennent que des points de l'ensemble à décrire,

— les boîtes extérieures, n'intersectant pas l'ensemble, ne contiennent aucun point de l'ensemble,

— les boîtes incertaines, qui n'ont pas pu être catégorisées en raison de la présence de points intérieurs et extérieurs à l'ensemble ou bien en raison de l'incertitude.

Pour construire un pavage intervalle, nous pouvons utiliser un *paveur* (voir en section 2.2 page 23) un peu différent. En effet nous avons vu qu'un paveur utilise un contracteur (voir page 19) afin d'obtenir un pavage de boîtes, dans lequel seules des boîtes extérieures peuvent être déterminées. Pour obtenir des boîtes intérieures, l'idée est alors d'utiliser un deuxième contracteur capable d'éliminer des boîtes intérieures. Un contracteur intérieur est en fait un contracteur extérieur appliqué au complémentaire de l'ensemble : les parties éliminées sont les parties extérieures du complémentaire, donc les parties intérieures. Le fonctionnement de ce paveur est illustré dans la figure 3.2.

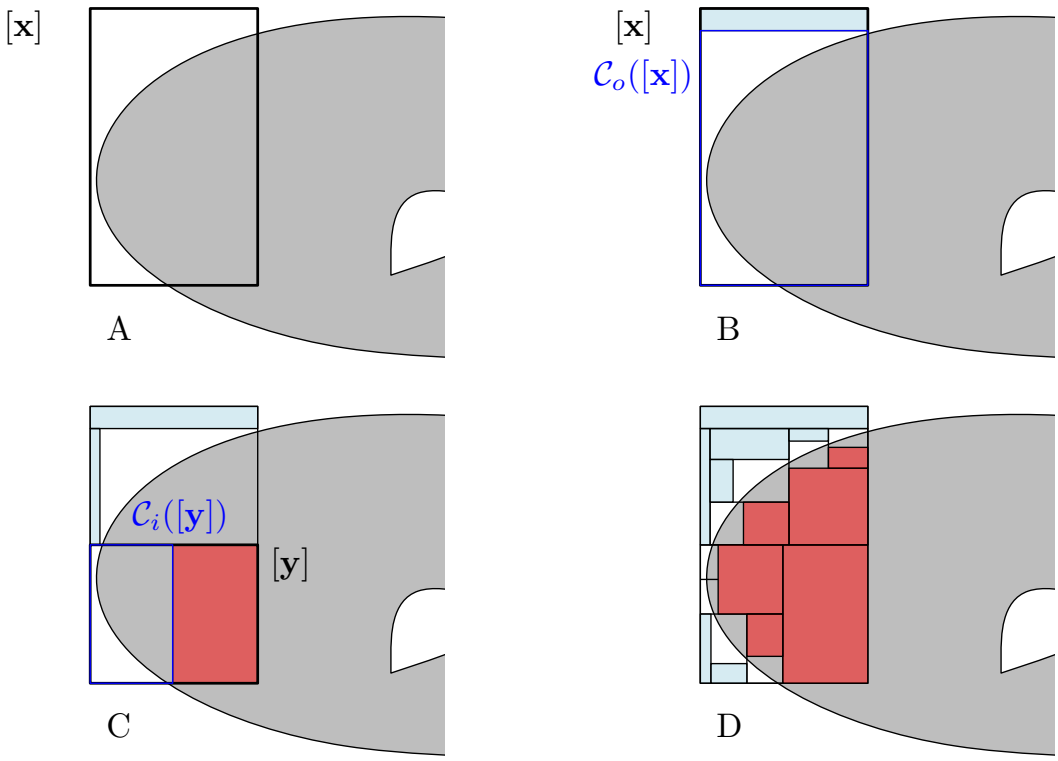


FIGURE 3.2 – Illustration de la construction d'un pavage intervalle à partir d'un contracteur extérieur à l'ensemble ($\mathcal{C}_o$) et un contracteur intérieur ($\mathcal{C}_i$). La boîte initiale [**x**] (A) est contractée avec le contracteur $\mathcal{C}_o$ (B), qui élimine une partie extérieure à l'ensemble en bleu. Cette dernière boîte est alors bissectée puis chacune est contractée (C), la cellule du bas est contractée par $\mathcal{C}_i$ et élimine une boîte intérieure à l'ensemble, en rouge. Le processus est répété encore deux fois (D) pour paver la boîte [**x**], pour un total de quatre phases de contraction.

Synthèse

Nous allons maintenant analyser ces représentations selon trois critères, nécessaires ou utiles pour les objectifs de cette thèse : la précision, la facilité de manipulation, la gestion des incertitudes.

Précision : Est-ce que la représentation peut approcher à précision désirée n'importe quel sous-ensemble de $\mathbb{R}^n$, c'est-à-dire étant donné une précision ε ; est-ce que la représentation permet de correctement identifier les points situés à distance d'au moins ε de la frontière comme étant intérieur ou extérieur lorsque l'incertitude est nulle, et pour quelle complexité spatiale ?

Les grilles de pixels sont une représentation assez particulière sur cet aspect, puisqu'à précision fixe. Les autres représentations sont incrémentales : pour les nuages de points, il est possible d'en rajouter, et les représentations arborescentes, dont les grilles d'occupation peuvent faire partie, peuvent s'étendre en profondeur.

Manipulation : A quel point la représentation est facile à manipuler avec d'autres outils ?

Comme évoqué dans leurs définitions respectives, toutes les représentations ont pour avantage d'utiliser des structures ou concepts riches dans la littérature, avec des outils performants pour une grande variété de tâches. Pour les pavages, nous allons considérer plus spécifiquement les pavages intervalles, qui intègrent très bien l'analyse par intervalles.

Incertitudes : Comment la représentation gère les incertitudes ?

C'est l'aspect sur lequel les représentations se distinguent le plus. Pour les approches probabilistes du SLAM, l'incertitude est modélisée par un modèle probabiliste. Les paramètres de ces modèles, comme le biais ou la variance, sont associés à chaque point du nuage. Pour les grilles d'occupation, chaque cellule possède une probabilité de présence d'obstacle, et une grille de pixel peut quantifier cette mesure par une nuance de gris. Les probabilités des cellules sont mises à jour selon le modèle probabiliste utilisé. Enfin, les pavages intervalles modélisent l'incertitude différemment, de manière ensembliste : toute l'incertitude est englobée dans des boîtes, donnant lieu aux boîtes incertaines des pavages. Les autres boîtes du pavages n'ont aucune incertitude. Les cellules incertaines peuvent être contractées avec des contracteurs, selon l'analyse par intervalles, qui garantit l'exactitude des calculs.

Parmi ces représentations, seul le pavage intervalle permet de gérer les incertitudes de manière garantie et exhaustive, c'est-à-dire en considérant toutes les possibilités. Il s'agit de l'atout principal de cette représentation choisie dans le cadre de la thèse. Nous avons aussi vu que ces pavages intervalles peuvent s'appuyer sur une paire de contracteurs. Il est alors possible d'utiliser une paire de contracteurs pour représenter un ensemble, résultant en une représentation analytique d'un ensemble : les contracteurs peuvent se substituer aux ensembles, comme nous l'avons vu à la page 21 avec l'algèbre des contracteurs.

Utiliser une paire de contracteurs, l'un cohérent avec l'ensemble et l'autre avec son complémentaire, permet d'être plus efficace pour l'approcher. Nous verrons dans la section 8.1 comment définir et utiliser ces opérateurs et cette structure de données.

3.3 Méthodes probabilistes de résolution de SLAM

Dans cette section, nous allons passer en revue quelques méthodes probabilistes de résolution du SLAM. Pour rappel, ces méthodes cherchent à déterminer la probabilité $P(\mathbf{x}(t_k), \mathbb{M} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}(t_0))$ pour tout instant k.

3.3.1 Filtres de Kalman

La méthode classique est l'utilisation des filtres de Kalman, et plus précisément l'*Extended Kalman Filter* (**EKF-SLAM**) [42, 43].

Cette méthode itérative sur k repose sur un calcul en deux étapes :

— l'étape de prédiction, basée sur le modèle de mouvement, qui prédit la distribution de probabilités sans prendre en compte les dernières mesures :

$$P(\mathbf{x}(t_k), \mathbb{M} \mid \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}(t_0)), \tag{3.9}$$

— l'étape de correction qui intègre les dernières mesures afin de corriger la prédiction de l'étape précédente, et correspond donc au calcul de la distribution qui nous intéresse :

$$P(\mathbf{x}(t_k), \mathbb{M} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}(t_0)). \tag{3.10}$$

Pour cette méthode, plusieurs hypothèses sont nécessaires :

— les modèles doivent être linéarisés,
— les incertitudes sont modélisées comme des distributions de probabilité gaussiennes.

Les filtres de Kalman ont en outre des limites :

— la linéarisation et les probabilités gaussiennes sont non réalistes,
— la carte doit être codée dans le vecteur d'état ; une carte trop riche en information non compressible, comme un environnement naturel, fait alors exploser la taille du problème.

3.3.2 Maximum de vraisemblance

Une autre manière d'approcher le problème est d'estimer le maximum de vraisemblance (*maximum likelihood estimation*). La vraisemblance est une fonction complémentaire aux probabilités, qui mesure comment le modèle statistique explique les données, en fonction des paramètres probabilistes donnés aux variables. Estimer le maximum de vraisemblance donne donc les paramètres qui expliquent au mieux les données.

Ce maximum de vraisemblance est calculé selon une approche basée plutôt sur l'Espérance-Maximization (*Expectation-Maximization* ou **EM** [44]). Le calcul du maximum par EM consiste en une méthode *hill-climbing* qui alterne entre deux étapes itérativement :

1. L'étape espérance (**E-step**) calcule les probabilités $P(\mathbf{x}_k \mid \mathbb{M}, \mathbf{Z}_N)$, c'est-à-dire calcule la trajectoire du robot selon la meilleure carte courante,
2. L'étape maximisation (**M-step**) trouve la carte $\mathbb{M}$ qui maximise la probabilité $P(\mathbb{M} \mid \mathbf{x}_k, \mathbf{Z}_N)$, c'est-à-dire construit la carte selon la trajectoire donnée par l'E-step.

Pour ces méthodes, les cartes sont codées sous forme de grille d'occupation ou de *landmarks*.

L'algorithme présenté dans [45] propose une solution au *Full SLAM*. **IML** [46] (pour *Incremental Maximum Likelihood*) en est une variante temps-réel.

Ces méthodes souffrent d'un manque de garantie de la solution retournée, dû à plusieurs causes :

— la méthode ne calcule qu'un maximum local,
— l'estimation de la carte est unique à chaque itération, donc il n'y a pas d'hypothèses multiples en cas d'ambiguïté, pouvant entraîner des mauvaises convergences,
— le modèle d'observation utilisée pour l'E-step est linéarisé,
— le maximum de vraisemblance ne garantit pas la solution : la solution correcte peut être moins vraisemblable qu'une mauvaise. Voir par exemple les exemples présentés dans le chapitre 13 de [38].

3.3.3 Filtres particulaires et Rao-Blackwellization

Un filtre particulaire (*Particle Filter*) est une méthode locale à population qui tire un nombre fini d'échantillons d'états, appelés *particules* en accord avec les distributions de probabilités et pondérées par leur vraisemblance. Ensuite, la méthode applique les étapes de prédiction et correction sur chacune des particules et ajuste leur poids, afin de recréer les distributions de probabilités suivantes.

En se basant sur la modélisation utilisée par les approches par filtres de Kalman, la carte fait partie du vecteur d'état, ce qui lui donne une dimension élevée. En essayant d'appliquer une approche particulaire, il nous faudrait un très grand nombre de particules pour que l'échantillonnage reste représentatif des distributions de probabilités, ce qui rend son application inutilisable en l'état.

Toutefois, il est possible de factoriser le problème dès lors que l'on considère la trajectoire plutôt que chaque pose prise séparément :

$$P(\mathbf{X}_{0:k}, \mathbb{M} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}(t_0)) = P(\mathbf{X}_{0:k} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}(t_0)) \cdot \prod_{\mathbf{m} \in \mathbb{M}} P(\mathbf{m} \mid \mathbf{X}_{0:k}, \mathbf{Z}_{0:k}). \tag{3.11}$$

Dans ce cas, l'estimation des *landmarks* peut être faite par EKF. Il ne reste alors plus que la trajectoire à échantillonner, ce qui permet d'avoir un nombre bien plus faible de particules. Ce procédé de factorisation est appelé *Rao-Blackwellization* [47].

Le nombre de particules nécessaires étant bien plus faible, les filtres particulaires deviennent possibles pour le SLAM. Nous pouvons citer **FastSLAM** [48], ou encore [49] qui utilise plutôt des grilles d'occupation.

Néanmoins, les filtres particulaires ont plusieurs limites :

— le nombre de particules reste trop élevé à grande dimension,
— une dégénérescence de la méthode due à la perte de diversité de l'échantillonnage (*particle depletion*, *particle deprivation* ou *sample impoverishment* [50]) :
  — la variance des poids des particules ne fait qu'augmenter dans ces méthodes [51]. Au final, la méthode dégénère jusqu'à ne considérer qu'une seule particule de poids significatif,
  — un rééchantillonnage (*resampling*), qui élimine les particules de trop faible poids et en sélectionne des nouvelles, est fait pour retarder le problème précédent,
— les erreurs d'estimation faites dans les étapes précédentes sont enregistrées dans les estimations de la carte, et ne sont donc pas éliminées par le rééchantillonnage. Ces méthodes souffrent donc d'un biais dès le départ.

3.3.4 GraphSLAM

L'algorithme *Full SLAM*, *feature-based GraphSLAM* [52] propose une approche différente, en codant le SLAM sous forme d'un graphe.

Chaque pose et chaque *feature* donne lieu à un nœud du graphe, et chaque arête code une relation impliquant les nœuds. Ainsi, il y a des arêtes reliant les poses consécutives pour les relations de mouvement, et une arête entre une pose et chacune des *features* observées depuis cette pose pour les relations d'observations. Ces relations sont linéarisées et résumées par une *matrice d'information*, comme illustré dans la figure 3.3. Le GraphSLAM diffère donc des autres méthodes en n'ayant pas de représentation explicite de la carte.

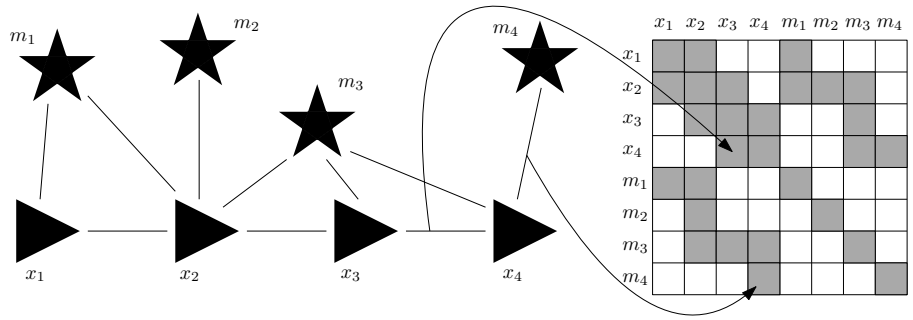


FIGURE 3.3 – Représentation graphique du problème (gauche). Les nœuds x (triangles) sont ceux de la trajectoire, les nœuds m (étoiles) sont les *landmarks*. Deux nœuds sont reliés si une relation existe entre eux, provenant du modèle de mouvement ou d'observation. Les relations sont représentées par une matrice d'information (droite), dans laquelle les entrées grisées correspondent à une relation du problème. Nous pouvons remarquer la structure creuse du problème, qui s'intensifie avec le temps écoulé, puisque le nombre de relations croît linéairement dans le temps, pour une taille de matrice quadratique.

Par les incertitudes des mesures, il y a des erreurs entre la valeur mesurée et la valeur prédite. L'objectif du GraphSLAM est de trouver la configuration qui minimise l'erreur globale, par des méthodes classiques d'optimisation.

La force de cette approche est d'exploiter la structure creuse (*sparse*) du problème de SLAM illustrée dans la figure 3.3, dans le sens où le robot ne perçoit que très peu de *features* à une position donnée par rapport à la totalité des *features* qu'il perçoit au cours de sa mission. Par sa méthode de résolution impliquant de l'élimination de variables, GraphSLAM est capable de traiter des problèmes avec un grand nombre de *features*. De plus, comme il s'agit d'une approche *full SLAM*, toute la trajectoire est estimée.

Cependant, le GraphSLAM possède quelques limitations au regard des objectifs de cette thèse :

— GraphSLAM utilise l'hypothèse de bruit gaussien indépendant, fausse en réalité,

— la linéarisation des relations induit des erreurs,

— la méthode est sensible à l'association de données, le graphe étant construit selon cette association. Une mauvaise association le rend non représentatif du problème.

3.4 Techniques et méthodes ensemblistes de résolution de SLAM

Nous avons vu que les techniques de SLAM classiques sont principalement basées sur sa définition probabiliste. Dans cette section, nous présentons des techniques de SLAM ensemblistes, traitant le problème de SLAM exprimé dans son formalisme CSP (3.8).

3.4.1 Choix de modélisation CSP et représentation de la carte

Le formalisme ensembliste propose une modélisation différente des incertitudes, qui est explicitée ici.

Hypothèses sur les données capteurs Dans une modélisation ensembliste, les incertitudes sont bornées par un intervalle, avec l'hypothèse que la réalité se trouve forcément dans cet intervalle. Pour faire le lien avec l'hypothèse probabiliste, l'intervalle englobe toutes les probabilités non nulles d'une modélisation probabiliste. Voir par exemple [12] pour une présentation de plusieurs applications ensemblistes basées sur le bornage des incertitudes, portant principalement sur de l'estimation de paramètres.

Représentation de la carte Nous avons évoqué trois différentes représentations de la carte pour les méthodes probabilistes, ainsi que le pavage intervalle. Lors de l'analyse de différents critères, nous avons souligné que le pavage intervalle était particulièrement adapté à la gestion des incertitudes.

Dans notre première contribution, nous utilisons une nouvelle version plus complète du pavage intervalle que nous appelons **pavage épais régulier**. Nous proposons aussi des méthodes à intervalles pour contracter la carte.

3.4.2 SLAM ensembliste

Bien qu'il existe différentes formulations de SLAM ensembliste [53, 54, 55], nous leur préférons les formulations CSP [56, 4] qui bénéficient du caractère déclaratif des CSP et des méthodes de résolution basées sur la propagation et la contraction.

La formalisation du problème que nous avons utilisée est similaire à celle du DigSLAM [4]. Dans celui-ci, l'environnement est abstrait sans *landmarks* identifiables, et le robot perçoit l'environnement au moyen de capteurs *range and bearing*, voire même *range-only* qui donnent la distance du plus proche objet sans direction associée. La figure 3.4 montre une mesure de type *range and bearing* et les ensembles que nous pouvons en extraire, avec leur propriétés.

Comme illustré dans le chapitre précédent consacré aux CSP, la manière classique d'approcher ce genre de problème est de le décomposer, quitte à introduire de nouvelles variables, jusqu'à obtenir des

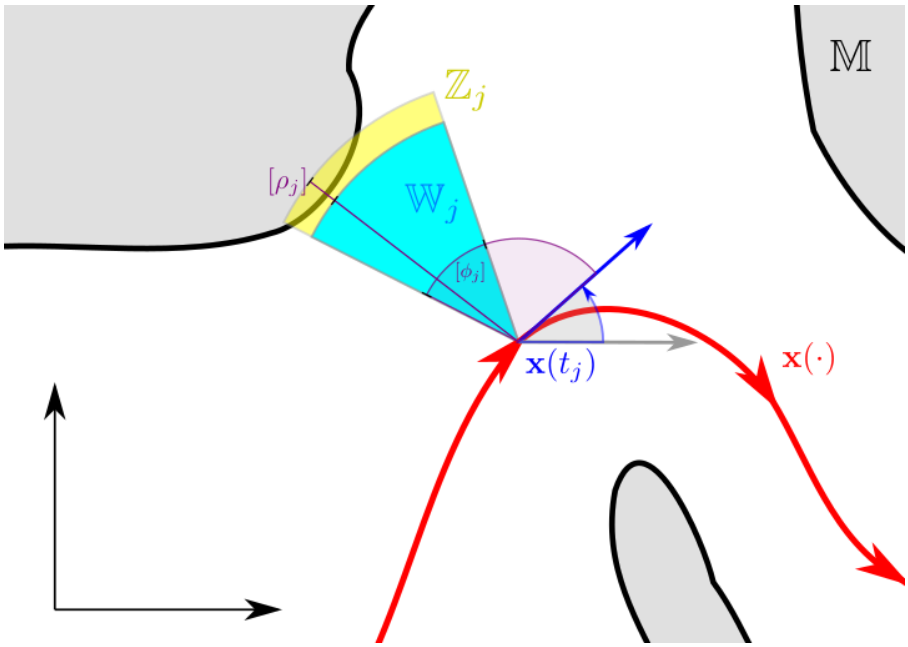


FIGURE 3.4 – Exemple d'une mesure prise à t_j par un capteur *range and bearing* de type sonar. Dans un cône $[\phi_j]$, le capteur retourne une distance incertaine $[\rho_j]$ du plus proche obstacle. Nous pouvons alors y associer une zone $\mathbb{W}_j = [0, \rho_j^-] \times [\phi_j]$ (en bleu) sans obstacle et une zone d'impact $\mathbb{Z}_j = [\rho_j] \times [\phi_j]$ (en jaune), garantie de contenir au moins un obstacle.

contraintes présentes dans notre catalogue de contraintes. Le DigSLAM traite le système suivant :

$$\begin{cases} \textbf{Variables :} \ \mathbb{M}, \mathbf{x}(\cdot) \\ \textbf{Domaines :} \ [\mathbb{M}], [\mathbf{x}](\cdot) \\ \textbf{Contraintes :} \\ \quad 1. \quad \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\ \quad 2. \quad \mathbf{x}(t_0) = \mathbf{0} \\ \quad 3. \quad \mathbf{h}(\mathbf{x}(t_j), \mathbb{W}_j) \subset \overline{\mathbb{M}} \qquad 0 \leq j \leq n-1 \\ \quad 4. \quad \mathbf{h}(\mathbf{x}(t_j), \mathbb{Z}_j) \cap \mathbb{M} \neq \emptyset \qquad 0 \leq j \leq n-1 \end{cases} \tag{3.12}$$

où la fonction $\mathbf{h}$ est le changement de repère décrit en introduction, qui permet ici de rassembler les mesures dans un repère commun. Les ensembles $\mathbb{W}_j$, $\mathbb{Z}_j$ illustrés dans la figure 3.4 sont des boîtes polaires déduites des mesures, donc considérées comme des paramètres.

La contrainte 1 est la contrainte d'évolution écrite comme une Équation Différentielle Ordinaire (**EDO**), qui décrit le comportement du robot en fonction de son état et de sa commande, et la contrainte 2 code la condition initiale. Nous avons déjà évoqué dans le chapitre 2 dédié aux CSP, et en particulier dans l'application décrite en section 2.4, comment traiter la contrainte 1, et la condition initiale est traitée avec le contracteur $\mathcal{C}_{eval}$.

Ce qui fait la particularité et donne son nom au DigSLAM est la contraction de la carte qui est permise par la contrainte 3 : partant d'une carte complètement inconnue, il est possible de *creuser* les zones dépourvues d'obstacles que sont les $\mathbb{W}_j$ en ayant pris soin de les rassembler dans un même repère. Cette contrainte consiste donc en pratique à une manipulation de sous-ensembles réels par de l'arithmétique

réelle (changement de repère) et l'algèbre des parties d'un ensemble (la zone creusée est une union des différentes zones sans obstacles).

La contrainte 4 est la seconde partie de la contrainte d'observation, qui permet de contracter l'état. En effet, la contrainte exprime que la zone d'impact contient des obstacles. Ce que l'on peut en déduire est que cette zone d'impact ne peut pas se retrouver dans une partie préalablement creusée par la contrainte 3. Toutes les positions où cette situation se produit sont donc incohérentes et retirées du domaine de la trajectoire. Tout comme la précédente, cette contrainte se base sur l'arithmétique réelle appliquée aux ensembles et sur la logique booléenne ensembliste, et peut être décomposée pour en déduire un contracteur.

La résolution d'une instance de DigSLAM passe par une contraction successive de chaque domaine induite par une boucle de propagation : la contrainte 3 permet de contracter la carte selon la trajectoire, et la contrainte 4 permet de contracter la trajectoire selon la carte. La figure 3.5 montre ces contractions de trajectoire et de carte pour une instance de DigSLAM, qui utilise des capteurs *range-only*, où en particulier les ensembles $\mathbb{W}_j$ sont des disques et $\mathbb{Z}_j$ sont des anneaux.

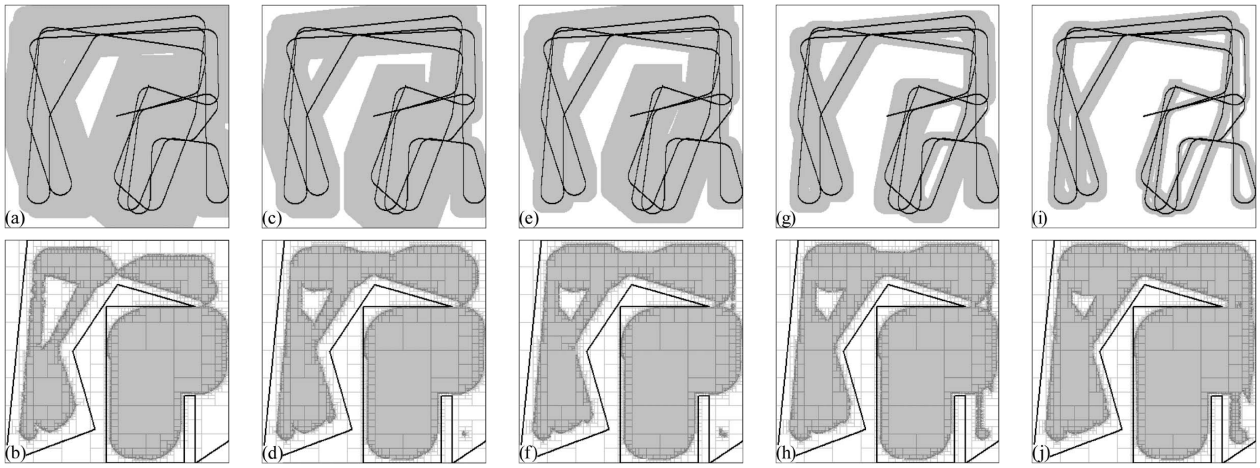


FIGURE 3.5 – Contractions successives du tube $[\mathbf{x}](\cdot)$ (en gris sur la ligne du haut) et de la zone dépourvue d'obstacle ($\overline{\mathbb{M}}$ dans le système 3.12 ; en gris sur la ligne du bas). La trajectoire réelle (en haut) et la frontière réelle de la carte (en bas) sont représentées en noir.

Deuxième partie

Contribution 1 : SLAM ensembliste sous transformation rigide

Chapitre 4

Présentation générale d'un problème de SLAM ensembliste

Plan du chapitre

Nous avons vu différents problèmes de recalage en introduction. Puis dans les deux chapitres consacrés à l'état de l'art, nous avons étudié les problèmes de satisfaction de contraintes (CSP), en particulier les CSP hybrides, où les variables du problème ont des natures différentes : tantôt réelles, tantôt trajectoires. Ensuite, le chapitre consacré au problème de SLAM en robotique nous a permis de voir comment celui-ci peut être exprimé sous la forme d'un CSP hybride.

Dans ce chapitre, nous allons détailler une version du problème de recalage et comment l'intégrer dans un problème de SLAM formalisé comme un CSP avec des variables ensemblistes.

4.1 Motivations et hypothèses

Commençons par rappeler le problème de recalage ensembliste sur ensembles partiels présenté en introduction.

Dans ce problème, nous avons un robot suivant une trajectoire $\mathbf{x}(\cdot)$, se déplaçant dans un environnement $\mathbb{M}$ d'obstacles, fermé, préalablement inconnu. Il dispose de capteurs, permettant de faire des observations modélisées par une fonction d'observation $\mathbf{g}$. À différents instants t_i, il mesure donc à la fois une zone sans obstacle $\mathbb{W}_i(t_i)$ et une zone d'impact $\mathbb{Z}_i(t_i)$ comprenant au moins un obstacle.

Nous notons $\mathcal{R}_i$ les repères locaux du robot aux instants t_i, et $\mathbf{x}_{\mathcal{R}_i}(\cdot)$ la trajectoire exprimée dans le repère $\mathcal{R}_i$. Pour les variables ensemblistes, seul l'indice du repère est indiqué, comme utilisé pour les

zones sans obstacle et d'impact.

Le repère dans lequel $\mathbf{x}(\cdot)$ est exprimé est $\mathcal{R}_0$:

$$\mathbf{x}_{\mathcal{R}_0}(\cdot) = \mathbf{x}(\cdot) \tag{4.1}$$

$$\forall i \in \{0, \dots, m-1\}, \quad \begin{cases} \mathbb{W}_i(t_i), \mathbb{Z}_i(t_i) = \mathbf{g}(\mathbf{x}(t_i), \mathbb{M}) \\ \mathbb{W}_i(t_i) \subset \overline{\mathbb{M}_i} \\ \mathbb{Z}_i(t_i) \subset \partial\mathbb{M}_i \end{cases} \tag{4.2}$$

$$\forall (i,j) \in \{0, \dots, m-1\}^2, \quad \begin{cases} \mathbb{W}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{W}_i(t_i)) \\ \mathbb{Z}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{Z}_i(t_i)) \end{cases} \tag{4.3}$$

Nous avons aussi vu que les dernières relations du système (4.2) peuvent être remplacées par les relations suivantes, plus complètes :

$$\forall j \in \{0, \dots, m-1\}, \quad \begin{cases} \bigcup\limits_{0 \leq i \leq m-1} \mathbb{W}_j(t_i) \subset \overline{\mathbb{M}_j}, \\ \bigcup\limits_{0 \leq i \leq m-1} \mathbb{Z}_j(t_i) \subset \partial\mathbb{M}_j. \end{cases} \tag{4.4}$$

Cette modélisation du problème induit plusieurs hypothèses :

(i) l'environnement est constant au cours du temps,

(ii) il n'y a pas de données aberrantes : les mesures sont correctes mais incertaines.

Afin de prendre en compte la dynamique du robot, nous considérons l'équation d'évolution du robot et sa condition initiale, comme vu dans la section 3.1 (page 38) :

$$\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \tag{4.5}$$

$$\mathbf{x}(t_0) = \mathbf{0}. \tag{4.6}$$

En général, la commande $\mathbf{u}(\cdot)$ consiste en la consigne donnée aux moteurs, qui permet d'en déduire certaines composantes de $\dot{\mathbf{x}}(\cdot)$.

Concernant la dynamique du robot, nous nous basons sur un « modèle char » classique. Dans celui-ci, l'état du robot est un vecteur en quatre dimensions avec :

— $x_1(\cdot), x_2(\cdot)$ étant la position du robot dans le repère cartésien,

— $x_3(\cdot)$ est l'orientation du robot,

— $x_4(\cdot)$ est la vitesse linéaire du robot, sachant que le robot se déplace devant lui, le vecteur vitesse étant orienté selon $x_3(\cdot)$.

La fonction d'évolution $\mathbf{f}$ est alors définie comme :

$$\mathbf{f} : \begin{cases} \mathbb{R}^4 \times \mathbb{R}^2 \to \mathbb{R}^4, \\ \mathbf{x}(\cdot), \mathbf{u}(\cdot) \mapsto \begin{pmatrix} x_4(\cdot)\cos(x_3(\cdot)) \\ x_4(\cdot)\sin(x_3(\cdot)) \\ u_1(\cdot) \\ u_2(\cdot) \end{pmatrix} \end{cases} \tag{4.7}$$

où $u_1(\cdot)$ est la vitesse angulaire du robot et $u_2(\cdot)$ est son accélération, données par la commande aux moteurs ainsi que mesurée par différents capteurs.

Cette relation s'appuie alors sur les hypothèses suivantes :

(iii) l'équation dynamique $\mathbf{f}$ régissant l'état du robot est connue,

(iv) la connaissance et/ou mesure de la commande du robot $\mathbf{u}(\cdot)$ est encadrée par un tube $[\mathbf{u}](\cdot)$.

4.2 Modélisation CSP du problème

Dans l'introduction, nous avons proposé une version ensembliste du SLAM, s'appuyant sur le **Dig-SLAM** [4] évoqué dans la section 3.4 (page 50). Dans cette version, nous utilisons des variables ensemblistes possédant différentes propriétés qui permettent ensuite de contraindre les variables du problème de SLAM. Pour comparer cette version du SLAM aux versions évoquées dans la section 3.2, les équations (4.2) à (4.4) permettent d'expliciter le modèle d'observation de manière ensembliste en rendant plus directs les liens entre les mesures, la trajectoire et l'environnement.

Cependant, l'obtention des ensembles $\mathbb{W}_i(t_i), \mathbb{Z}_i(t_i)$ n'est pas directe avec les capteurs utilisés. Pour ce problème, nous utilisons des capteurs *range and bearing* qui effectuent des mesures $k \in K$ aux instants t_i selon l'équation d'observation suivante :

$$(\rho_k(t_i), [\phi_k(t_i)]) = \mathbf{g}_k(\mathbf{x}(t_i), \mathbb{M}). \tag{4.8}$$

La fonction d'observation $\mathbf{g}_k$ est celle d'un capteur *range and bearing* qui fonctionne selon le principe suivant : dans un cône $\mathbb{R}^+ \times [\phi_k(t_i)]$ prédéterminé, le capteur retourne la distance de l'obstacle $\mathbf{z}_k(t_i) = (\rho_k(t_i), \phi_k(t_i))$ le plus proche, comme illustré dans la figure 4.1 à gauche.

Les ensembles $\mathbb{W}_i(t_i), \mathbb{Z}_i(t_i)$ sont déduits des mesures précédentes par les équations suivantes :

$$\begin{cases} \mathbb{W}_i(t_i) = \bigcup_{k \in K} [0, \rho_k(t_i)[\times [\phi_k(t_i)] \\ \mathbb{Z}_i(t_i) = \bigcup_{k \in K} \{\mathbf{z}_k(t_i)\} \end{cases} \tag{4.9}$$

En modélisant les incertitudes sur la distance par un intervalle $[\rho_k(t_i)]$ tout en garantissant que $\rho_k^-(t_i) < \rho_k(t_i)$, nous obtenons les relations suivantes :

$$\begin{cases} \mathbb{W}_i(t_i) \supset \bigcup_{k \in K} [0, \rho_k^-(t_i)] \times [\phi_k(t_i)] \\ (\mathbb{Z}_i(t_i))_k \in [\rho_k(t_i)] \times [\phi_k(t_i)] \qquad \forall k \in K \end{cases} \tag{4.10}$$

Ces relations sont illustrées dans la figure 4.1 à droite.

Remarque 16. *Les données capteurs, avec leurs incertitudes, sont considérées comme des paramètres dorénavant : elles sont présentes dans la modélisation, mais leurs incertitudes n'ont pas vocation à être réduites. C'est pourquoi les contraintes qui les font intervenir s'appuieront sur les données incertaines à disposition, et non les $(\rho_k(t_i), \phi_k(t_i))$ réels.*

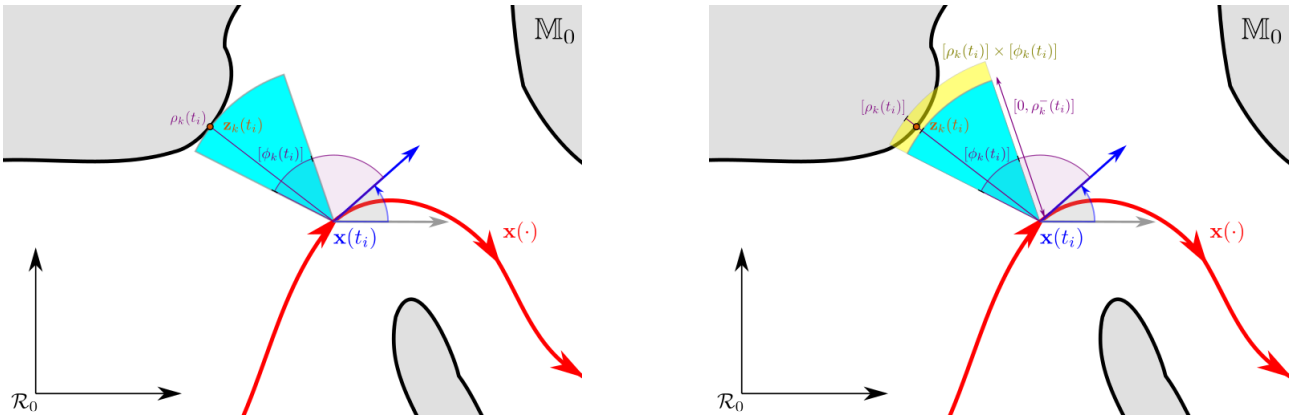


FIGURE 4.1 – Illustration d'une mesure $k \in K$ prise à l'instant t_i dans un cône $\mathbb{R}^+ \times [\phi_k(t_i)]$. À gauche : la boîte polaire bleue $[0, \rho_k(t_i)[\times[\phi_k(t_i)]$ est garantie de ne contenir aucun obstacle, et représente une partie de $\mathbb{W}_i(t_i)$, et le point $\mathbf{z}_k(t_i)$ ayant déclenché la mesure se situe à la frontière de $\mathbb{M}_0$. À droite : en comptant les incertitudes, nous pouvons garantir que la boîte polaire bleue $[0, \rho_k(t_i)^-] \times [\phi_k(t_i)]$ ne contient aucun obstacle, et la boîte polaire jaune $[\rho_k(t_i)] \times [\phi_k(t_i)]$ est garantie de contenir le point $\mathbf{z}_k(t_i)$, et donc au moins un point obstacle.

À partir des équations (4.2), (4.3) et (4.10), nous pouvons en déduire un CSP :

$$
\begin{cases}
\textbf{Variables :} \ \mathbf{x}(\cdot),\ (\mathbb{M}_i)_{0\le i\le m-1},\ (\mathbb{W}_i(t_j))_{0\le i,j\le m-1},\ (\mathbb{Z}_i(t_j))_{0\le i,j\le m-1} \\
\textbf{Domaines :} \ [\mathbf{x}](\cdot),\ ([\mathbb{M}_i])_{0\le i\le m-1},\ ([\mathbb{W}_i(t_j)])_{0\le i,j\le m-1},\ ([\mathbb{Z}_i(t_j)])_{0\le i,j\le m-1} \\
\textbf{Paramètres :} \ [\rho_k(t_i)]_{0\le i\le m-1,\,k\in K},\ [\phi_k(t_i)]_{0\le i\le m-1,\,k\in K} \\
\textbf{Contraintes :} \\
\quad 1. \quad \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \qquad 0 \le j \le m-1 \\
\quad 2. \quad \mathbf{x}(t_0) = \mathbf{0} \qquad 0 \le j \le m-1 \\
\quad 3. \quad \mathbb{W}_i(t_i) \supset \bigcup_{k\in K} [0, \rho_k^-(t_i)] \times [\phi_k(t_i)] \qquad 0 \le i \le m-1 \\
\quad 4. \quad (\mathbb{Z}_i(t_i))_k \in [\rho_k(t_i)] \times [\phi_k(t_i)] \qquad 0 \le i \le m-1,\ k \in K \\
\quad 5.1. \quad \mathbb{W}_j(t_i) \subset \overline{\mathbb{M}_j} \qquad 0 \le i,j \le m-1 \\
\quad 5.2. \quad \mathbb{W}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{W}_i(t_i)) \qquad 0 \le i,j \le m-1 \\
\quad 6.1. \quad \mathbb{Z}_j(t_i) \subset \partial\mathbb{M}_j \qquad 0 \le i,j \le m-1 \\
\quad 6.2. \quad \mathbb{Z}_j(t_i) = \mathbf{h}(\mathbf{x}_{\mathcal{R}_j}(t_i), \mathbb{Z}_i(t_i)) \qquad 0 \le i,j \le m-1
\end{cases}
\tag{4.11}
$$

où $\mathbf{x}_{\mathcal{R}_j}(t_i)$, qui est l'expression de la pose $\mathbf{x}(t_i)$ dans le repère $\mathcal{R}_j$, correspond à la différence de pose entre t_i et t_j dans le repère $\mathcal{R}_j$, puisque $\mathbf{x}_{\mathcal{R}_j}(t_j) = \mathbf{0}$ par définition du repère $\mathcal{R}_j$.

Remarque 17. *Pour les contraintes* 5.2 *et* 6.2*, la quantité* $\mathbf{x}_{\mathcal{R}_j}(t_i)$ *peut se calculer à partir de l'expression suivante :*

$$\mathbf{x}_{\mathcal{R}_j}(t_i) = \int_{t_j}^{t_i} \mathbf{f}(\mathbf{x}_{\mathcal{R}_j}(t), \mathbf{u}(t))\mathrm{d}t. \tag{4.12}$$

En pratique, seule la pose du robot nous intéresse, ce que nous pouvons calculer en intégrant la vitesse pour la position dans le plan, et l'orientation par l'intégrale de la vitesse angulaire, ou alors mesurée directement par une boussole.

Remarque 18. *Les contraintes* 5.1 *et* 6.1 *sont toutes les deux des conjonctions de contraintes, sur i et sur j. Elles sont issues des équations* (4.4) *mais, plutôt que d'ajouter d'un coup des éléments à $\mathbb{M}_j$ avec des unions, ceux-ci sont ajoutés au fur et à mesure au moyen d'une conjonction sur i.*

Cette modélisation du problème permet de tirer des relations pour toutes les paires d'instants t_i, t_j, ce qui permet donc de considérer n'importe quelle potentielle fermeture de boucle. Cependant, cette modélisation demande à la fois un nombre quadratique de variables ensemblistes et de contraintes. Les complexités temporelle et spatiale de cette modélisation rendent alors sa résolution trop coûteuse.

Une possibilité est de ne considérer qu'un sous-système, en ne sélectionnant que certaines contraintes. Afin de savoir comment faire cette sélection, nous pouvons nous référer à quelles paires d'instants il est intéressant de considérer pour le SLAM évoqué dans la section 3.2 (page 40). En particulier, nous avons relevé trois cas de figure :

— entre instants consécutifs t_i et $t_i + 1$ pour du *scan matching*,

— entre deux instants t_i et t_j éloignés dans le temps mais localisés au même endroit afin de faire une fermeture de boucle efficace,

— entre les instants t_i et t_0, ce qui revient à tout exprimer dans un référentiel commun donné par $\mathbf{x}(t_0)$.

Comme nous ne détectons pas les fermetures de boucles, nous allons nous intéresser aux deux autres cas de figure, c'est-à-dire les instants consécutifs ou les paires t_i, t_0. Avec cette sélection, nous avons un nombre linéaire de variables ensemblistes et de contraintes, ce qui rend la résolution du CSP réalisable. Dans la suite, nous n'allons considérer que les paires d'instants t_i, t_0, puisqu'elles sont suffisantes pour traiter le problème global. En effet, les informations issues d'une paire t_i, t_j quelconque sont exploitées par décomposition, en traitant les paires t_i, t_0 et t_j, t_0 au prix d'un pessimisme plus important : notamment, on ne pourra plus exploiter la remarque 17.

Cela reste néanmoins réaliste tant que les dérives entre t_i et t_0, et entre t_j et t_0, ne sont pas trop importantes.

De plus, les contraintes 5.1, 5.2 et 6.1, 6.2 sont sujettes à la substitution, ce qui permet de simplifier l'énoncé du problème tout en évitant des variables intermédiaires dans le CSP, maintenant obsolètes.

Remarque 19. *Ces substitutions rendent la modélisation du problème plus simple, où chaque contrainte du système a un rôle spécifique détaillé dans la suite. Pour la résolution cependant, ces contraintes ont tout intérêt à être décomposées en contraintes plus simples. Cette décomposition passe en partie par celle avant la substitution des variables, mais ne sera décrite que dans le chapitre 6 dédié à l'implémentation des outils de résolution.*

Après ces simplifications, le CSP correspondant est le suivant :

$$
\left\{
\begin{array}{l}
\textbf{Variables :}\ \mathbf{x}(\cdot),\ \mathbb{M}_0,\ (\mathbb{W}_i(t_i))_{0\leq i\leq m-1}\,,\ (\mathbb{Z}_i(t_i))_{0\leq i\leq m-1} \\
\textbf{Domaines :}\ [\mathbf{x}](\cdot),\ [\mathbb{M}_0],\ ([\mathbb{W}_i(t_i)])_{0\leq i\leq m-1}\,,\ ([\mathbb{Z}_i(t_i)])_{0\leq i\leq m-1} \\
\textbf{Paramètres :}\ [\rho_k(t_i)]_{0\leq i\leq m-1,\,k\in K},\ [\phi_k(t_i)]_{0\leq i\leq m-1,\,k\in K} \\
\textbf{Contraintes :} \\
\quad 1.\quad \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\
\quad 2.\quad \mathbf{x}(t_0) = \mathbf{0} \\
\quad 3.\quad \mathbb{W}_i(t_i) \supset \bigcup_{k\in K} [0, \rho_k^-(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1 \\
\quad 4.\quad (\mathbb{Z}_i(t_i))_k \in [\rho_k(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1,\, k \in K \\
\quad 5.\quad \mathbf{h}(\mathbf{x}(t_i), \mathbb{W}_i(t_i)) \subset \overline{\mathbb{M}_0} \qquad 0 \leq i \leq m-1 \\
\quad 6.\quad \mathbf{h}(\mathbf{x}(t_i), \mathbb{Z}_i(t_i)) \subset \partial\mathbb{M}_0 \qquad 0 \leq i \leq m-1
\end{array}
\right. \tag{4.13}
$$

Cette dernière modélisation fait intervenir plusieurs contraintes différentes. Dans le reste de ce chapitre nous détaillons ces contraintes, et l'implémentation de contracteurs associés se fera dans le chapitre suivant.

La contrainte d'évolution (1) et la condition initiale (2) ayant été abordées à plusieurs reprises, nous pourrons nous référer aux sections 2.4 et 3.5 pour voir comment les traiter.

Les contraintes 3 et 4 permettent de définir des variables ensemblistes intermédiaires, qui sont ensuite utilisées dans les contraintes 5 pour cartographier l'environnement, et les contraintes 6 pour localiser le robot.

4.2.1 Contraintes de cartographie de cartes locales

Les contraintes 3 du système (4.13) sont rappelées ici :

$$3.\quad \mathbb{W}_i(t_i) \supset \bigcup_{k\in K} [0, \rho_k^-(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1$$

Elles impliquent une inclusion et une union d'ensembles. Nous verrons dans le chapitre 5 comment traiter chacune de ces opérations de l'algèbre des parties d'un ensemble afin d'en proposer un contracteur adapté.

Les contraintes 4 sont aussi assez explicites :

$$4.\quad (\mathbb{Z}_i(t_i))_k \in [\rho_k(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1,\, k \in K$$

Cette contrainte permet de borner un ensemble d'obstacles $(\mathbb{Z}_i(t_i))$ avec les données capteurs. Comme ces obstacles sont ici ponctuels, un contracteur associé à cette contrainte est trivial et sera rappelé dans la section 6.1.

4.2.2 Contrainte de cartographie de la carte globale

Rappelons les contraintes 5 du système (4.13) :

$$5. \quad \mathbf{h}(\mathbf{x}(t_i), \mathbb{W}_i(t_i)) \subset \overline{\mathbb{M}_0} \qquad 0 \leq i \leq m-1.$$

Ces contraintes permettent de contracter le domaine de la carte $\mathbb{M}_0$ en ajoutant les ensembles $\mathbb{W}_i(t_i)$, une fois ceux-ci exprimés dans le repère $\mathcal{R}_0$, à notre connaissance de l'extérieur de $\mathbb{M}_0$.

Nous avons déjà évoqué une décomposition possible de cette contrainte en le système suivant :

$$\left\{ \begin{array}{l} \textbf{Variables :} \ \mathbb{M}_0,\ \mathbb{W}_0(t_i),\ \mathbb{W}_i(t_i),\ \mathbf{x}(t_i) \\ \textbf{Domaines :} \ [\mathbb{M}_0],\ [\mathbb{W}_0(t_i)],\ [\mathbb{W}_i(t_i)],\ [\mathbf{x}(t_i)] \\ \textbf{Contraintes :} \\ \quad 5.1. \quad \mathbb{W}_0(t_i) \subset \overline{\mathbb{M}_0} \qquad\qquad 0 \leq i \leq m-1 \\ \quad 5.2. \quad \mathbb{W}_0(t_i) = \mathbf{h}(\mathbf{x}(t_i), \mathbb{W}_i(t_i)) \qquad 0 \leq i \leq m-1 \end{array} \right. \tag{4.14}$$

Les contraintes 5.1 ne font intervenir que des opérations de l'algèbre des parties d'un ensemble, et nous verrons qu'il est possible de traiter directement ce type de contrainte.

En revanche, les contraintes 5.2 sont plus compliquées à traiter, puisqu'elles reviennent à estimer l'image d'un pavage épais par une fonction arithmétique, qui impliquent en plus des paramètres incertains.

L'objectif de cette partie est de se ramener à des calculs plus simples pour lesquels il est facile de proposer un contracteur.

Dans un premier temps nous décomposons la transformation rigide en une rotation suivie d'une translation. Ensuite, nous verrons comment passer d'un système de coordonnées cartésiennes à polaires, et vice-versa. Comme la translation (respectivement la rotation) est une somme dans un système de coordonnées cartésiennes (respectivement polaires), nous n'aurons besoin au total que de deux contraintes : la somme entre un sous-ensemble réel et un vecteur, et le changement de coordonnées cartésiennes/ polaires.

Décomposition de la transformation en une rotation et une translation

En explicitant le changement de repère, la contrainte de transformation 5.2 que nous étudions est la suivante :

$$\left\{ \begin{array}{l} \textbf{Variables :} \ \mathbb{W}_0(t_i),\ \mathbb{W}_i(t_i),\ \mathbf{x}(t_i) \\ \textbf{Domaines :} \ [\mathbb{W}_0(t_i)],\ [\mathbb{W}_i(t_i)],\ [\mathbf{x}(t_i)] \\ \textbf{Contrainte :} \ \mathbb{W}_0(t_i) = \mathbf{R}_{x_3(t_i)} \mathbb{W}_i(t_i) + \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix}. \end{array} \right. \tag{4.15}$$

où $\mathbf{R}_{x_3(t_i)}$ est la matrice de rotation d'angle $x_3(t_i)$.

En introduisant une variable intermédiaire, nous pouvons obtenir le système suivant :

$$\begin{cases} \textbf{Variables :} \ \mathbb{W}_0(t_i),\ \mathbb{W}_i(t_i),\ \mathbb{W}_{\mathbf{R}}(t_i),\ \mathbf{x}(t_i) \\ \textbf{Domaines :} \ [\mathbb{W}_0(t_i)],\ [\mathbb{W}_i(t_i)],\ [\mathbb{W}_{\mathbf{R}}(t_i)],\ [\mathbf{x}(t_i)] \\ \textbf{Contraintes :} \\ \quad 1. \quad \mathbb{W}_{\mathbf{R}}(t_i) = \mathbf{R}_{x_3(t_i)} \mathbb{W}_i(t_i) \\ \quad 2. \quad \mathbb{W}_0(t_i) = \mathbb{W}_{\mathbf{R}}(t_i) + \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} \end{cases} \tag{4.16}$$

où $\mathbb{W}_{\mathbf{R}}(t_i)$ est une variable intermédiaire, à une rotation près de $\mathbb{W}_i(t_i)$ et une translation près de $\mathbb{W}_0(t_i)$.

Nous disposons désormais de deux contraintes : l'une consistant en une translation incertaine, l'autre en une rotation incertaine. Afin de simplifier leur traitement, nous introduisons un changement de coordonnées permettant d'exprimer la translation en coordonnées cartésiennes et la rotation en coordonnées polaires.

Changement de coordonnées cartésiennes/polaires

La deuxième technique utilisée est le changement de coordonnées cartésiennes/polaires. En effet, d'une part les coordonnées polaires sont plus adaptées pour effectuer des rotations, qui sont alors une somme, ainsi que pour traiter les ensembles $\mathbb{W}_i(t_i)$ qui consistent en une union de boîtes polaires (voir la section précédente). D'autre part, les coordonnées cartésiennes sont adaptées aux translations, qui reviennent aussi à une somme.

Nous voulons donc des représentations tantôt cartésiennes, tantôt polaires, afin d'avoir le repère le plus adapté pour la transformation, ainsi que proposer un contracteur permettant de faire le lien entre ces représentations.

Pour un sous-ensemble réel $\mathbb{X} \subset \mathbb{R}^2$ de points en coordonnées cartésiennes, nous notons $\mathbb{X}^P$ ce même ensemble de points, mais exprimé en coordonnées polaires. Nous avons donc les relations suivantes :

$$\mathbb{X}^P = \left\{ (\rho, \theta) \mid \exists (x, y) \in \mathbb{X},\ \rho = \sqrt{x^2 + y^2},\ \theta \equiv \operatorname{atan2}(y, x) \pmod{2\pi} \right\} \tag{4.17}$$

$$\mathbb{X} = \left\{ (x, y) \mid \exists (\rho, \theta) \in \mathbb{X}^P,\ x = \rho \cos(\theta),\ y = \rho \sin(\theta) \right\}. \tag{4.18}$$

La fonction $\operatorname{atan2}(y, x)$ est une variante de l'arc tangente qui, en étudiant indépendemment les signes de ses arguments, renvoie l'angle de (x, y) dans l'intervalle $]-\pi, \pi]$.

La contrainte $\mathcal{L}_{\text{cart/pol}}$ permettant de faire le lien entre ensemble cartésien et polaire est la suivante :

$$\begin{cases} \textbf{Variables :} \ \mathbb{X},\ \mathbb{X}^P \\ \textbf{Domaines :} \ [\mathbb{X}],\ [\mathbb{X}^P] \\ \textbf{Contrainte :} \ \mathbb{X} \overset{cart/pol}{\longleftrightarrow} \mathbb{X}^P \end{cases} \tag{4.19}$$

En utilisant cette nouvelle contrainte, nous pouvons à nouveau décomposer la contrainte de transfor-

mation (4.15) en le système suivant :

$$\begin{cases} \textbf{Variables :} \ \mathbb{W}_0(t_i),\ \mathbb{W}_i^P(t_i),\ \mathbb{W}_{\mathbf{R}}(t_i),\ \mathbb{W}_{\mathbf{R}}^P(t_i),\ \mathbf{x}(t_i) \\ \textbf{Domaines :} \ [\mathbb{W}_0(t_i)],\ [\mathbb{W}_i^P(t_i)],\ [\mathbb{W}_{\mathbf{R}}(t_i)],\ [\mathbb{W}_{\mathbf{R}}^P(t_i)],\ [\mathbf{x}(t_i)] \\ \textbf{Contraintes :} \\ \quad 1. \quad \mathbb{W}_{\mathbf{R}}(t_i) \overset{cart/pol}{\longleftrightarrow} \mathbb{W}_{\mathbf{R}}^P(t_i) \\ \quad 2. \quad \mathbb{W}_{\mathbf{R}}^P(t_i) = \mathbb{W}_i^P(t_i) + \begin{pmatrix} 0 \\ x_3(t_i) \end{pmatrix} \\ \quad 3. \quad \mathbb{W}_0(t_i) = \mathbb{W}_{\mathbf{R}}(t_i) + \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} \end{cases} \tag{4.20}$$

Cette dernière décomposition est intéressante puisqu'elle ne fait intervenir que deux types de contraintes, à savoir une contrainte de changement de coordonnées cartésiennes/polaires $\mathcal{L}_{\text{cart/pol}}$ (1) et deux contraintes de somme $\mathcal{L}_+$ (2, 3) définies comme suit :

$$\begin{cases} \textbf{Variables :} \ \mathbb{X},\ \mathbb{Y},\ \mathbf{z} \\ \textbf{Domaines :} \ [\mathbb{X}],\ [\mathbb{Y}],\ [\mathbf{z}] \\ \textbf{Contrainte :} \ \mathbb{X} = \mathbb{Y} + \mathbf{z} \end{cases} \tag{4.21}$$

Le système (4.20) est synthétisé dans le graphe de contraintes présenté dans la figure 4.2.

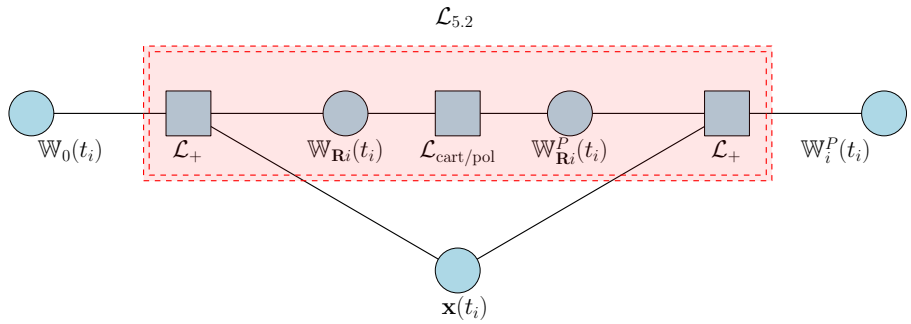


FIGURE 4.2 – Graphe de contraintes résumant la décompositon de la transformation rigide incertaine d'un sous-ensemble réel.

Remarque 20. *Les variables $\mathbb{W}_0^P(t_i)$ ne sont nécessaires que pour les observations faites à t_0 ce qui peut être traité à part, et les coordonnées cartésiennes des variables $\mathbb{W}_i^P(t_i)$ n'ont pas d'intérêt ici.*

Amélioration : double décomposition de la transformation

La décomposition de la transformation rigide n'est pas unique, et nous allons aussi nous intéresser à une seconde décomposition de la contrainte 5.2 qui consiste en une translation suivie d'une rotation.

Cette seconde décomposition présente plusieurs intérêts :

— elle permet de définir la transformation inverse de la première décomposition avec les mêmes paramètres,

— elle amène une autre manière de calculer la même quantité, en faisant intervenir d'autres incertitudes et permet donc de meilleures contractions.

Le graphe de contraintes, présenté dans la figure 4.3, synthétise cette double décomposition en introduisant une variable intermédiaire $\mathbb{W}_{\mathbf{T}}(t_i)$, à la fois dans un repère cartésien et polaire.

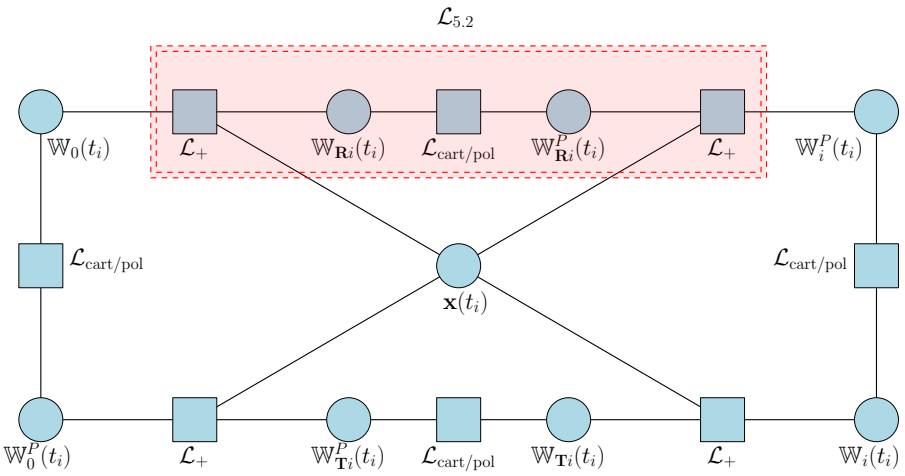


FIGURE 4.3 – Graphe de contraintes résumant la décompositon de la transformation rigide incertaine d'un sous-ensemble réel, d'une part en une rotation puis translation (ligne du haut), d'autre part en translation puis rotation (ligne du bas).

La seconde formulation de la fonction $\mathbf{h}$ sur laquelle nous nous basons est la suivante :

$$\mathbf{h}(\mathbf{x}(t_i), \mathbb{W}_i(t_i)) = \mathbf{R}_{x_3(t_i)} \left(\mathbb{W}_i(t_i) + \begin{pmatrix} x_{\mathcal{R}_i,1}(t_0) \\ x_{\mathcal{R}_i,2}(t_0) \end{pmatrix} \right). \tag{4.22}$$

La rotation utilise le même paramètre que précédemment, mais la translation s'effectue maintenant dans le repère $\mathcal{R}_i$, via la position du robot dans le repère $\mathcal{R}_i$ à l'instant t_0.

En effectuant le changement de repère, nous avons :

$$\begin{pmatrix} x_{\mathcal{R}_i,1}(t_0) \\ x_{\mathcal{R}_i,2}(t_0) \end{pmatrix} = \mathbf{R}_{-x_3(t_i)} \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix}. \tag{4.23}$$

Remarque 21. *Une autre manière d'obtenir ces paramètres est de calculer* $\mathbf{x}_{\mathcal{R}_i}(t_0)$ *comme nous l'avons vu en remarque 17, c'est-à-dire par intégration des vitesses dans le repère* $\mathcal{R}_i$.

Avec cette deuxième décomposition, nous faisons intervenir une nouvelle variable intermédiaire dans deux systèmes de coordonnées, ainsi que trois nouvelles contraintes :

$$
\begin{cases}
\textbf{Variables :} & \mathbb{W}_0(t_i),\ \mathbb{W}_0^P(t_i),\ \mathbb{W}_i(t_i),\ \mathbb{W}_i^P(t_i),\ \mathbb{W}_{\mathbf{R}}(t_i),\ \mathbb{W}_{\mathbf{R}}^P(t_i),\ \mathbb{W}_{\mathbf{T}}(t_i), \\
 & \mathbb{W}_{\mathbf{T}}^P(t_i),\ \mathbf{x}(t_i) \\
\textbf{Domaines :} & [\mathbb{W}_0(t_i)],\ [\mathbb{W}_0^P(t_i)],\ [\mathbb{W}_i(t_i)],\ [\mathbb{W}_i^P(t_i)],\ [\mathbb{W}_{\mathbf{R}}(t_i)],\ [\mathbb{W}_{\mathbf{R}}^P(t_i)],\ [\mathbb{W}_{\mathbf{T}}(t_i)], \\
 & [\mathbb{W}_{\mathbf{T}}^P(t_i)],\ [\mathbf{x}(t_i)] \\
\textbf{Contraintes :} & \\
\quad 1. & \mathbb{W}_0(t_i) \overset{cart/pol}{\longleftrightarrow} \mathbb{W}_0^P(t_i) \\
\quad 2. & \mathbb{W}_i(t_i) \overset{cart/pol}{\longleftrightarrow} \mathbb{W}_i^P(t_i) \\
\quad 3. & \mathbb{W}_{\mathbf{R}}(t_i) \overset{cart/pol}{\longleftrightarrow} \mathbb{W}_{\mathbf{R}}^P(t_i) \\
\quad 4. & \mathbb{W}_{\mathbf{T}}(t_i) \overset{cart/pol}{\longleftrightarrow} \mathbb{W}_{\mathbf{T}}^P(t_i) \\
\quad 5. & \mathbb{W}_{\mathbf{R}}^P(t_i) = \mathbb{W}_i^P(t_i) + \begin{pmatrix} 0 \\ x_3(t_i) \end{pmatrix} \\
\quad 6. & \mathbb{W}_0(t_i) = \mathbb{W}_{\mathbf{R}}(t_i) + \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} \\
\quad 7. & \mathbb{W}_{\mathbf{T}}(t_i) = \mathbb{W}_i(t_i) + \mathbf{R}_{-x_3(t_i)} \begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} \\
\quad 8. & \mathbb{W}_0^P(t_i) = \mathbb{W}_{\mathbf{T}}^P(t_i) + \begin{pmatrix} 0 \\ x_3(t_i) \end{pmatrix}
\end{cases}
\tag{4.24}
$$

Afin de ne pas surcharger la modélisation du problème, nous allons considérer le calcul du paramètre de la translation de la contrainte 7 à partir de la pose $\mathbf{x}(t_i)$, mais dans les expérimentations, ce calcul sera effectué à partir de variables $\mathbf{x}_{\mathcal{R}_j}(t_0)$.

4.2.3 Contrainte d'estimation d'état par les mesures

Nous considérons maintenant les contraintes 6 du système (4.13) :

$$6. \quad \mathbf{h}(\mathbf{x}(t_i), \mathbb{Z}_i(t_i)) \subset \partial\mathbb{M}_0 \qquad 0 \leq i \leq m-1.$$

Chacune de ces contraintes permet de contracter la trajectoire aux instants t_i en éliminant les poses incohérentes, une pose étant incohérente lorsque la contrainte est violée.

En particulier, nous savons qu'avec les contraintes 5 nous obtenons de l'information sur la partie sans obstacles $\overline{\mathbb{M}_0}$. Ainsi, pour un ensemble d'états $[\mathbf{x}]$ et une zone d'impact $[\mathbf{z}]$ issue du nuage de points $[\mathbb{Z}_i(t_i)]$, nous définissons un test d'exclusion :

$$\mathbf{h}([\mathbf{x}], [\mathbf{z}]) \subset \overline{\mathbb{M}_0} \implies \mathbf{x}(t_i) \notin [\mathbf{x}]. \tag{4.25}$$

La condition du test d'exclusion (4.25) implique le changement de repère d'une boîte avec des paramètres incertains, mais se calcule facilement avec des fonctions d'inclusion, même si cette méthode est sujette au *wrapping effect* (voir page 19).

Dans la suite, nous supposons alors que nous pouvons calculer les quantités suivantes :

$$[\mathbf{h}]([\mathbf{x}](t_i), [\mathbb{Z}_i(t_i)]_k), \tag{4.26}$$

où la fonction d'inclusion $[\mathbf{h}]$ est obtenue par l'analyse intervalle classique, comme montré dans le chapitre 2 (voir page 17), et $[\mathbb{Z}_i(t_i)]_k$ représente le domaine de la mesure $\mathbf{z}_k(t_i) = (\mathbb{Z}_i(t_i))_k$.

Nous verrons dans le chapitre suivant comment implémenter un contracteur pour ces contraintes, basé sur le test d'exclusion (4.25) d'une sous-boîte $[\mathbf{x}]$ du domaine $[\mathbf{x}](t_i)$.

4.3 Graphe de contraintes après décomposition

Maintenant que nous avons vu toutes les briques nécessaires à la résolution du problème, nous les résumons en un seul graphe de contraintes dans la figure 4.4. Celui-ci concerne un cas avec trois pas de temps et sans mesures à l'instant t_0.

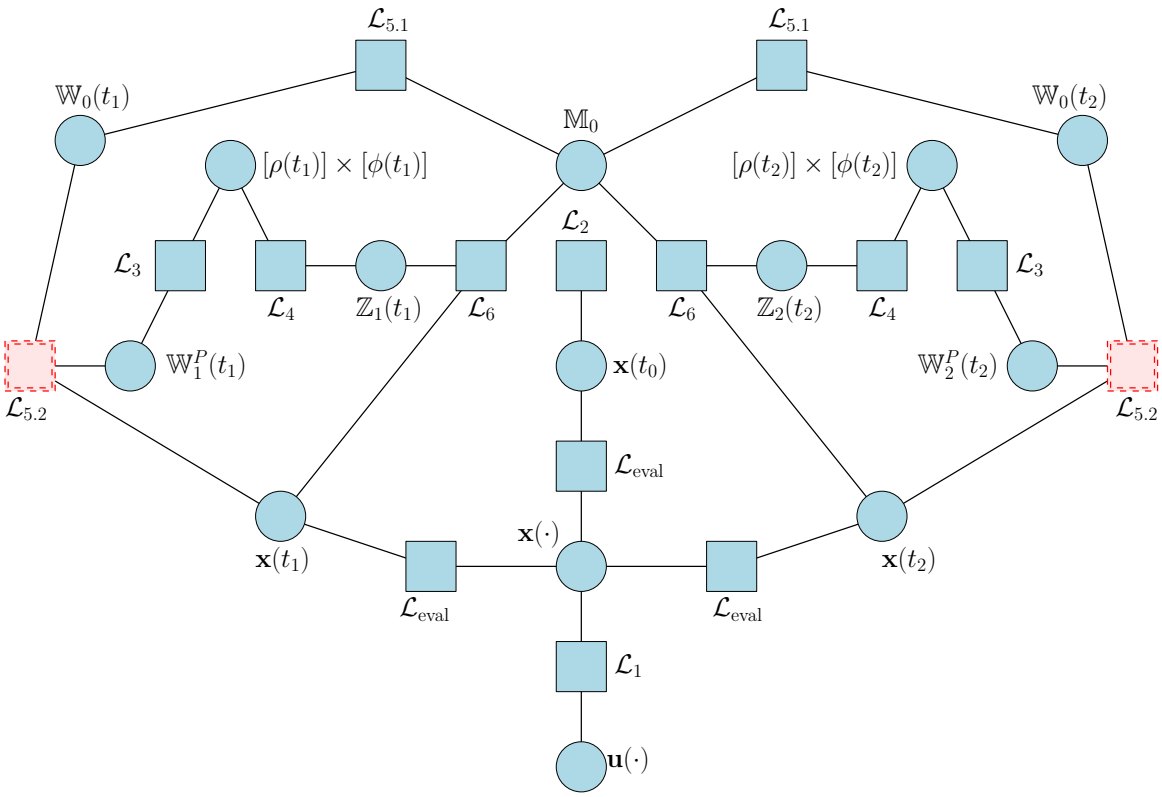


FIGURE 4.4 – Graphe de contraintes du système (4.13). Les contraintes $\mathcal{L}_{5.2}$ peuvent être décomposées selon le graphe de contraintes de la figure 4.2 pour la première décomposition évoquée, ou selon le graphe de contraintes de la figure 4.3 pour la double décomposition. Les contraintes $\mathcal{L}_{\text{eval}}$ sont des contraintes implicites, et font le lien entre la trajectoire $\mathbf{x}(\cdot)$ et ses évaluations aux instants t_0, t_1, t_2.

Ce graphe de contraintes permet d'illustrer la structure du problème de SLAM que nous étudions dans cette contribution, et cette structure est définitive puisqu'au cours du chapitre suivant, aucune

décomposition supplémentaire ne sera nécessaire. La présence de cycles dans le graphe illustre des chaînes de contractions cycliques potentielles. En particulier, le problème de SLAM est connu pour ses cycles de contraintes entre variables : mieux estimer la trajectoire à l'instant t_1, permet d'affiner la cartographie (contrainte 5.2, gauche), qui donne à son tour de meilleures contractions de la trajectoire à l'instant t_2 (contrainte 6, droite) qui se repropage à l'instant t_1 (contrainte 1).

Au chapitre suivant, nous proposons des contracteurs associés à chacune des contraintes apparaissant dans le graphe de contraintes de la figure 4.4, avec leur implémentation. Nous montrons aussi comment piloter tous ces outils de résolution, afin de les appliquer sur différentes instances du problème de SLAM (4.13).

Chapitre 5

Pavages épais réguliers comme représentation des ensembles épais

Plan du chapitre

L'objectif de ce chapitre est d'introduire les pavages épais réguliers, et de voir comment s'en servir comme domaine de variables ensemblistes dans un CSP, et plus précisément comme implémentation d'un ensemble épais. Nous rappelons qu'un ensemble épais $[\mathbb{X}] = [\mathbb{X}^-, \mathbb{X}^+]$ forme une tripartition $(\mathbb{X}^-, \mathbb{X}^+ \setminus \mathbb{X}^-, \overline{\mathbb{X}^+})$ de l'espace. Nous nommons respectivement ces trois ensembles l'**intérieur**, la **pénombre** [20] et l'**extérieur**.

Remarque 22. *Ici, l'extérieur consiste bien en l'ensemble des points en dehors de l'ensemble épais, et ne correspond donc pas à une approximation extérieure.*

Après avoir motivé ce choix (section 5.1), nous verrons comment utiliser un opérateur équivalent aux ensembles épais pour construire un pavage épais régulier (section 5.2), et le manipuler de manière garantie en développant une algèbre (section 5.3). Enfin, nous montrerons comment procéder à une contraction d'un pavage épais en l'appliquant directement à l'une des contraintes du problème de cette contribution (section 5.4).

5.1 Motivation du choix du pavage épais régulier

Dans ce manuscrit, un pavage est une structure de données arborescente, où chaque nœud correspond à une boîte et une classification de celle-ci. Nous avons déjà introduit les pavages intervalles (voir

page 44). Ceux-ci impliquent des contractions, et possèdent une structure dite irrégulière.

Dans la suite de ce chapitre, nous allons nous intéresser aux pavages possédant une structure **régulière**.

Définition 15. *Un pavage est dit* ***régulier*** *quand il est associé à un algorithme de bissection sans contraction.*

Comparaison entre pavages réguliers et irréguliers Ces différentes structures ont des avantages différents. Pour les pavages réguliers [57], leur définition repose sur une classification par des tests booléens que nous développons dans la section 5.2, qui sont des méthodes plus simples à définir que les méthodes de contraction des pavages irréguliers. La structure des pavages réguliers est aussi déterministe, et donne la propriété suivante : si deux pavages réguliers ont la même racine, alors il existe une correspondance nœud à nœud directe entre ces pavages. Cette propriété rend certains calculs entre pavages réguliers faciles et rapides, comme nous le verrons dans la suite. De leur côté, les pavages irréguliers sont plus efficaces, puisque leurs méthodes de contractions amènent une information plus riche que les tests des pavages réguliers. Ils nécessitent alors moins de boîtes pour atteindre le même niveau de précision qu'un pavage régulier.

Les pavages intervalles, qu'ils soient réguliers ou irréguliers, se révèlent inadaptés pour traiter les ensembles épais : afin d'être le plus précis possible, le pavage doit bissecter toute boîte incertaine, comme illustré dans la figure 5.1. Ainsi, lorsque l'incertitude $\mathbb{X}^+ \setminus \mathbb{X}^-$ est trop importante, le nombre de boîtes nécessaire pour couvrir l'incertitude explose.

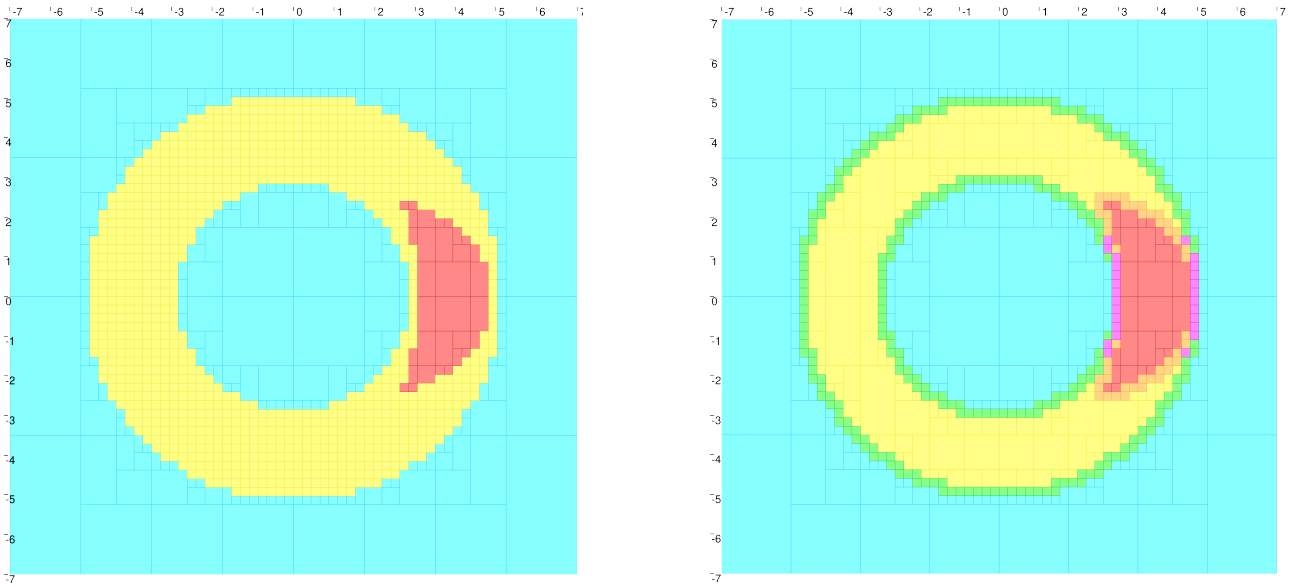


FIGURE 5.1 – À gauche : utilisation d'un pavage intervalle *classique* régulier pour approcher un ensemble épais, avec 3 classes. Lorsque l'incertitude de celui-ci est importante, il faut un nombre important de boîtes (ici en jaune) pour couvrir cette incertitude. À droite : utilisation d'un pavage *épais* régulier avec 6 classes pour approcher le même ensemble épais : le nombre de boîtes pour obtenir la même précision est réduit.

Dans la littérature, les pavages existent avec trois classes : boîtes intérieures, incertaines et extérieures, celles-ci étant déterminées par l'inclusion du nœud dans l'ensemble intérieur, incertain ou extérieur correspondant de la partition.

Les boîtes non classées sont alors bissectées, jusqu'à être classées ou jusqu'à une certaine précision.

Pour traiter le problème de cette contribution, nous allons utiliser une classification plus complète, avec six classes. Ainsi, aux trois classes précédentes, nous rajoutons trois classes frontières afin de mieux caractériser les boîtes incertaines. La figure 5.1 illustre cette classification, où une couleur correspond une classe différente. Pour résumer, nous avons les classes suivantes :

— intérieure (rouge),
— extérieure (bleue),
— pénombre (jaune),
— frontière intérieure, ou pénombre et intérieure (orange),
— frontière extérieure, ou pénombre et extérieure (vert),
— double frontière, ou intérieure et extérieure (et donc prénombre) (violette)

Nous appelons **étiquettes** ces différentes classes.

Afin de bien définir les pavages, nous aurons besoin de classer les boîtes de celui-ci. Cette classification sera formalisée comme une fonction (**fonction d'étiquetage**) capable de classer n'importe quelle boîte. Nous verrons qu'il est possible de composer ces fonctions pour reproduire des calculs sur ensembles épais, par exemple l'intersection $\cap$ que nous rappelons ici :

$$[\mathbb{X}] \cap [\mathbb{Y}] = [\mathbb{X}^- \cap \mathbb{Y}^-, \mathbb{X}^+ \cap \mathbb{Y}^+]. \tag{5.1}$$

La figure 5.2 illustre le pavage épais résultant de cette opération, calculé à partir d'une composition de fonctions d'étiquetage.

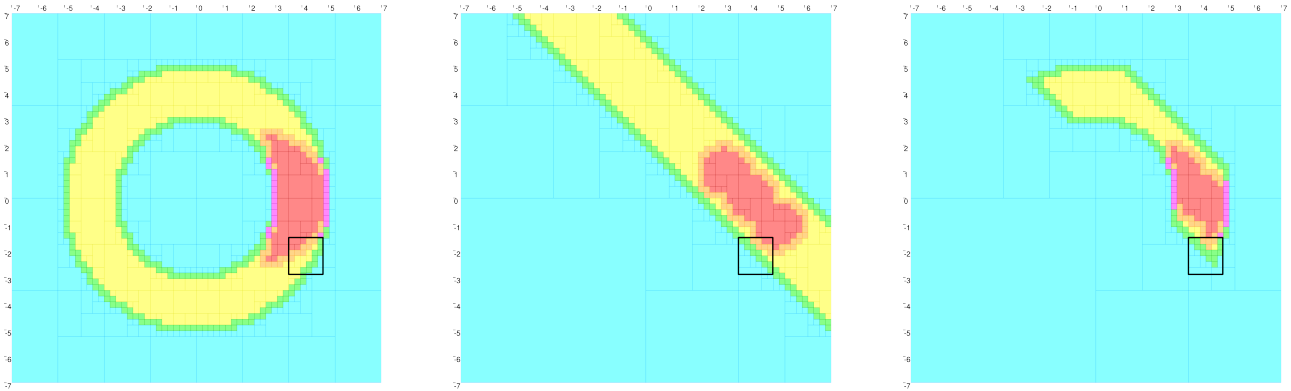


FIGURE 5.2 – De gauche à droite : pavage épais régulier de $[\mathbb{X}]$, pavage épais régulier de $[\mathbb{Y}]$, pavage épais régulier de $[\mathbb{X}] \cap [\mathbb{Y}]$, obtenu par une intersection de fonctions d'étiquetage. Dans la figure 5.3, nous zoomons sur une partie, indiquée par une boîte noire, afin de montrer les particularités de ce calcul.

Pour faire cette opération, nous avons adapté l'intersection des ensembles épais aux classifications des boîtes du pavage. Cependant, nous devons nécessairement autoriser les classifications incertaines pour être cohérent : la figure 5.3 montre un zoom sur le calcul précédent. La boîte $[\mathbf{x}]$ étant en vert sur les deux pavages en entrée, elle contient la frontière extérieure (trait noir fin) des deux ensembles épais en entrée. Cependant, nous ne pouvons pas assurer que $[\mathbf{x}]$ contient la nouvelle frontière extérieure. Sur le résultat, cette boîte a alors une classification incertaine, illustrée par des hachures vertes et bleues, indiquant qu'elle peut être soit verte (frontière extérieure), soit bleue (extérieure).

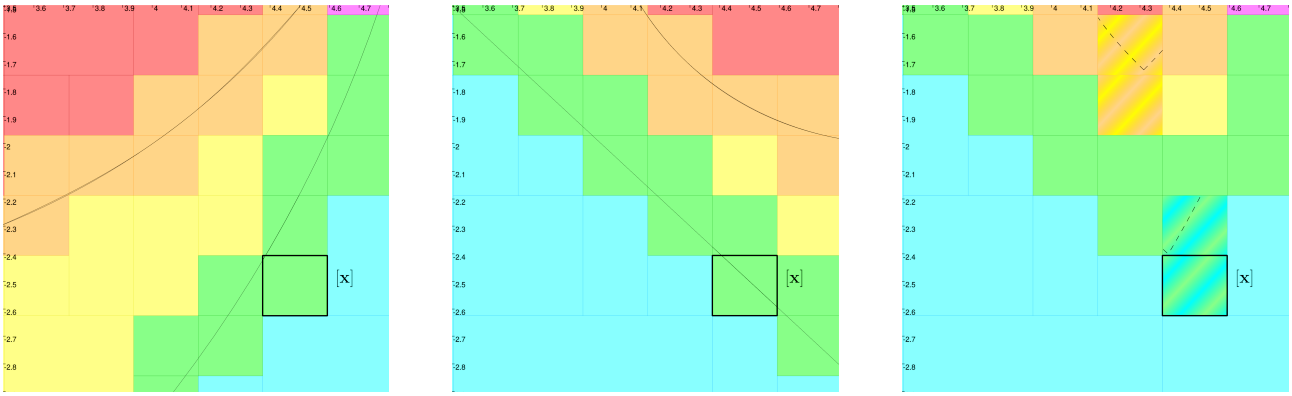


FIGURE 5.3 – Zoom sur la figure précédente. Le calcul de l'intersection entre deux pavages épais est incertain, avec 4 boîtes (hachurées) qui n'ont pas pu être classées précisément. Par exemple, la boîte $[\mathbf{x}]$ est frontière extérieure pour les deux ensembles épais en entrée, mais nous ne pouvons pas savoir si elle contient la frontière résultant de l'intersection (en pointillé), et donc si $[\mathbf{x}]$ possède au moins un point dans la pénombre ou non.

Avec cette méthode, chaque opération ensembliste doit être adaptée aux classifications incertaines. Pour simplifier la démarche, nous formalisons ces classifications incertaines comme des **intervalles d'étiquettes**, puis nous développons une **algèbre des intervalles d'étiquettes** en section 5.3. Cette algèbre nous permet alors de manipuler facilement les ensembles épais, en calculant automatiquement toute composition d'opérations de l'algèbre.

Enfin, nous montrons comment contracter un pavage épais en section 5.4 : cela revient à mieux classer les boîtes intersectant la pénombre, idéalement en des boîtes intérieures et/ou extérieures.

5.2 Fonction d'étiquetage : un opérateur équivalent à un ensemble épais

Lorsque nous avons introduit les contracteurs, nous avons aussi montré qu'ils sont équivalent à des sous-ensembles de $\mathbb{R}^n$ (voir page 21), ce qui justifie le paradigme de **programmation par contracteur** (*contractor programming* [8]).

Pour ce qui est des ensembles épais, nous définissons aussi un nouvel opérateur, et nous montrons qu'il est équivalent à un ensemble épais. Cet opérateur prend une boîte $[\mathbf{x}]$ en entrée, et retourne le résultat de trois tests d'intersection, rassemblé en une **étiquette** :

Définition 16. *Pour un ensemble épais* $[\mathbb{X}]$ *et une boîte* $[\mathbf{x}]$*, l'étiquette de* $[\mathbf{x}]$ *pour* $[\mathbb{X}]$ *est un triplet de booléens* $(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u})$ *où :*

- $\mathsf{x_i}$ *vaut* $\top$ *si* $[\mathbf{x}] \cap \mathbb{X}^- \neq \emptyset$*, c'est-à-dire contient au moins un point intérieur, et* $\bot$ *sinon,*
- $\mathsf{x_o}$ *vaut* $\top$ *si* $[\mathbf{x}] \cap \overline{\mathbb{X}^+} \neq \emptyset$*, c'est-à-dire contient au moins un point extérieur, et* $\bot$ *sinon,*
- $\mathsf{x_u}$ *vaut* $\top$ *si* $[\mathbf{x}] \cap (\mathbb{X}^+ \setminus \mathbb{X}^-) \neq \emptyset$*, c'est-à-dire contient au moins un point dans la pénombre, et* $\bot$ *sinon.*

Remarque 23. *Comme les bornes d'un ensemble épais respectent une propriété d'inclusion* ($\mathbb{X}^- \subset \mathbb{X}^+$) *et d'exclusion (aucun point n'est à la fois intérieur et extérieur, c'est-à-dire* $\mathbb{X}^- \cap \overline{\mathbb{X}^+} = \emptyset$*), alors les étiquettes aussi. En particulier, deux étiquettes sont impossibles :*

— $(\bot, \bot, \bot)$, *puisque l'ensemble épais est une tripartition de l'espace,*
— $(\top, \top, \bot)$, *puisque la frontière est toujours considérée comme étant dans la pénombre.*

Cet opérateur est appelé une **fonction d'étiquetage**.

Définition 17. *Une* ***fonction d'étiquetage*** *pour un ensemble épais* $[\mathbb{X}]$*, notée* $\mathbf{label}_{[\mathbb{X}]}$*, est une fonction qui permet d'étiqueter n'importe quelle boîte selon* $[\mathbb{X}]$ *:*

$$\mathbf{label}_{[\mathbb{X}]} : \begin{cases} \mathbb{IR}^n \to \{\bot, \top\}^3 \\ [\mathbf{x}] \mapsto (\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u}) \end{cases} \tag{5.2}$$

où $(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u})$ *est l'étiquette de* $[\mathbf{x}]$ *par rapport à* $[\mathbb{X}]$*.*

La figure 5.4 illustre l'équivalence entre un ensemble épais et sa fonction d'étiquetage, et nous notons alors $[\mathbb{X}] \sim \mathbf{label}_{[\mathbb{X}]}$. Plus formellement, il y a bien équivalence puisqu'il est possible de déterminer une fonction d'étiquetage à partir d'un ensemble épais par définition, mais il est aussi possible de déterminer les bornes d'un ensemble épais à partir d'une fonction d'étiquetage :

$$\mathbb{X}^- = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{label}_{[\mathbb{X}]}(\{\mathbf{x}\}) = (\top, \cdot, \cdot)\}, \tag{5.3}$$
$$\mathbb{X}^+ = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{label}_{[\mathbb{X}]}(\{\mathbf{x}\}) = (\cdot, \bot, \cdot)\}. \tag{5.4}$$

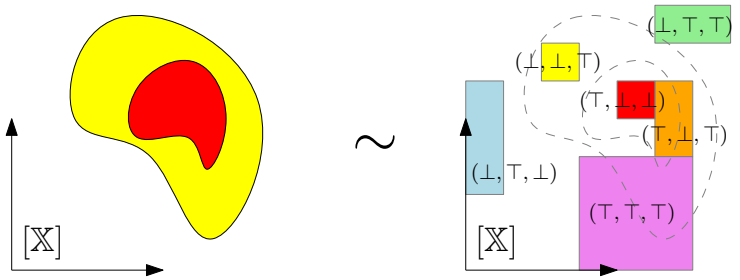


FIGURE 5.4 – Illustration de la fonction d'étiquetage d'un ensemble épais : les six étiquettes possibles sont représentées ainsi que la couleur correspondante.

En réalité, les trois tests d'intersection nécessaires pour définir une étiquette sont imprécis, ce qui est dû à des imprécisions numériques par exemple. En accord avec le paradigme intervalle, nous bornons cette imprécision par des intervalles, au moyen d'un **intervalle de booléens** [58].

Définition 18. *Un* ***intervalle de booléens*** *est un élément à valeur dans* $\mathbb{IB} = \{\bot, \top, [\bot, \top]\}$*. Les opérations logiques booléennes s'étendent aux intervalles de booléens. Si* $[\mathsf{b}]$ *est un intervalle de booléens, alors :*

$$\begin{cases} \bot \wedge [\mathsf{b}] = \bot, \\ \top \wedge [\mathsf{b}] = [\mathsf{b}], \\ \bot \vee [\mathsf{b}] = [\mathsf{b}], \\ \top \vee [\mathsf{b}] = \top, \\ [\mathsf{b}] \vee [\mathsf{b}] = [\mathsf{b}] \wedge [\mathsf{b}] = [\mathsf{b}]. \end{cases} \tag{5.5}$$

Quant à la négation, elle se définit ainsi :

$$\neg\top = \bot; \quad \neg\bot = \top; \quad \neg[\bot, \top] = [\bot, \top]. \tag{5.6}$$

Tout comme les calculs sur intervalles de réels, les opérations entre intervalles de booléens permettent de garantir les calculs sur booléens, car pour deux booléens $\mathsf{b}_1 \in [\mathsf{b}_1]$, $\mathsf{b}_2 \in [\mathsf{b}_2]$ *nous avons bien :*

$$\begin{aligned} \mathsf{b}_1 \wedge \mathsf{b}_2 &\in [\mathsf{b}_1] \wedge [\mathsf{b}_2], \\ \mathsf{b}_1 \vee \mathsf{b}_2 &\in [\mathsf{b}_1] \vee [\mathsf{b}_2], \\ \neg\mathsf{b}_1 &\in \neg[\mathsf{b}_1]. \end{aligned}$$

Pour deux intervalles de booléens $[\mathsf{b}_1], [\mathsf{b}_2]$*, nous définissons une nouvelle opération* $[\wedge]$ *:*

$$[\mathsf{b}_1]\,[\wedge]\,[\mathsf{b}_2] = [\bot, \top] \wedge [\mathsf{b}_1] \wedge [\mathsf{b}_2]. \tag{5.7}$$

C'est une opération plus pessimiste que $\wedge$*. En particulier, celle-ci ne peut retourner que* $\bot$ *ou* $[\bot, \top]$*.*

Ainsi, à l'aide de ces intervalles de booléens, il est possible de définir des **intervalles d'étiquettes** qui contiennent l'étiquette réelle, ainsi que des **fonctions d'étiquetage d'inclusion** d'une manière similaire aux fonctions d'inclusion (cf. page 2.22).

Définition 19. *La fonction* $[\mathbf{label}] : \mathbb{IR}^n \to \mathbb{IB}^3$ *est une* ***fonction d'étiquetage d'inclusion*** *pour la fonction d'étiquetage* **label** *si :*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{label}([\mathbf{x}]) \in [\mathbf{label}]([\mathbf{x}]), \tag{5.8}$$

où une étiquette $(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u})$ *appartient à l'****intervalle d'étiquette*** $([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}])$ *si :*

$$\begin{cases} \mathsf{x_i} \in [\mathsf{x_i}] \\ \mathsf{x_o} \in [\mathsf{x_o}] \\ \mathsf{x_u} \in [\mathsf{x_u}] \end{cases} \tag{5.9}$$

Une fonction d'étiquetage d'inclusion effectue alors les trois tests d'intersection pour déterminer une étiquette, mais retournent l'intervalle $[\bot, \top]$ lorsqu'ils ne sont pas capables de conclure.

Remarque 24. *Il faut bien distinguer ici deux types d'incertitudes, comme énoncé dans [58]. L'incertitude de cette fonction d'étiquetage d'inclusion est une incertitude sur les bornes de l'ensemble épais. Il est possible de réduire cet intervalle d'étiquette à l'aide de bissections, comme nous le ferons avec notre algorithme* `label_to_paving`*, voir algorithme 1*

L'incertitude de l'ensemble épais que nous appelons ***pénombre****, c'est-à-dire l'ensemble* $\mathbb{X}^+ \setminus \mathbb{X}^-$*, sera quant à elle réduite par des contracteurs, selon les contraintes du problème, et est le sujet de la section 5.4.*

5.2.1 Passage d'une fonction d'étiquetage d'inclusion à un pavage épais régulier

L'objectif de ce chapitre est l'utilisation du pavage épais régulier comme domaine pour des variables ensemblistes dans un CSP. Comme nous allons manipuler les fonctions d'étiquetage d'inclusion, nous montrons comment construire un pavage épais régulier à partir d'une telle fonction.

Pour présenter l'algorithme utilisé, nous commençons par définir une classe `Node` pour les nœuds du pavage épais régulier, en accord avec la catégorisation selon un intervalle d'étiquette.

class Node {
  $\mathbb{IR}^n$: [**x**] // boîte englobante du nœud
  $\mathbb{IB}^3$: ([x$_i$], [x$_o$], [x$_u$]) // intervalle d'étiquette du nœud
  List(Node) : Children // liste des fils (éventuellement vide)

  Bool : isBisectible(ε) // vrai si la boîte est bisécable à la précision ε
  (Node, Node) : bisect() // renvoie deux nœuds issus de la bissection
  void : merge() // fusionne l'étiquette des fils
  void : simplify() // vide Children lorsque l'étiquette a une composante $\top$, les autres $\bot$
}

La méthode isBisectible(ε) décide si un nœud a intérêt à être bissecté afin d'en tirer l'information maximale étant donnée la fonction d'étiquetage d'inclusion. Un nœud est bisécable quand sa liste de fils est vide, que son intervalle d'étiquette a plusieurs composantes $\top$ ou au moins une composante $[\bot, \top]$, et que sa taille est inférieure au paramètre de précision ε.

L'algorithme 1 est un algorithme récursif qui construit, modifie ou actualise un pavage épais régulier à partir de son nœud racine, d'une fonction d'étiquetage d'inclusion [**label**] et d'un critère de précision ε à l'aide d'une fonction `label_to_paving`.

Algorithme 1 : Pavage de l'espace selon une fonction d'étiquetage d'inclusion

Entrées : Racine, [**label**], ε
Sorties : Actualisation du pavage épais selon [**label**]

```
1  Fonction label_to_paving(N, [label], ε) :
2  |  N.([x_i], [x_o], [x_u]) ← [label](N.[x])
3  |  simplify()
4  |  si isBisectible(ε) alors
5  |  |  N.Children ← bisect()
6  |  fin
7  |  pour chaque M ∈ N.Children faire
8  |  |  M.label_to_paving([label], ε) ;
9  |  fin
10 |  merge()
11 |  simplify()
12 label_to_paving(Racine, [label], ε)
```

Nous verrons alors que la contraction de ce domaine peut se faire de manière incrémentale, en ré-étiquetant les nœuds du pavage, et ceci avec l'algorithme 1.

5.2.2 Passage d'un pavage épais régulier à une fonction d'étiquetage d'inclusion

Nous avons montré comment passer d'une fonction d'étiquetage d'inclusion à un pavage épais régulier. Comme nous allons manipuler ces fonctions, nous montrons comment déduire une fonction d'étiquetage d'inclusion pour n'importe quel pavage épais régulier, au moyen d'une méthode `paving_to_label`.

Si une boîte correspond à l'un des nœuds du pavage épais régulier, il est facile d'obtenir son intervalle

d'étiquette. Pour une boîte quelconque cependant, l'obtention de son intervalle d'étiquette n'est pas directe, mais reste possible.

Cette étape est intéressante puisqu'elle permet de profiter des bissections du pavage épais, qui permet d'améliorer certaines incertitudes sur les intervalles d'étiquettes. Aussi, la fonction d'étiquetage d'inclusion calculée à partir d'un pavage épais régulier a une complexité linéaire en la taille du pavage épais régulier, et donc potentiellement une meilleure complexité que la fonction initiale, si tant est qu'elle soit disponible.

Pour y arriver, nous procédons de la manière suivante. Nous considérons chaque feuille $F = ([\mathbf{f}], ([\mathsf{f_i}], [\mathsf{f_o}], [\mathsf{f_u}]), \emptyset)$, telle que $[\mathbf{x}] \cap [\mathbf{f}] \neq \emptyset$. Nous distinguons les cas de figures suivants :

(i) si $[\mathbf{f}] \subset [\mathbf{x}]$,
(ii) si $[\mathbf{x}] \cap [\mathbf{f}] \neq \emptyset$ et un seul intervalle de booléens de $([\mathsf{f_i}], [\mathsf{f_o}], [\mathsf{f_u}])$ vaut $\top$ et les autres $\bot$,
(iii) sinon.

Selon ces cas, nous pouvons en déduire un intervalle d'étiquette $([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}])$, qui contient l'étiquette réelle $(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u})$ de $[\mathbf{x}]$:

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) = \bigvee_{\substack{\text{Pour tout } F \\ \text{vérifiant (i) ou (ii)}}} ([\mathsf{f_i}], [\mathsf{f_o}], [\mathsf{f_u}]) \vee \bigvee_{\substack{\text{Pour tout } F \\ \text{vérifiant (iii)}}} [\bot, \top]^3 \wedge ([\mathsf{f_i}], [\mathsf{f_o}], [\mathsf{f_u}]), \tag{5.10}$$

en distribuant les opérations logiques $\vee$ et $\wedge$ sur chaque intervalle de booléens.

En effet dans les cas (i) et (ii), nous pouvons bien en déduire que $[\mathbf{x}]$ contient au moins l'étiquette réelle $(\mathsf{f_i}, \mathsf{f_o}, \mathsf{f_u})$. Dans le cas (iii), il est impossible de trancher de manière certaine quelles parties (intérieure, extérieure, pénombre) intersectent bien $[\mathbf{x}]$. Nous ajoutons donc, de manière incertaine, ces étiquettes au résultat.

Les pertes d'information dûes au cas (iii) sont inévitables en utilisant des pavages épais réguliers, puisqu'ils ne représentent numériquement l'ensemble épais que jusqu'à une certaine précision.

Nous appelons `paving_to_label` la méthode réalisant ce calcul qui, à partir d'un pavage épais régulier pour $[\mathbb{X}]$, donne la fonction qui à une boîte $[\mathbf{x}]$ retourne un intervalle d'étiquette garantie de contenir l'étiquette de $[\mathbf{x}]$ pour $[\mathbb{X}]$.

5.3 Définition d'une algèbre des intervalles d'étiquettes pour la manipulation des ensembles épais

Maintenant que nous avons une équivalence entre ensemble épais et fonction d'étiquetage, nous voulons étendre la manipulation des ensembles épais aux fonctions d'étiquetage. Dans cette section, nous montrons comment garantir certaines opérations de $\mathbb{IP}(\mathbb{R}^n)$ avec des fonctions d'étiquetage d'inclusion.

Proposition 1. *Pour toute opération binaire $\diamond \in \{\sqcap, \cap, \cup, \cap^+, \cup^-\}$ sur ensembles épais, telles que décrites dans la sous-section 2.5.2 (voir page 31), et pour deux ensembles épais $[\mathbb{X}]$ et $[\mathbb{Y}]$, nous définissons les mêmes opérations sur les intervalles d'étiquettes qui vérifient alors :*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathbf{label}_{[\mathbb{X}]\diamond[\mathbb{Y}]}([\mathbf{x}]) \in [\mathbf{label}]_{[\mathbb{X}]}([\mathbf{x}]) \diamond [\mathbf{label}]_{[\mathbb{Y}]}([\mathbf{x}]). \tag{5.11}$$

Pour le complémentaire, nous définissons aussi l'opération unaire sur intervalle d'étiquette qui vérifie :

$$\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathbf{label}_{\overline{[\mathbb{X}]}}([\mathbf{x}]) \in \overline{[\mathbf{label}]_{[\mathbb{X}]}([\mathbf{x}])}. \tag{5.12}$$

Pour deux intervalles d'étiquettes $([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}])$ et $([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}])$, les opérations utilisées dans la proposition 1 sont définies de la manière suivante :

— le complémentaire :

$$\overline{([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}])} = ([\mathsf{x_o}], [\mathsf{x_i}], [\mathsf{x_u}]), \tag{5.13}$$

— l'intersection $\sqcap$:

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) \sqcap ([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}]) = ([\mathsf{x_i}] \vee [\mathsf{y_i}], [\mathsf{x_o}] \vee [\mathsf{y_o}], [\mathsf{x_u}]\,[\wedge]\,[\mathsf{y_u}]), \tag{5.14}$$

— l'intersection $\cap$:

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) \cap ([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}]) = ([\mathsf{x_i}]\,[\wedge]\,[\mathsf{y_i}], [\mathsf{x_o}] \vee [\mathsf{y_o}], ([\mathsf{x_u}]\,[\wedge]\,[\mathsf{y_u}]) \vee ([\mathsf{x_u}]\,[\wedge]\,[\mathsf{y_i}]) \vee ([\mathsf{x_i}]\,[\wedge]\,[\mathsf{y_u}]))\,, \tag{5.15}$$

— l'union :

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) \cup ([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}]) = ([\mathsf{x_i}] \vee [\mathsf{y_i}], [\mathsf{x_o}]\,[\wedge]\,[\mathsf{y_o}], ([\mathsf{x_u}]\,[\wedge]\,[\mathsf{y_u}]) \vee ([\mathsf{x_u}]\,[\wedge]\,[\mathsf{y_o}]) \vee ([\mathsf{x_o}]\,[\wedge]\,[\mathsf{y_u}]))\,, \tag{5.16}$$

— l'intersection des bornes supérieures :

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) \cap^{+} ([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}]) = ([\mathsf{x_i}], [\mathsf{x_o}] \vee [\mathsf{y_o}], [\mathsf{x_u}]\,[\wedge]\,([\mathsf{y_u}] \vee [\mathsf{y_i}]))\,, \tag{5.17}$$

— l'union des bornes inférieures :

$$([\mathsf{x_i}], [\mathsf{x_o}], [\mathsf{x_u}]) \cup^{-} ([\mathsf{y_i}], [\mathsf{y_o}], [\mathsf{y_u}]) = ([\mathsf{x_i}] \vee [\mathsf{y_i}], [\mathsf{x_o}], [\mathsf{x_u}]\,[\wedge]\,([\mathsf{y_u}] \vee [\mathsf{y_o}]))\,, \tag{5.18}$$

où l'opération $[\wedge]$ est définie dans la définition 18.

Lemme 1. *Pour deux sous-ensembles $\mathbb{X}$ et $\mathbb{Y}$ de $\mathbb{R}^n$ et une boîte $[\mathbf{x}]$ dans $\mathbb{IR}^n$, nous avons :*

$$[\mathbf{x}] \cap (\mathbb{X} \cup \mathbb{Y}) \neq \emptyset \iff (([\mathbf{x}] \cap \mathbb{X}) \neq \emptyset) \vee (([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset) \tag{5.19}$$

$$[\mathbf{x}] \cap (\mathbb{X} \cap \mathbb{Y}) \neq \emptyset \in (([\mathbf{x}] \cap \mathbb{X}) \neq \emptyset)\ [\wedge]\ (([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset)\,. \tag{5.20}$$

Preuve du lemme 1. Considérons d'abord un test d'intersection pour une union d'ensembles. Nous avons :

$$\begin{aligned} [\mathbf{x}] \cap (\mathbb{X} \cup \mathbb{Y}) \neq \emptyset &\iff ([\mathbf{x}] \cap \mathbb{X}) \cup ([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset \\ &\iff (([\mathbf{x}] \cap \mathbb{X}) \neq \emptyset) \vee (([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset)\,. \end{aligned}$$

Dans le cas de l'intersection, ce n'est pas aussi simple :

$$\begin{aligned} [\mathbf{x}] \cap (\mathbb{X} \cap \mathbb{Y}) \neq \emptyset &\iff ([\mathbf{x}] \cap \mathbb{X} \cap \mathbb{Y}) \neq \emptyset \\ &\in [\bot, \top] \wedge (([\mathbf{x}] \cap \mathbb{X}) \neq \emptyset) \wedge (([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset)\,. \end{aligned}$$

□

Preuve de la proposition 1 pour $\diamond = \sqcap$. Pour rappel, pour deux ensembles épais $[\mathbb{X}]$ et $[\mathbb{Y}]$, nous avons :

$$[\mathbb{X}] \sqcap [\mathbb{Y}] = [\mathbb{X}^{-} \cup \mathbb{Y}^{-}, \mathbb{X}^{+} \cap \mathbb{Y}^{+}]. \tag{5.21}$$

Calculons maintenant les trois tests d'intersection formant l'étiquette pour une boîte $[\mathbf{x}]$, à l'aide du lemme 1 :

$$\begin{aligned}
[\mathbf{x}] \cap (\mathbb{X}^- \cup \mathbb{Y}^-) \neq \emptyset &\iff ([\mathbf{x}] \cap \mathbb{X}^- \neq \emptyset) \vee ([\mathbf{x}] \cap \mathbb{Y}^- \neq \emptyset) \\
[\mathbf{x}] \cap \overline{(\mathbb{X}^+ \cap \mathbb{Y}^+)} \neq \emptyset &\iff ([\mathbf{x}] \cap \overline{\mathbb{X}^+} \neq \emptyset) \vee ([\mathbf{x}] \cap \overline{\mathbb{Y}^+} \neq \emptyset) \\
[\mathbf{x}] \cap \big((\mathbb{X}^+ \cap \mathbb{Y}^+) \setminus (\mathbb{X}^- \cup \mathbb{Y}^-)\big) \neq \emptyset &\iff [\mathbf{x}] \cap (\mathbb{X}^+ \cap \mathbb{Y}^+) \cap \overline{(\mathbb{X}^- \cup \mathbb{Y}^-)} \neq \emptyset \\
&\iff [\mathbf{x}] \cap \mathbb{X}^+ \cap \mathbb{Y}^+ \cap \overline{\mathbb{X}^-} \cap \overline{\mathbb{Y}^-} \neq \emptyset \\
&\iff [\mathbf{x}] \cap (\mathbb{X}^+ \setminus \mathbb{X}^-) \cap (\mathbb{Y}^+ \setminus \mathbb{Y}^-) \neq \emptyset \\
&\in \big([\mathbf{x}] \cap (\mathbb{X}^+ \setminus \mathbb{X}^-) \neq \emptyset\big) \; [\wedge] \; \big([\mathbf{x}] \cap (\mathbb{Y}^+ \setminus \mathbb{Y}^-) \neq \emptyset\big)
\end{aligned}$$

Ainsi, en notant $[\mathbf{label}]_{[\mathbb{X}]}([\mathbf{x}]) = ([\mathsf{x}_\mathsf{i}], [\mathsf{x}_\mathsf{o}], [\mathsf{x}_\mathsf{u}])$ et $[\mathbf{label}]_{[\mathbb{Y}]}([\mathbf{x}]) = ([\mathsf{y}_\mathsf{i}], [\mathsf{y}_\mathsf{o}], [\mathsf{y}_\mathsf{u}])$, alors :

$$\begin{aligned}
\mathbf{label}_{[\mathbb{X}]\sqcap[\mathbb{Y}]}([\mathbf{x}]) &\in ([\mathsf{x}_\mathsf{i}] \vee [\mathsf{y}_\mathsf{i}], [\mathsf{x}_\mathsf{o}] \vee [\mathsf{y}_\mathsf{o}], [\mathsf{x}_\mathsf{u}] \; [\wedge] \; [\mathsf{y}_\mathsf{u}]) \\
&\in [\mathbf{label}]_{[\mathbb{X}]}([\mathbf{x}]) \sqcap [\mathbf{label}]_{[\mathbb{Y}]}([\mathbf{x}]).
\end{aligned}$$

□

Les autres opérations peuvent aussi êtres prouvées de la même manière.

Manipulation des pavages épais réguliers. Avec cette algèbre, il devient alors facile de manipuler des pavages épais réguliers. En effet pour les pavages réguliers, la position d'un nœud dans l'arbre du pavage suffit à déterminer complètement sa boîte $[\mathbf{x}]$ associée. Ainsi lorsque deux pavages épais ont la même racine, les nœuds peuvent être mis en correspondance. Il suffit alors de faire l'opération sur les intervalles d'étiquettes de ces nœuds.

Amélioration des opérations

Dans le lemme 1, nous avons pu déterminer de manière minimale un test d'intersection non vide d'une union d'ensembles. Cependant pour l'intersection, nous avons du utiliser l'opérateur $[\wedge]$ au prix d'un pessimisme.

Remarque 25. *L'opération* $[\wedge]$ *ne peut retourner que* $[\bot, \top]$ *ou* $\bot$*. Il existe pourtant deux cas de figure dans lesquels il est possible de conclure* $\top$ *pour le test d'intersection :*

$$[\mathbf{x}] \subset \mathbb{X} \wedge (([\mathbf{x}] \cap \mathbb{Y}) \neq \emptyset) \implies [\mathbf{x}] \cap (\mathbb{X} \cap \mathbb{Y}) \neq \emptyset \tag{5.22}$$

$$[\mathbf{x}] \subset \mathbb{Y} \wedge (([\mathbf{x}] \cap \mathbb{X}) \neq \emptyset) \implies [\mathbf{x}] \cap (\mathbb{X} \cap \mathbb{Y}) \neq \emptyset \tag{5.23}$$

Il est possible de connaître ces cas de figure en considérant les intervalles d'étiquettes, puisque les intervalles d'étiquettes $(\top, \bot, \bot)$*,* $(\bot, \top, \bot)$ *ou* $(\bot, \bot, \top)$ *garantissent que la boîte associée est respectivement intérieure, extérieure ou complètement incertaine.*

En se basant sur la remarque 25, il est possible d'améliorer les opérations sur les intervalles d'étiquettes.

Motivé par notre application, nous avons implémenté ces opérations en traitant ces cas particuliers à part. Une autre façon plus générique d'en tenir compte est de faire les tables de vérité de chaque

opération, c'est-à-dire traiter tous les cas de figure. Avec $3^3 = 27$ intervalles d'étiquettes possibles, cela porte à pour $27^2 = 729$ cas de figure pour les opérations binaires, qu'il est néanmoins possible de réduire en ignorant les intervalles d'étiquettes incohérentes ou en prenant en compte la commutativité des opérations.

Remarque 26. *Parmi les intervalles d'étiquettes incohérentes, nous pouvons citer les deux étiquettes* $(\top, \top, \bot)$ *et* $(\bot, \bot, \bot)$*, qui sont impossibles à obtenir. De plus, si l'intervalle d'étiquettes a deux composantes* $\bot$*, la troisième est nécessairement* $\top$*. Ainsi, les intervalles* $([\bot, \top], \bot, \bot)$*,* $(\bot, [\bot, \top], \bot)$ *et* $(\bot, \bot, [\bot, \top])$ *peuvent être directement améliorés en* $(\top, \bot, \bot)$*,* $(\bot, \top, \bot)$*,* $(\bot, \bot, \top)$ *respectivement.*

5.4 Contraction des ensembles épais, des fonctions d'étiquetage, et des pavages épais

Nous avons déjà vu ce qu'était un contracteur d'ensemble épais (voir la définition 14, page 35). Puisque les ensembles épais et les fonctions d'étiquetage sont équivalents, il est possible de définir un **contracteur de fonction d'étiquetage** équivalent à ceux-ci.

Dans ce contexte, les éléments à contracter sont les nœuds, et le résultat de la contraction ne modifie pas la boîte du nœud, mais actualise l'étiquette tout en respectant la contractance et la complétude.

Pour la contractance, nous avons besoin d'une relation d'inclusion, et nous rappelons ici l'inclusion $\sqsubset$ entre ensembles épais, qui nous a permis de définir la contractance des **contracteurs d'ensemble épais** :

$$[\mathbb{X}] \sqsubset [\mathbb{Y}] \iff (\mathbb{Y}^- \subset \mathbb{X}^-) \wedge (\overline{\mathbb{Y}^+} \subset \overline{\mathbb{X}^+}). \tag{5.24}$$

Celle-ci s'étend aux étiquettes, et donc aux fonctions d'étiquetage :

$$\begin{aligned}(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u}) \sqsubset (\mathsf{y_i}, \mathsf{y_o}, \mathsf{y_u}) &\iff (\mathsf{y_i} \implies \mathsf{x_i}) \wedge (\mathsf{y_o} \implies \mathsf{x_o}) \\ &\iff (\mathsf{x_i} \vee \neg \mathsf{y_i}) \wedge (\mathsf{x_o} \vee \neg \mathsf{y_o}),\end{aligned}$$

de sorte que si $[\mathbb{X}] \sqsubset [\mathbb{Y}]$, alors $\mathbf{label}_{[\mathbb{X}]} \sqsubset \mathbf{label}_{[\mathbb{X}]}$.

Définition 20. *Étant donné une contrainte* $\mathcal{L}$*, un* ***contracteur de fonction d'étiquetage*** *est un opérateur de* $\mathcal{F}(\mathbb{IR}^n, \{\bot, \top\}^3)$ *vers* $\mathcal{F}(\mathbb{IR}^n, \{\bot, \top\}^3)$ *vérifiant les deux propriétés suivantes :*

$$\begin{aligned}&\mathcal{C}(\mathbf{label}_{[\mathbb{X}]}) \sqsubset \mathbf{label}_{[\mathbb{X}]} && \text{contractance} \\ (\mathbb{X} \in [\mathbb{X}] \wedge \mathcal{L}(\mathbb{X})) \implies \forall [\mathbf{x}] \in \mathbb{IR}^n, &\begin{cases} \mathcal{C}(\mathbf{label}_{[\mathbb{X}]})([\mathbf{x}]) = (\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u}) \\ \wedge\ \big(\mathsf{x_i} \vee ([\mathbf{x}] \subset \overline{\mathbb{X}})\big) \\ \wedge\ (\mathsf{x_o} \vee ([\mathbf{x}] \subset \mathbb{X})) \end{cases} && \text{complétude}\end{aligned}$$

Aussi, à partir d'un contracteur $\mathcal{C}$ *d'un ensemble épais, nous notons et définissons un contracteur* $\mathcal{C}$ *de fonction d'étiquetage équivalent :*

$$\mathcal{C}\colon \begin{cases} \mathcal{F}(\mathbb{IR}^n, \{\bot, \top\}^3) \to \mathcal{F}(\mathbb{IR}^n, \{\bot, \top\}^3) \\ \mathbf{label}_{[\mathbb{X}]} \mapsto \mathbf{label}_{\mathcal{C}([\mathbb{X}])} \end{cases} \tag{5.25}$$

La condition de complétude vérifie que le pavage n'exclue pas d'ensemble $\mathbb{X}$ vérifiant la contrainte. En effet pour un nœud $([\mathbf{x}], (\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u}), \cdot)$ du pavage, les ensembles possibles sont les ensembles $\mathbb{X}$ vérifiant :

$$\mathsf{x_i} \implies \mathbb{X} \cap [\mathbf{x}] \neq \emptyset \tag{5.26}$$

$$\mathsf{x_o} \implies \overline{\mathbb{X}} \cap [\mathbf{x}] \neq \emptyset \tag{5.27}$$

Ces deux relations permettent de définir la relation de complétude précédente.

Les contraintes du problème de cette contribution consistent en des comparaisons entre ensembles, par des inclusions ou égalités. Nous montrons dans la suite comment faire un contracteur pour ces contraintes.

Contrainte d'inclusion entre ensembles

Considérons d'abord la contrainte $\mathcal{L}_{\subset}$ suivante :

$$\begin{cases} \textbf{Variables :} \ \mathbb{X}, \ \mathbb{Y} \\ \textbf{Domaines :} \ [\mathbb{X}], \ [\mathbb{Y}] \\ \textbf{Contrainte :} \ \mathbb{X} \subset \mathbb{Y} \end{cases} \tag{5.28}$$

Pour cette contrainte, il s'agit d'ajouter l'intérieur de $[\mathbb{X}]$ à $[\mathbb{Y}]$, et l'extérieur de $[\mathbb{Y}]$ à $[\mathbb{X}]$, donc d'actualiser les étiquettes avec la fonction de ré-étiquetage en utilisant les opérateurs $\cap^+$ et $\cup^-$, comme nous l'avons déterminé dans la remarque 11 (page 34) :

$$\mathcal{C}_{\subset}\begin{pmatrix} \mathbf{label}_{[\mathbb{X}]} \\ \mathbf{label}_{[\mathbb{Y}]} \end{pmatrix} = \begin{pmatrix} \mathbf{label}_{[\mathbb{X}] \cap^+ [\mathbb{Y}]} \\ \mathbf{label}_{[\mathbb{Y}] \cup^- [\mathbb{X}]} \end{pmatrix} \in \begin{pmatrix} [\mathbf{label}]_{[\mathbb{X}]} \cap^+ [\mathbf{label}]_{[\mathbb{Y}]} \\ [\mathbf{label}]_{[\mathbb{Y}]} \cup^- [\mathbf{label}]_{[\mathbb{X}]} \end{pmatrix}. \tag{5.29}$$

Contrainte d'égalité entre ensembles

Considérons maintenant la contrainte $\mathcal{L}_{=}$ suivante :

$$\begin{cases} \textbf{Variables :} \ \mathbb{X}, \ \mathbb{Y} \\ \textbf{Domaines :} \ [\mathbb{X}], \ [\mathbb{Y}] \\ \textbf{Contrainte :} \ \mathbb{X} = \mathbb{Y} \end{cases} \tag{5.30}$$

Cette contrainte peut être exprimée au moyen de deux contraintes d'inclusion, mais l'actualisation des étiquettes se fait plus simplement avec l'intersection $\sqcap$:

$$\mathcal{C}_{=}\begin{pmatrix} \mathbf{label}_{[\mathbb{X}]} \\ \mathbf{label}_{[\mathbb{Y}]} \end{pmatrix} = \begin{pmatrix} \mathbf{label}_{[\mathbb{X}] \sqcap [\mathbb{Y}]} \\ \mathbf{label}_{[\mathbb{X}] \sqcap [\mathbb{Y}]} \end{pmatrix} \in \begin{pmatrix} [\mathbf{label}]_{[\mathbb{X}]} \sqcap [\mathbf{label}]_{[\mathbb{Y}]} \\ [\mathbf{label}]_{[\mathbb{X}]} \sqcap [\mathbf{label}]_{[\mathbb{Y}]} \end{pmatrix}. \tag{5.31}$$

Application directe de la méthode à l'une des contraintes du SLAM

Nous pouvons appliquer les concepts de ce formalisme afin de proposer directement un contracteur pour la contrainte 5.1 du problème, que nous rappelons ici :

$$5.1. \quad \mathbb{W}_0(t_i) \subset \overline{\mathbb{M}_0} \qquad 0 \leq i \leq m-1$$

En effet, cette contrainte ne fait intervenir que des opérations et relations que nous avons abordées durant tout le début de cette section. Il est dès lors facile de déterminer un contracteur $\mathcal{C}_{5.1}$ capable de contracter le domaine de $\mathbb{M}_0$:

$$\mathcal{C}_{5.1}\begin{pmatrix}\mathbf{label}_{[\mathbb{W}_0(t_i)]}\\ \mathbf{label}_{[\mathbb{M}_0]}\end{pmatrix} = \begin{pmatrix}\mathbf{label}_{[\mathbb{W}_0(t_i)]}\\ \mathbf{label}_{[\mathbb{M}_0(t_i)]\cap^+\overline{[\mathbb{W}_0(t_i)]}}\end{pmatrix} \in \begin{pmatrix}[\mathbf{label}]_{[\mathbb{W}_0(t_i)]}\\ [\mathbf{label}]_{[\mathbb{M}_0(t_i)]} \cap^+ (\neg[\mathbf{label}]_{[\mathbb{W}_0(t_i)]})\end{pmatrix}. \tag{5.32}$$

Contraction des pavages épais réguliers

Nous venons de voir comment proposer un contracteur de fonction d'étiquetage, et de garantir cette fonction à l'aide d'une fonction d'étiquetage d'inclusion.

De plus, nous avons montré comment obtenir une fonction d'étiquetage d'inclusion à partir d'un pavage, avec la méthode `paving_to_label` (cf. algorithme 1), ainsi que comment calculer ou affiner un pavage avec la fonction `label_to_paving` (cf. sous-section 5.2.2).

La contraction d'un pavage épais s'en déduit alors : pour un contracteur $\mathcal{C}$ et un pavage épais régulier $[\mathbb{X}]$, nous effectuons la suite d'opérations suivante :

1. Détermination d'une fonction d'étiquetage d'inclusion $[\mathbf{label}]$ pour $[\mathbb{X}]$ avec `paving_to_label`,
2. Contraction de cette fonction par $\mathcal{C}([\mathbf{label}])$,
3. Actualisation du pavage de $[\mathbb{X}]$ avec `label_to_paving` et la fonction d'étiquetage d'inclusion contractée.

Nous avons désormais les outils suffisants pour proposer des contracteurs associés à chaque contrainte du graphe de contraintes vu en figure 4.4. Ceux-ci sont décrits dans le chapitre suivant.

Chapitre 6

Implémentation et expérimentation d'un problème SLAM modélisé par un CSP

Plan du chapitre

Dans ce chapitre, nous commençons par décrire des contracteurs associés à chaque contrainte du problème, tel qu'illustré par le graphe de contraintes de la figure 4.4. Ensuite, nous les mettons en œuvre afin de résoudre différentes instances du problème. Nous faisons varier différents paramètres de résolution afin d'évaluer leur impact sur la qualité de la sortie, c'est-à-dire la taille de l'ensemble obtenu, contenant la solution réelle.

6.1 Contracteur de la contrainte de cartographie locale

Rappelons les contraintes 3 du système (4.13) :

$$3. \quad \mathbb{W}_i(t_i) \supset \bigcup_{k \in K} [0, \rho_k^-(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1$$

Cette contrainte fait intervenir des unions et une inclusion, un contracteur $\mathcal{C}_3$ pouvant contracter le domaine de $\mathbb{W}_i(t_i)$ est facile à obtenir :

$$\mathcal{C}_3\left(\mathbf{label}_{[\mathbb{W}_0(t_i)]}\right) = \mathbf{label}_{[\mathbb{W}_0(t_i)]} \cap^- \left(\bigcup_{k \in K} \mathbf{label}_{[0, \rho_k^-(t_i)] \times [\phi_k(t_i)]} \right), \tag{6.1}$$

où la fonction d'étiquetage $\mathbf{label}_{[0,\rho_k^-(t_i)]\times[\phi_k(t_i)]}$ est issue des paramètres du problème.

Les contraintes 4, décrivant les obstacles ayant déclenché une mesure capteur, sont rappelées ici :

$$4. \quad (\mathbb{Z}_i(t_i))_k \in [\rho_k(t_i)] \times [\phi_k(t_i)] \qquad 0 \leq i \leq m-1,\, k \in K$$

Le contracteur $\mathcal{C}_4$ suivant permet de contracter le domaine de $(\mathbb{Z}_i(t_i))_k$. Nous rappelons que $\mathbb{Z}_i(t_i)$ est un nuage de points, et que son domaine est un ensemble de boîtes polaires.

$$\mathcal{C}_4\left([\mathbb{Z}_i(t_i)]_k\right) = [\mathbb{Z}_i(t_i)]_k \cap ([\rho_k(t_i)] \times [\phi_k(t_i)]). \tag{6.2}$$

6.2 Contracteurs de la contrainte de cartographie globale

Construisons maintenant un contracteur pour la contrainte 5 du système (4.13), que nous rappelons dans sa forme décomposée ici :

$$\begin{cases} \textbf{Variables :} \ \mathbb{M}_0,\, \mathbb{W}_0(t_i),\, \mathbb{W}_i(t_i),\, \mathbf{x}(t_i) \\ \textbf{Domaines :} \ [\mathbb{M}_0],\, [\mathbb{W}_0(t_i)],\, [\mathbb{W}_i(t_i)],\, [\mathbf{x}(t_i)] \\ \textbf{Contraintes :} \\ \quad 5.1. \quad \bigcup_{0\leq i\leq m-1} \mathbb{W}_0(t_i) \subset \overline{\mathbb{M}_0} \\ \quad 5.2. \quad \mathbb{W}_0(t_i) = \mathbf{h}(\mathbf{x}(t_i), \mathbb{W}_i(t_i)) \end{cases} \tag{6.3}$$

Dans la section 5.4 (page 80), nous avons mis en œuvre les outils développés au chapitre 5 afin de proposer un contracteur $\mathcal{C}_{5.1}$ associé à la contrainte 5.1.

Ensuite, dans la sous-section 4.2.2 (page 61) du chapitre précédent, nous avons montré comment décomposer la contrainte 5.2 en un ensemble de contraintes de deux types : des contraintes $\mathcal{L}_+$ de somme entre un ensemble et un vecteur de réels, ainsi que des contraintes $\mathcal{L}_{\text{cart/pol}}$ de changement de coordonnées cartésiennes/polaires.

6.2.1 Contracteur de somme entre un sous-ensemble réel et un vecteur

Rappelons la contrainte de somme $\mathcal{L}_+$:

$$\begin{cases} \textbf{Variables :} \ \mathbb{X},\, \mathbb{Y},\, \mathbf{z} \\ \textbf{Domaines :} \ [\mathbb{X}],\, [\mathbb{Y}],\, [\mathbf{z}] \\ \textbf{Contrainte :} \ \mathbb{X} = \mathbb{Y} + \mathbf{z} \end{cases}$$

Pour ce contracteur, nous aurons besoin de nouvelles opérations ensemblistes : la somme de Minkowski $\oplus$, et la différence de Minkowski $\ominus$ [59, 60, 61] définies comme suit :

$$\mathbb{X} \oplus \mathbb{Y} = \{\mathbf{x} + \mathbf{y} \mid \mathbf{x} \in \mathbb{X},\, \mathbf{y} \in \mathbb{Y}\}, \tag{6.4}$$

$$\mathbb{X} \ominus \mathbb{Y} = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbb{Y} + \mathbf{z} \subset \mathbb{X}\}. \tag{6.5}$$

La différence de Minkowski est la pseudo-inverse de la somme de Minkowski, en particulier nous avons :

$$\mathbb{X} \subset (\mathbb{Z} \ominus \mathbb{Y}) \iff (\mathbb{X} \oplus \mathbb{Y}) \subset \mathbb{Z}. \tag{6.6}$$

La somme et la différence de Minkowski sont utilisées notamment dans le domaine d'image, dans lequel elles sont appelées respectivement **dilatation** et **érosion**.

Ce sont de plus des notions duales :

$$\mathbb{X} \ominus \mathbb{Y} = \overline{(\overline{\mathbb{X}} \oplus (-\overline{\mathbb{Y}}))}. \tag{6.7}$$

Concernant le membre de droite de la contrainte $\mathcal{L}_+$, si $\mathbb{Y} \in [\mathbb{Y}]$ et $\mathbf{z} \in [\mathbf{z}]$, alors :

$$\mathbb{Y} + \mathbf{z} \in [\mathbb{Y}] + [\mathbf{z}] = [\mathbb{Y}^- \ominus (-[\mathbf{z}]), \mathbb{Y}^+ \oplus [\mathbf{z}]]. \tag{6.8}$$

Nous cherchons une fonction d'étiquetage $\mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}$ pour cet ensemble épais. Celui-ci correspond à une érosion de l'intérieur et de l'extérieur de $[\mathbb{Y}]$, ainsi qu'une dilatation de la pénombre, par l'ensemble $[\mathbf{z}]$.

Besoin d'un contracteur dédié L'algèbre des étiquettes ne traite pas de ce genre d'opération. Nous rappelons que pour un vecteur $\mathbf{z}$, son étiquette pour un ensemble épais $[\mathbb{Z}]$ correspond à son appartenance à l'intérieur, extérieur ou pénombre de $[\mathbb{Z}]$. Pour les opérations binaires $\diamond$ que nous avons étudiées jusque là, l'étiquette d'un point $\mathbf{z}$ dans $[\mathbb{X}] \diamond [\mathbb{Y}]$ peut être déduite de son étiquette dans $[\mathbb{X}]$ et dans $[\mathbb{Y}]$, ce qui donne une manière efficace de calculer ces opérations. Cette propriété n'existe pas avec l'opération $\oplus$: il est nécessaire de considérer toutes les paires $(\mathbf{x}, \mathbf{y})$ telles que $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Nous avons alors besoin de concevoir un contracteur dédié.

Nous procédons par étapes pour classer au mieux chaque boîte en l'une des 6 catégories des pavages épais réguliers. D'après l'équation (6.6), nous savons que la somme et la différence de Minkowski interagissent bien avec l'inclusion.

Dans un premier temps, nous cherchons donc à classer une boîte $[\mathbf{x}]$ dans l'une des 3 catégories non-frontières, à savoir intérieure (étiquette $(\top, \bot, \bot)$), extérieure (étiquette $(\bot, \top, \bot)$) ou pénombre (étiquette $(\bot, \bot, \top)$).

Ces étiquettes s'obtiennent avec des tests d'inclusions, à savoir respectivement les trois tests suivants :

$$[\mathbf{x}] \subset \mathbb{Y}^- \ominus (-[\mathbf{z}]) \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\top, \bot, \bot), \tag{6.9}$$

$$[\mathbf{x}] \subset \overline{\mathbb{Y}^+} \ominus (-[\mathbf{z}]) \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\bot, \top, \bot), \tag{6.10}$$

$$[\mathbf{x}] \subset (\mathbb{Y}^+ \setminus \mathbb{Y}^-) \oplus [\mathbf{z}] \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\bot, \bot, \top). \tag{6.11}$$

En utilisant la relation (6.6), nous avons les équivalences suivantes :

$$[\mathbf{x}] \subset \mathbb{Y}^- \ominus (-[\mathbf{z}]) \iff [\mathbf{x}] \oplus (-[\mathbf{z}]) \subset \mathbb{Y}^-, \tag{6.12}$$

$$[\mathbf{x}] \subset \overline{\mathbb{Y}^+} \ominus (-[\mathbf{z}]) \iff [\mathbf{x}] \oplus (-[\mathbf{z}]) \subset \overline{\mathbb{Y}^+}. \tag{6.13}$$

D'après les équations (6.12) et (6.13), pour toute boîte $[\mathbf{x}]$ nous avons :

$$\mathbf{label}_{[\mathbb{Y}]}([\mathbf{x}] - [\mathbf{z}]) = (\top, \bot, \bot) \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\top, \bot, \bot), \tag{6.14}$$

$$\mathbf{label}_{[\mathbb{Y}]}([\mathbf{x}] - [\mathbf{z}]) = (\bot, \top, \bot) \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\bot, \top, \bot). \tag{6.15}$$

Remarque 27. *La somme entre deux boîtes, selon l'arithmétique intervalle, correspond à la somme de Minkowski entre ces boîtes, considérées comme des sous-ensembles de* $\mathbb{R}^n$.

Pour calculer si une boîte se situe dans la pénombre, le calcul basé sur l'équation (6.11) est plus coûteux à effectuer. Ce dernier test consiste à dilater chaque boîte de la pénombre de $[\mathbb{Y}]$ par $[\mathbf{z}]$, et à comparer $[\mathbf{x}]$ à cet ensemble :

$$\left([\mathbf{x}] \subset \bigcup_{[\mathbf{y}] \subset \mathbb{Y}^+ \setminus \mathbb{Y}^-} [\mathbf{y}] + [\mathbf{z}]\right) \implies \left([\mathbf{x}] \subset \left((\mathbb{Y}^+ \setminus \mathbb{Y}^-) \oplus [\mathbf{z}]\right)\right) \implies \mathbf{label}_{[\mathbb{Y}]+[\mathbf{z}]}([\mathbf{x}]) = (\bot, \bot, \top). \tag{6.16}$$

Concernant les boîtes restantes, nous appliquons des tests supplémentaires pour prouver qu'au moins un point est respectivement intérieur, extérieur ou dans la pénombre. Dans notre implémentation, nous utilisons deux heuristiques différentes dans ce but.

- Nous effectuons les tests précédents sur certains points $\mathbf{x}$ de la boîte $[\mathbf{x}]$. Si l'un des tests est validé, alors il existe au moins un point respectivement intérieur, extérieur ou dans la pénombre. Nous pouvons donc mettre à $\top$ le test d'intersection non vide correspondant. Dans notre implémentation, nous testons chaque coin de la boîte.
- S'il existe une boîte $[\mathbf{z}'] \subset ([\mathbf{x}] + [\mathbf{z}])$ de la même taille que $[\mathbf{z}]$, telle que $\mathbf{label}_{[\mathbb{Y}]}([\mathbf{z}']) = (\top, \bot, \bot)$, alors il existe un point $\mathbf{x}$ dans $[\mathbf{x}]$ tel que :

$$\left([\mathbf{z}'] = \mathbf{x} + [\mathbf{z}]\right) \wedge \left([\mathbf{z}'] \subset \mathbb{Y}^-\right) \implies \mathbf{x} \in \mathbb{Y}^- \ominus (-[\mathbf{z}]). \tag{6.17}$$

  Il est donc possible de conclure que $[\mathbf{x}] \cap \mathbb{Y}^- \ominus (-[\mathbf{z}]) \neq \emptyset$. Il est possible de tenir le même raisonnement pour prouver que la boîte $[\mathbf{x}]$ est en partie extérieure.

Les autres intervalles de booléens prennent alors leur valeur par défaut, à savoir $[\bot, \top]$.

Pour la contraction de $[\mathbb{Y}]$, nous pouvons écrire de manière équivalente :

$$\mathbb{Y} = \mathbb{X} - \mathbf{z}, \tag{6.18}$$

et donc utiliser le procédé de classification précédent, en inversant le signe de $\mathbf{z}$.

Remarque 28. *La contraction de* $[\mathbf{z}]$ *n'est pas possible dans notre modélisation particulière du problème de SLAM du fait de la nature des ensembles. En effet, la technologie des capteurs utilisés ne permet pas d'obtenir de parties extérieures aux ensembles, là où il faut nécessairement à la fois des parties intérieures et extérieures pour espérer une contraction. Celle-ci ne sera donc pas étudiée ici.*

6.2.2 Contracteur du changement de coordonnées cartésiennes/polaires d'un sous-ensemble réel

Montrons comment définir une fonction d'étiquetage cohérente avec un changement de coordonnées cartésiennes/polaires, pour une contrainte $\mathcal{L}_{\text{cart/pol}}$:

$$\begin{cases} \textbf{Variables :} \ \mathbb{X}, \mathbb{X}^P \\ \textbf{Domaines :} \ [\mathbb{X}], [\mathbb{X}^P] \\ \textbf{Contrainte :} \ \mathbb{X} \overset{cart/pol}{\longleftrightarrow} \mathbb{X}^P \end{cases} \tag{6.19}$$

Le principe d'étiquetage ici est assez simple, comme pour l'égalité entre deux ensembles. Pour une boîte cartésienne $[\mathbf{x}]$, nous vérifions quelle partie de $[\mathbb{X}^P]$ elle intersecte après changement de coordonnées, et rajoutons sa valeur à l'étiquette. Cela revient à calculer une nouvelle étiquette $(\mathsf{x_i}, \mathsf{x_o}, \mathsf{x_u})$ où :

— $\mathsf{x_i} = \big([\mathbf{x}] \cap \mathbb{X}^{P^-} \neq \emptyset\big)$,
— $\mathsf{x_o} = \big([\mathbf{x}] \cap \overline{\mathbb{X}^{P^+}} \neq \emptyset\big)$,
— $\mathsf{x_u} = \big([\mathbf{x}] \cap \mathbb{X}^{P^+} \setminus \mathbb{X}^{P^-} \neq \emptyset\big)$.

Pour pouvoir mener à bien ces étiquetages, nous avons donc besoin de tests d'intersection entre une boîte cartésienne et un sous-pavage polaire ; ainsi qu'entre une boîte polaire et un sous-pavage cartésien pour contracter $[\mathbb{X}^P]$, où un sous-pavage désigne un sous ensemble de boîtes du pavages étant toutes intérieures, toutes extérieures ou toutes dans la pénombre.

Ces tests ne sont pas explicités dans ce manuscrit, mais sont triviaux avec le contracteur minimal $\mathcal{C}_{\text{Polar}}$ [62] qui permet de tester une intersection entre une boîte polaire et une boîte cartésienne.

6.3 Contracteur de la contrainte d'estimation d'état par les mesures

La contrainte 6 du système (4.13), rappelée ici, est la contrainte qui nous permettra de contracter la trajectoire :

$$6. \quad \bigcup_{0 \leq i \leq m-1} \mathbf{h}(\mathbf{x}(t_i), \mathbb{Z}_i(t_i)) \subset \partial\mathbb{M}_0. \tag{6.20}$$

Dans la section 4.2.3 du chapitre précédent, nous avons relevé la condition suivante :

$$\mathbf{h}([\mathbf{x}], [\mathbf{z}]) \subset \overline{\mathbb{M}_0} \implies \mathbf{x}(t_i) \notin [\mathbf{x}]. \tag{6.21}$$

Il devient alors possible de contracter la trajectoire à l'instant t_i avec l'algorithme suivant.

Algorithme 2 : Contraction de la pose en fonction des mesures

Données : $[\mathbf{h}], [\mathbf{z}], [\mathbb{M}_0], \varepsilon$
Entrée/Sortie : $[\mathbf{x}](t_i)$

$L \leftarrow \{[\mathbf{x}](t_i)\}$
$[\mathbf{r}] \leftarrow \emptyset$
tant que $L \neq \emptyset$ **faire**
  $[\mathbf{x}] \leftarrow L.\text{pop}()$
  si $[\mathbf{h}]([\mathbf{x}], [\mathbf{z}]) \cap \mathbb{M}_0^+ \neq \emptyset$ **alors**
    si $[\mathbf{x}].\mathit{width}() < \varepsilon$ **alors**
      $[\mathbf{r}] \leftarrow [\mathbf{r}] \sqcup [\mathbf{x}]$
    sinon
      $[\mathbf{x_l}], [\mathbf{x_r}] \leftarrow [\mathbf{x}].\text{bisect}()$
      $L.\text{push}([\mathbf{x_l}], [\mathbf{x_r}])$
    fin
  fin
fin
$[\mathbf{x}](t_i) \leftarrow [\mathbf{x}](t_i) \cap [\mathbf{r}]$

L'algorithme 2 fait des subdivisions de la boîte initiale $[\mathbf{x}](t_i)$ jusqu'à une précision ε, et retourne l'enveloppe intervalle du résultat. Il s'apparente à une forme de contraction par rognage (*shaving*) comme le font la consistance $3-B$ [63] ou encore la consistance **CID** [64].

Utilisation d'un pavage épais. Une autre façon d'envisager cette contraction est d'utiliser un pavage épais (se référer au chapitre 5). Nous définissons un pavage épais $[\mathbb{X}(t_i)]$, obtenu à partir de la fonction d'étiquetage qui à une boîte $[\mathbf{x}]$ en déduit :

— si $\mathbf{h}([\mathbf{x}],[\mathbf{z}]) \subset \overline{\mathbb{M}_0}$, alors $[\mathbf{x}]$ est étiquetée extérieure,

— sinon $[\mathbf{x}]$ est étiquetée frontière extérieure.

Nous obtenons ensuite la contraction de $[\mathbf{x}](t_i)$ en faisant l'union intervalle des boîtes non-extérieures de $[\mathbb{X}(t_i)]$. C'est cette seconde option qui a été retenue pour les expérimentations.

6.4 Expérimentations

Cette section illustre le comportement et les performances de notre approche ensembliste appliquée au problème de SLAM, à travers plusieurs scénarios simulés. Nous évaluons à la fois la capacité de la méthode à réduire les incertitudes sur la trajectoire et la carte, ainsi que l'influence sur les résultats des paramètres de résolution.

6.4.1 Protocole expérimental

Pour nos expérimentations, nous avons développé un générateur qui nous permet de simuler une trajectoire et un environnement réels, afin d'en déduire des données capteurs cohérentes.

Nous faisons alors varier différents paramètres du problème pour en déterminer les effets à l'aide d'indicateurs de performances que nous explicitons.

Données simulées

Les données simulées sont produites à partir d'une trajectoire analytique connue ainsi que ses dérivées, et d'un environnement défini par une image binaire où les pixels noirs correspondent à des obstacles. À partir de ces deux informations, nous pouvons alors simuler le comportement des différents capteurs : mesures d'accélération (IMU), mesure de cap et mesures LiDAR. Chaque mesure est bruitée puis bornée par un intervalle.

La figure 6.1 montre la trajectoire et l'environnement réels utilisés dans la simulation, ainsi que le tube associé à l'estimation odométrique seule, c'est-à-dire les contraintes 1 et 2 du système 4.13, page 60.

À chaque instant d'observation, nous simulons 1000 mesures LiDAR angulairement équiréparties. Les cônes ainsi formés (voir figure 4.1, page 58) se chevauchent, permettant alors de calculer une surface continue dépourvue d'obstacle.

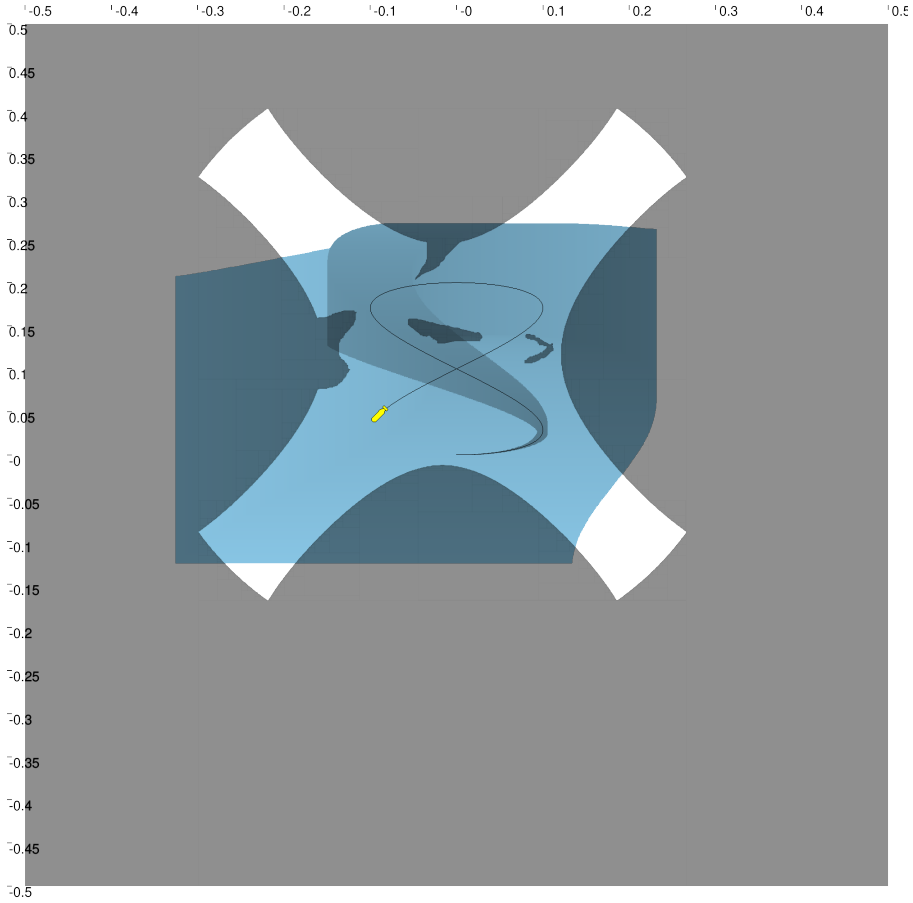


FIGURE 6.1 – Environnement et trajectoire utilisés pour la simulation. Le robot se déplace selon une courbe de Lissajous (trait noir), et est représenté à sa position finale. Le tube bleu représente l'estimation de la trajectoire réelle avec uniquement les contraintes 1 et 2 du problème, produisant sa pose parfaitement connue au départ mais très incertaine à la fin.

Paramètres expérimentaux

Nous faisons varier deux paramètres de résolution afin d'étudier leur impact :

- la précision des pavages, c'est-à-dire la plus petite taille possible de boîte du pavage, variant d'une taille de 16×16 pixels (1/256) à 1×1 pixel (1/4096) de l'image qui décrit l'environnement de base,
- le nombre d'observations que nous faisons varier de 6 à 21, faites à intervalles réguliers.

Indicateurs de performance

Nous évaluons le résultat de la résolution en utilisant plusieurs indicateurs :

- la **zone visible** : nous mesurons le pourcentage de surface cartographiée par rapport à la zone visible, c'est-à-dire calculée sans aucune incertitude sur les mesures,
- la **cartographie** : nous mesurons le pourcentage de surface cartographiée par rapport à la zone libre totale,

— l'**estimation du tube** : nous mesurons le rapport de réduction du volume du tube par rapport au volume de base illustré dans la figure 6.1, avec d'une part la position (x, y) (surface) et d'autre part l'orientation (θ),
— l'**estimation des poses** : toutes les missions simulées ont 6 instants d'observation en commun. Nous sommons les volumes des 6 poses obtenus à ces instants, avec d'une part la position (x, y) et d'autre part l'orientation (θ), valeurs que nous normalisons par le cas le plus défavorable.

6.4.2 Résultats et analyse

Résultats

La combinaison des deux paramètres conduit à neuf configurations expérimentales, dont les résultats sont synthétisés dans la table 6.1.

Préc. pavage	Nb obs.	Zone visible (%)	Carto. (%)	Estimation tube (x, y) (%)	Estimation tube θ (%)	Estimation poses (x, y) (%)	Estimation poses θ (%)	Temps (s)
1/256	6	77.78	76.42	22.27	85.26	100	100	167
1/256	6	77.78	76.42	21.97	84.41	92.18	97.32	502
1/256	11	79.09	77.87	15.62	71.71	96.08	90.65	582
1/256	21	79.94	78.80	12.01	64.30	89.22	85.48	1450
1/1024	6	91.30	89.71	17.59	63.85	25.69	52.19	724
1/1024	11	91.73	90.31	11.52	51.02	25.53	49.00	1130
1/1024	11	91.73	90.31	11.53	51.02	25.65	49.02	1990
1/1024	21	92.38	91.05	8.04	43.74	24.63	45.88	2440
1/4096	6	95.03	93.38	16.16	56.41	13.01	39.00	1830
1/4096	11	95.42	93.95	10.31	43.63	14.48	35.59	3940
1/4096	21	95.72	94.34	6.89	37.04	12.54	32.80	8190

TABLE 6.1 – Résultats expérimentaux obtenus pour différentes configurations de SLAM ensembliste. Les lignes en vert utilisent une autre modélisation du problème, en faisant des appariements (t_{i-1}, t_i) d'instants consécutifs, en plus des appariements (t_0, t_i) pour i dans $\{0, \ldots, m-1\}$.

La figure 6.2 illustre visuellement ces neuf cas, organisés selon la précision du pavage (du plus grossier en haut au plus fin en bas) et le nombre d'observations (de 6 à 21, de gauche à droite).

Analyse

Dans nos expérimentations, nous pouvons remarquer qu'en fin de trajectoire, le robot se perd rapidement : c'est parce que dans ce problème de SLAM, nous n'estimons pas la vitesse, donc celle-ci devient de plus en plus incertaine.

Une autre remarque concerne l'estimation de la pose du robot lorsque celui-ci fait de nouvelles observations. En effet, lorsque le robot passe en haut de la carte, il ne voit quasiment que des nouveaux obstacles. Sa pose doit alors être recalée sur des zones cartographiées a posteriori, donc plus incertaines. En revanche, en position finale, le robot perçoit à nouveau les mêmes obstacles qu'au début, d'où une

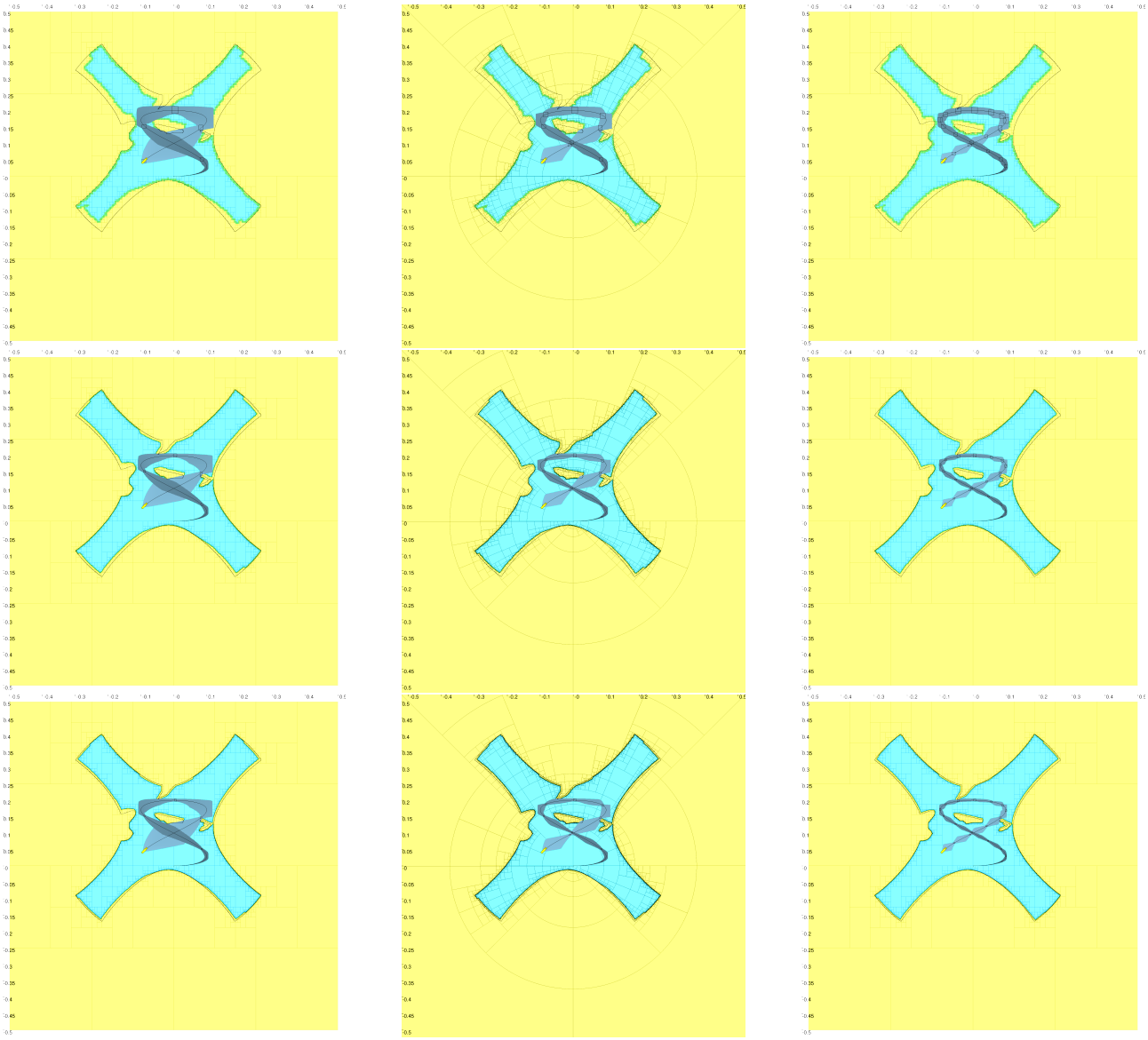


FIGURE 6.2 – Illustration graphique des neuf expériences en faisant varier la précision du pavage du moins précis (en haut) au plus précis (en bas), et le nombre d'observations de 6 (à gauche) à 21 (à droite). La trajectoire réelle et la frontière réelle de l'environnement sont représentées en trait noir. Le tube bleu est celui de la trajectoire après contraction. Il est notamment contracté à chaque instant d'observation (boîtes noires). Le pavage épais de chaque figure représente notre connaissance de la carte en trois couleurs (bleu = libre, jaune = incertain, vert = frontière). Afin de montrer les différentes représentations possibles, les cartes de la colonne du milieu (correspondant aux cas comprenant 11 instants d'observations) sont représentées par des pavages épais de boîtes *polaires*, contre des pavages de boîtes *cartésiennes* pour les autres colonnes.

bonne estimation de la pose. Ce phénomène illustre bien le rôle de la fermeture de boucle en SLAM : le fait de revisiter une zone connue permet de réduire drastiquement l'incertitude sur la position. Ici, notre approche ensembliste reproduit naturellement ce comportement.

Discussion

Les résultats mettent en évidence un impact important de la précision des pavages sur la zone visible, la cartographie et les poses, et du nombre d'instants d'observation sur la trajectoire et la cartographie. Le nombre d'observations a aussi un effet modéré sur la zone visible. En effet, en multipliant les points de vue, les zones visibles sont maintenant vues sous plusieurs angles différents, et ont donc plus de chances d'être cartographiées par nos méthodes.

L'ajout du nombre de points d'observation améliore aussi la précision des poses, en raison de cette meilleure cartographie. Nous pouvons cependant noter une exception à cette règle : l'avant dernière ligne du tableau (meilleure précision, 11 instants d'observation) donne un moins bon résultat que la ligne précédente (même précision, 6 instants d'observation). Cet effet est dû à notre contraction de la pose qui utilise des bissections. La grille induite par ces points de bissection entraîne un caractère aléatoire à la contraction qui en résulte, et un mauvais alignement entre la grille et l'ensemble associé peut donner un moins bon résultat. Ce contracteur n'est donc pas minimal. Une méthode basée sur du *shaving* (voir section 6.3 page 87) pourrait remédier à ce problème, mais n'a pas été testée.

Lorsque la précision du pavage devient très élevée, nous observons une forme de saturation, avec des contractions marginales de la carte, et donc un intérêt limité compte tenu du temps de calcul supplémentaire.

Un point important à noter est qu'une meilleure précision, avec moins d'instants d'observation, peut aboutir à des meilleurs résultats tout en ayant un meilleur temps d'exécution. Comparons par exemple dans la table 6.1 le cas avec la précision 1/256 et 21 observations et celui avec la précision 1/1024 et 11 observations. Ce constat est intéressant en pratique, puisque la précision des pavages est un paramètre interne au processus de résolution et peut être ajustée a posteriori au contraire du nombre d'observations, qui nécessite de nouvelles acquisitions.

Dans la modélisation choisie du problème, détaillée au chapitre 4, nous nous sommes restreints aux paires d'instants (t_0, t_i) pour tout i dans $\{1, \ldots, m-1\}$. Nous avons aussi testé d'autres appariements, à savoir les paires (t_{i-1}, t_i), pour tout i dans $\{1, \ldots, m-1\}$. Pour bien fonctionner, ces nouveaux appariements nécessitent $m-1$ nouvelles variables, qui sont les cartes complètes dans chaque repère $\mathcal{R}_i$. Ces ajouts entraîneraient des calculs supplémentaires et rendent le point fixe plus long à atteindre. C'est pourquoi nous avons testé une implémentation de « compromis », qui définit les $\mathbb{M}_i$ comme les ensembles complémentaires des $\mathbb{W}_i(t_i)$, pour tout i dans $\{1, \ldots, m-1\}$. Les résultats correspondant à cette modélisation apparaissent en vert dans la table 6.1. Ils ne montrent pratiquement aucune amélioration par rapport à la version restreinte aux paires d'instants (t_0, t_i), sauf dans le cas avec 6 poses seulement, et concernant le volume des poses aux instants d'observation, et pour un coût additionnel conséquent (un facteur 2 à 3). Le compromis choisi n'est donc pas prometteur.

Les expérimentations soulignent l'aspect garanti et exhaustif de notre approche, qui contraste avec les méthodes probabilistes. De plus, ces résultats confirment la pertinence du paradigme déclaratif des CSP pour la résolution du problème de SLAM. Enfin, le pouvoir expressif de notre approche permet de traduire de manière relativement directe les équations du problème vers des contracteurs.

Troisième partie

Contribution 2 : Procuste ensembliste et application au SLAM

Chapitre 7

Recalage ensembliste sous transformation de similarité

Plan du chapitre

Dans cette seconde contribution, nous allons nous intéresser au cas du recalage d'ensembles 2D sous transformation de similarité. Il s'agit alors de déterminer l'ensemble des transformations consistant en une série de translations, rotations et mises à l'échelle qui met en correspondance deux sous-ensembles de $\mathbb{R}^n$. Nous allons traiter ce problème de manière ensembliste, c'est-à-dire en éliminant des non-solutions au problème. À la différence de la contribution précédente, les ensembles sont entièrement connus, avec leur intérieur et extérieur, à une précision donnée, et ils seront manipulés au moyen d'une représentation différente des pavages épais réguliers.

Formulation mathématique classique du problème de recalage de formes

La transformation qui nous intéresse est la **transformation de similarité**.

Définition 21. *Dans le cadre de cette thèse, une* ***transformation de similarité*** *désigne une composition de rotation, translation et changement d'échelle uniforme, c'est-à-dire identique pour toutes les directions. Nous désignons par* $\mathbb{S} = \big(\mathbb{R}/(2\pi\mathbb{Z})\big) \times \mathbb{R}^2 \times \mathbb{R}^+$ *l'espace des paramètres de toutes les transformations de similarité dans un espace 2D.*

Le problème est alors formalisé de la manière suivante. Soient $\mathbb{A}$ et $\mathbb{B}$ deux ensembles de $\mathbb{R}^n$. Nous cherchons à trouver une fonction de similarité $\mathbf{h_p}(\cdot)$ telle que :

$$\mathbf{h_p}(\mathbb{A}) = \mathbb{B}, \tag{7.1}$$

où $\mathbf{h_p}(\cdot)$ est sous la forme suivante :

$$\mathbf{h_p}(\mathbf{a}) = k\mathbf{R_r}\mathbf{a} + \mathbf{t}, \tag{7.2}$$

avec $\mathbf{p} = (\mathbf{r}, \mathbf{t}, k)^\intercal \in \mathbb{S}$ étant le vecteur des paramètres de la fonction. La composante de rotation est ici effectuée grâce à la matrice de rotation $\mathbf{R_r}$.

Ce problème peut aussi être écrit dans version linéaire, sous forme matricielle, à l'aide de coordonnées homogènes :

$$\begin{pmatrix} \mathbb{B} \\ 1 \end{pmatrix} = \left(\begin{array}{c|c} k\mathbf{R_r} & \mathbf{t} \\ \hline 0 \quad \cdot\cdot \quad 0 & 1 \end{array}\right) \begin{pmatrix} \mathbb{A} \\ 1 \end{pmatrix}, \tag{7.3}$$

Dans la suite, la matrice de transformation sera notée $\mathbf{M}$.

L'objectif est alors de caractériser le vecteur $\mathbf{p}$, ou la matrice $\mathbf{M}$.

Applications du problème de recalage

Comme vu dans les différents exemples en introduction, le problème de recalage d'ensemble apparaît dans divers contextes. Ainsi, que ce soit en géométrie, vision par ordinateur, image - notamment imagerie médicale, robotique, ou bien en analyse factorielle qui réutilise plutôt le nom de Procuste, ce problème et ses variantes sont particulièrement étudiés, avec de nombreuses applications variées.

Fusion de données Plusieurs images du même sujet peuvent être combinées afin d'aligner au mieux ces mesures. Ces images peuvent être issues de différents points de vue, de différents instants, d'une même vidéo, ou encore être des données multi-modales. Ainsi, l'alignement de ces données peut améliorer le rapport signal-bruit (SNR), améliorer la résolution d'une image [65], construire incrémentalement une carte dans le cas du SLAM, faire de la surveillance (*monitoring*) comme celle d'une tumeur, pouvoir être combinées avec d'autres modes d'acquisition – une image issue de Tomographie par Émission de Positron (PET) peut être alignée avec une Imagerie par Résonance Magnétique (MRI) en une image PET-MRI [66]. Elle permet aussi la reconstruction 3D comme l'alignement de tranches adjacentes d'une MRI, permettre la stéréovision (ou mesure stéréoscopique) [67] et mettre en oeuvre de la *Structure from Motion* (SfM) [68].

Estimation du mouvement Outre les images alignées, le recalage d'image nous donne aussi les paramètres de la fonction qui aligne au mieux ces images. Ainsi pour un sujet en mouvement, le recalage d'image peut être utilisé à partir de vidéos [69] et nous permet alors d'estimer ce mouvement et ses caractéristiques, à des fins d'étude du mouvement ou encore de compression vidéo par *compensation de mouvement* [70] comme c'est le cas dans plusieurs standards de vidéo-compression tels H.264 ou MPEG-2. De manière duale, ce déplacement peut être interprété comme un changement de point de vue, et donc celui de la caméra plutôt que du sujet, comme c'est le cas pour la partie localisation du SLAM [71].

7.1 Présentation de l'approche par cercles minimaux

Le recalage d'ensembles 2D sous transformation de similarité est un problème à cinq dimensions. Ce problème revient à réaliser une inversion de fonction ensembliste afin de trouver l'ensemble des transformations $\mathbb{P} \in \mathcal{P}(\mathbb{S})$ étant donné deux ensembles bornés $\mathbb{A}$ et $\mathbb{B}$. Cependant, une inversion ensembliste est une tâche difficile pour des problèmes à grande dimension, du fait de l'utilisation de méthodes de séparation et évaluation, ou Branch and Bound (B&B), qui effectuent des bissections dans chaque dimension de l'espace [57].

7.1.1 Étapes de l'approche

L'approche par cercles minimaux envisagée ici, inspirée des méthodes Procuste, consiste à transformer ce problème en un problème de recalage par rotation, et donc d'effectuer des bissections sur une seule dimension.

La figure 7.1 représente le schéma général de cette approche, et les étapes principales sont détaillées ci-après.

Étape 1 : cercle englobant minimum

L'idée centrale est de s'appuyer sur les cercles englobants minimaux – et nécessairement uniques – pour $\mathbb{A}$ et $\mathbb{B}$, que l'on peut considérer comme l'enveloppe circulaire d'un ensemble. Ce problème équivaut au problème bien connu du calcul du « cercle minimum » [72], qui vise à trouver le plus petit cercle englobant un nombre donné de points. Pour d'autres formulations, il s'agit également du cas non pondéré d'un problème d'optimisation : le problème du 1-centre, qui consiste à trouver un point optimal de l'espace qui minimise la distance maximale à un ensemble de points donné. Celui-ci peut être trouvé en temps linéaire [73]. Cependant, les méthodes classiques de résolution de ce problème ne s'appliquent pas dans notre cas, en raison de la nature de nos ensembles en entrée, non finis, et de la nécessité d'obtenir des résultats fiables et exhaustifs.

Lorsqu'il est étendu aux ensembles, le problème consiste à identifier le cercle englobant un ensemble donné $\mathbb{X}$, avec le plus petit rayon. La première contribution de ce document est de donner une solution en théorie des ensembles pour calculer une approximation de ce cercle pour les ensembles bornés. Cette solution sera développée dans le chapitre 9 avec l'utilisation de séparateurs et de projections. Ainsi, étant donné deux ensembles bornés $\mathbb{A}$ et $\mathbb{B}$, nous pourrons identifier deux vecteurs uniques $\mathbf{c}^{\mathbb{A}}$ et $\mathbf{c}^{\mathbb{B}}$ dans $\mathbb{R}^2 \times \mathbb{R}^+$, en décrivant leur centre $\mathbf{c}_{1,2}$ et leur rayon c_3.

Le cercle englobant minimum possède des propriétés utiles, telles que l'existence et l'unicité lorsque les ensembles sont bornés. Les deux vecteurs identifiés $\mathbf{c}^{\mathbb{A}}$ et $\mathbf{c}^{\mathbb{B}}$ fourniront les paramètres de normalisation pour l'étape suivante.

Étape 2 : standardisation par recentrage et normalisation

L'étape de normalisation permettra l'alignement par rotation. Les ensembles $\mathbb{A}$ et $\mathbb{B}$ seront respectivement normalisés en $\mathbb{A}_N$ et $\mathbb{B}_N$ après translation et mise à l'échelle, comme suit :

FIGURE 7.1 – **Étapes de l'approche de type Procuste.** 1 – Étant donné deux ensembles $\mathbb{A}$ et $\mathbb{B}$, nous identifions les deux cercles minimaux uniques pour chaque ensemble. 2 – L'étape de normalisation est divisée en centrage et mise à l'échelle : 2.1 – Centrage : sur la base des centres des deux cercles, nous translatons $\mathbb{A}$ et $\mathbb{B}$ de manière à ce qu'ils partagent le même centre. 2.2 : Mise à l'échelle : sur la base des rayons des cercles, nous mettons à l'échelle les ensembles translatés en fonction du rapport des rayons. 3 – Correspondance par rotation : les paramètres de rotation possibles sont identifiés en faisant tourner les deux ensembles normalisés. Dans cet exemple, trois rotations sont possibles en raison des symétries des formes initiales.

$$\mathbb{A}_N := \left(\mathbb{A} - \mathbf{c}^{\mathbb{A}}_{1,2}\right)/c^{\mathbb{A}}_3 \quad \text{et} \quad \mathbb{B}_N := \left(\mathbb{B} - \mathbf{c}^{\mathbb{B}}_{1,2}\right)/c^{\mathbb{B}}_3. \tag{7.4}$$

Étape 3 : alignement par rotation

Cette étape vise à trouver toutes les rotations qui alignent les deux ensembles normalisés, décrivant ainsi l'ensemble Θ défini comme suit :

$$\Theta := \{\theta \mid \mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N\} \tag{7.5}$$

7.1.1.1 Lien avec la transformation de similarité

Nous pouvons exprimer Θ en utilisant les ensembles initiaux $\mathbb{A}$ et $\mathbb{B}$ et la transformation de similarité par substitution, en utilisant (7.4).

Cette substitution nous donne :

$$\Theta = \left\{ \theta \in \mathbb{R} \;\middle|\; \begin{pmatrix} \mathbb{B} \\ 1 \end{pmatrix} = \left(\begin{array}{cc|c} & & \mathbf{c}^{\mathbb{B}}_{\mathbf{12}} - \frac{c^{\mathbb{B}}_3}{c^{\mathbb{A}}_3}\mathbf{R}_\theta \cdot \mathbf{c}^{\mathbb{A}}_{\mathbf{12}} \\ \multicolumn{2}{c|}{\raisebox{1.5ex}[0pt]{$\frac{c^{\mathbb{B}}_3}{c^{\mathbb{A}}_3}\mathbf{R}_\theta$}} & \\ \hline 0 & 0 & 1 \end{array}\right) \begin{pmatrix} \mathbb{A} \\ 1 \end{pmatrix} \right\} \tag{7.6}$$

Nous pouvons déduire l'ensemble $\mathbb{P} \in \mathcal{P}(\mathbb{S})$ des transformations de similarité cohérentes avec (7.3) en identifiant les éléments de la matrice avec la matrice de transformation de similarité :

$$\mathbb{P} := \left\{ (\theta, \mathbf{t}, k)^\intercal \in \mathbb{S} \;\middle|\; (\theta \in \Theta) \wedge (k = \frac{c^{\mathbb{B}}_3}{c^{\mathbb{A}}_3}) \wedge (\mathbf{t} = \mathbf{c}^{\mathbb{B}}_{\mathbf{12}} - k\mathbf{R}_\theta \cdot \mathbf{c}^{\mathbb{A}}_{\mathbf{12}}) \right\} \tag{7.7}$$

7.1.2 Correction de l'approche

La procédure de normalisation présentée ci-dessus repose sur l'hypothèse que les paramètres du cercle minimal découplent efficacement les facteurs de translation et de mise à l'échelle de la transformation de similarité. Nous fournissons ci-après une justification formelle de cette approche.

Proposition 2. *Soit $\mathbf{f_p}(\cdot)$ une transformation de similarité caractérisée par un facteur d'échelle k, une matrice de rotation $\mathbf{R}_\theta$ et un vecteur de translation $\mathbf{t}$. S'il existe un vecteur de paramètres $\mathbf{p}$ tel que $\mathbb{B} = \mathbf{f_p}(\mathbb{A})$, alors les ensembles normalisés $\mathbb{A}_N$ et $\mathbb{B}_N$ définis en (7.4) ne diffèrent que par une rotation $\mathbf{R}_\theta$.*

Démonstration. Le cercle minimal de l'ensemble $\mathbb{A}$ est défini de manière unique par son centre $\mathbf{c}^{\mathbb{A}}_{\mathbf{1,2}}$ et son rayon $c^{\mathbb{A}}_3$ comme suit :

$$\mathbf{c}^{\mathbb{A}}_{\mathbf{1,2}} = \arg\min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} \|\mathbf{a} - \mathbf{x}\| \tag{7.8}$$

$$c^{\mathbb{A}}_3 = \min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} \|\mathbf{a} - \mathbf{x}\| \tag{7.9}$$

Considérant l'ensemble $\mathbb{B} = \mathbf{f_p}(\mathbb{A})$, le rayon $c_3^{\mathbb{B}}$ de son cercle minimum satisfait :

$$\begin{aligned} c_3^{\mathbb{B}} &= \min_{\mathbf{y}\in\mathbb{R}^2} \max_{\mathbf{b}\in\mathbb{B}} \|\mathbf{b}-\mathbf{y}\| \\ &= \min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} \|\mathbf{f_p}(\mathbf{a}) - \mathbf{f_p}(\mathbf{x})\| \quad \text{puisque } \mathbf{f_p} \text{ est une bijection} \\ &= \min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} \|(k\mathbf{R}_\theta\mathbf{a}+\mathbf{t}) - (k\mathbf{R}_\theta\mathbf{x}+\mathbf{t})\| \\ &= \min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} k\|\mathbf{R}_\theta(\mathbf{a}-\mathbf{x})\| \\ &= k \min_{\mathbf{x}\in\mathbb{R}^2} \max_{\mathbf{a}\in\mathbb{A}} \|\mathbf{a}-\mathbf{x}\| \quad \text{comme } \mathbf{R}_\theta \text{ est orthogonal} \\ &= k\, c_3^{\mathbb{A}} \end{aligned} \tag{7.10}$$

En utilisant le même raisonnement et en prêtant attention au changement de variables à la deuxième ligne, le centre du cercle minimum de $\mathbb{B}$ est donné par :

$$\mathbf{c}_{\mathbf{1,2}}^{\mathbb{B}} = \mathbf{f_p}(\mathbf{c}_{\mathbf{1,2}}^{\mathbb{A}}). \tag{7.11}$$

Enfin, en substituant $\mathbf{c}_{\mathbf{1,2}}^{\mathbb{B}}$ et $c_3^{\mathbb{B}}$ dans la définition de l'ensemble normalisé $\mathbb{B}_N$:

$$\begin{aligned} \mathbb{B}_N &= (\mathbb{B} - \mathbf{c}_{\mathbf{1,2}}^{\mathbb{B}})/c_3^{\mathbb{B}} \\ &= (\mathbf{f_p}(\mathbb{A}) - \mathbf{f_p}(\mathbf{c}_{\mathbf{1,2}}^{\mathbb{A}}))/(k\, c_3^{\mathbb{A}}) \\ &= (k\mathbf{R}_\theta\mathbb{A} + \mathbf{t} - (k\mathbf{R}_\theta\mathbf{c}_{\mathbf{1,2}}^{\mathbb{A}} + \mathbf{t}))/(k\, c_3^{\mathbb{A}}) \\ &= \mathbf{R}_\theta(\mathbb{A} - \mathbf{c}_{\mathbf{1,2}}^{\mathbb{A}})/c_3^{\mathbb{A}} \\ &= \mathbf{R}_\theta\mathbb{A}_N \end{aligned} \tag{7.12}$$

Les ensembles normalisés ne diffèrent donc que par rotation $\mathbf{R}_\theta$, ce qui conclut la preuve. □

7.1.3 Synthèse de l'approche : système de contraintes

Pour résumer l'approche, le système présenté en (7.13) englobe toutes les équations nécessaires afin de trouver l'ensemble $\mathbb{P} = \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) = \mathbb{B}\}$ de tous les paramètres des solutions. Le système implique également des variables intermédiaires qui seront estimées en même temps que les variables du problème. La fonction $\texttt{MinCircle} : \mathcal{P}(\mathbb{R}^2) \to \mathbb{R}^3$ renvoie le cercle minimum d'un ensemble dense et borné.

$$\begin{cases} 1.\ \mathbf{c}^{\mathbb{A}} = \texttt{MinCircle}(\mathbb{A}), \\ 2.\ \mathbf{c}^{\mathbb{B}} = \texttt{MinCircle}(\mathbb{B}), \\ 3.\ \mathbb{A}_N = \left(\mathbb{A} - \mathbf{c}_{\mathbf{12}}^{\mathbb{A}}\right)/c_3^{\mathbb{A}}, \\ 4.\ \mathbb{B}_N = \left(\mathbb{B} - \mathbf{c}_{\mathbf{12}}^{\mathbb{B}}\right)/c_3^{\mathbb{B}}, \\ 5.\ \mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N, \\ 6.\ k = c_3^{\mathbb{B}}/c_3^{\mathbb{A}}, \\ 7.\ \mathbf{t} = \mathbf{c}_{\mathbf{12}}^{\mathbb{B}} - k\mathbf{R}_\theta \cdot \mathbf{c}_{\mathbf{12}}^{\mathbb{A}}. \end{cases} \tag{7.13}$$

7.2 Méthodes de recalage de formes

Intéressons-nous dorénavant aux méthodes existantes pour résoudre ce problème. Deux types d'approche sont envisagés, répondant chacun à une problématique différente du même problème.

7.2.1 Méthodes statistiques : méthode de Procuste et variantes

En analyse factorielle, le problème de recalage apparaît sous le nom de superposition de Procuste, ou *Procrustes Superimposition* (**PS**) et est au cœur de ce que l'on appelle l'analyse procustéenne. Dans celle-ci, on se donne deux configurations possibles – deux nuages de points – $\mathbf{A}$ et $\mathbf{B}$ sous forme matricielle, et l'on cherche la transformation $\mathbf{M}$ qui les aligne au mieux selon une métrique choisie.

Remarque 29. *En procédant de cette manière, il est possible de trouver une solution quels que soient les ensembles de départ, même si ceux-ci n'ont pas la même forme. C'est ce qui a motivé au départ le nom de Procuste pour ces méthodes, empruntant le nom de ce personnage qui, dans la mythologie grecque, faisait rentrer de force ses victimes $\mathbb{A}$ sur son lit $\mathbb{B}$, même lorsque le lit n'avait pas la bonne taille.*

En général, la métrique choisie est la norme de Frobenius, notée $||\cdot||_F$, c'est-à-dire la racine de la somme des carrés de tous les éléments de la matrice :

$$||\mathbf{X}||_F = \sqrt{\sum_{1\leq i,j\leq n} x_{i,j}^2} \tag{7.14}$$

La méthode cherche donc une matrice de transformation $\mathbf{M}$ telle que $||\mathbf{B}-\mathbf{MA}||_F$ soit minimisée.

$$\mathbf{M} = \arg\min_{\mathbf{M}'} ||\mathbf{B}-\mathbf{M}'\mathbf{A}||_F. \tag{7.15}$$

Remarque 30. *Si la transformation fait intervenir des translations, et est donc non linéaire, cela revient à trouver le couple $(\mathbf{M},\mathbf{t})$ qui minimise $||\mathbf{B}-\mathbf{MA}-\mathbf{t}||_F$, ou encore à utiliser les coordonnées homogènes comme vu dans l'équation* (7.3).

Variante orthogonale

Les premières versions [74, 75] étaient dédiées au cas où $\mathbf{M}$ est orthogonale, c'est-à-dire :

$$\mathbf{M}\mathbf{M}^\intercal = \mathbf{M}^\intercal\mathbf{M} = 1. \tag{7.16}$$

On y retrouve les matrices de rotation, avec ou sans réflexion.

Il y a différentes techniques permettant d'estimer cette matrice orthogonale. Il existe deux types d'approches, des méthodes itératives ou des méthodes retournant une solution analytique. Cependant, comme les méthodes itératives ne garantissent pas la convergence vers le minimum global [76], nous étudions plutôt les solutions analytiques. Celles-ci peuvent s'obtenir par la décomposition en valeurs

1. centrage de $\mathbf{A}$ et $\mathbf{B}$ avec leur centre de masse,
2. normalisation : définition de $\mathbf{A}^\star$ et $\mathbf{B}^\star$ par mise à l'échelle telle que $||\mathbf{A}^\star||_F = ||\mathbf{B}^\star||_F = 1$,
3. application de `ProcusteOrthogonal` sur $\mathbf{A}^\star$ et $\mathbf{B}^\star$,
4. calcul du paramètre de mise à l'échelle selon [80],
5. calcul du paramètre de translation par correspondance des centres de masses.

FIGURE 7.2 – Description de la méthode `ProcustePaire`

singulières, ou *Singular Value Decomposition* (SVD) [77], par les matrices orthonormales [78], ou encore par les quaternions [79].

Dans la suite, `ProcusteOrthogonal` désignera une méthode qui prend en entrée deux matrices $\mathbf{A}$ et $\mathbf{B}$ et donne la matrice orthogonale $\mathbf{M}$ minimisant la quantité $||\mathbf{B} - \mathbf{MA}||_F$.

Variante avec transformation de similarité : `ProcustePaire`

Pour une transformation de similarité, [80] propose une solution. Schématiquement, elle s'appuie sur une étape préliminaire de normalisation avant d'effectuer la version orthogonale de Procuste. Cette méthode, nommée `ProcustePaire` dans la suite, est décrite dans la Figure 7.2.

Remarque 31. *Avec la prise en compte des translations et de la mise à l'échelle dans la transformation à identifier, le nom de Procuste a acquis une nouvelle signification a posteriori : puisque dans la légende, Procuste raccourcissait ou allongeait les membres de ses victimes pour les faire correspondre à son lit, la transformation nécessaire pour placer la victime* $\mathbf{A}$ *sur son lit* $\mathbf{B}$ *s'apparente à une transformation de similarité.*

Variante étendant la méthode à plus de deux configurations : GPA

L'analyse procustéenne généralisée, ou *Generalized Procrustes Analysis* (GPA) [81], est une variante du problème où le domaine d'application est étendu à un plus grand nombre de configurations.

Cette version du problème est formalisée ainsi : soient $\mathbf{A_1}, \ldots, \mathbf{A_m}$ m configurations différentes. GPA cherche à la fois une forme consensus $\mathbf{B}$, ou moyenne de toutes les configurations, ainsi que les m transformations de similarité $\mathbf{M_1}, \ldots, \mathbf{M_m}$ telles que :

$$\mathbf{B}, \mathbf{M_1}, \ldots, \mathbf{M_m} = \underset{\mathbf{B}, \mathbf{M_1}, \ldots, \mathbf{M_m}}{\arg\min} \sum_{i=1}^{m} ||\mathbf{B} - \mathbf{M_i A_i}||_\mathbf{F}. \tag{7.17}$$

Les méthodes classiques de résolution pour ce problème sont itératives [81, 82, 83], et sont donc sensibles à une bonne estimation initiale de la configuration consensus. Une bonne technique d'initialisation est décrite dans [84]. Les méthodes itératives alternent entre une estimation de la transformation et une estimation de la configuration consensus. Dans sa forme simplifiée, la méthode itérative donne :

1. initialisation : on choisit un ensemble de transformations initiales, par exemple $(\mathbf{M_1}, \mathbf{M_2}, \ldots, \mathbf{M_m}) = (\mathbf{I}, \mathbf{0}, \ldots, \mathbf{0})$,

2. la nouvelle forme consensus $\mathbf{B}$ est estimée comme moyenne des $\mathbf{M_i A_i}$,
3. application de `ProcustePaire` sur les m paires $(\mathbf{A_i}, \mathbf{B})$ pour actualiser les $\mathbf{M_i}$,
4. retour à l'étape 2 jusqu'à convergence.

Variante où $\mathbb{A}$ est inclus dans $\mathbb{B}$ après transformation : ICP

En robotique, le problème de SLAM présenté comme exemple introductif a comme sous-problème une variante du problème de Procuste. En effet, il s'agit de trouver une transformation $\mathbf{h_p}(\cdot)$ telle que la carte locale $\mathbb{A}$ perçue par le robot corresponde avec la carte globale $\mathbb{B}$. Ainsi, il s'agit de décrire $\mathbf{p}$ tel que :

$$\mathbf{h_p}(\mathbb{A}) \subset \mathbb{B}. \tag{7.18}$$

Dans sa version originale, les ensembles $\mathbb{A}$ et $\mathbb{B}$ sont décrits avec des nuages de points $\mathbf{A}$ et $\mathbf{B}$, et la transformation $\mathbf{h_p}(\cdot)$ est rigide. En effet, lorsque l'on ajoute le changement d'échelle, le cas $k = 0$ donne lieu a une multitude de solutions qui vérifient facilement la condition d'inclusion, mais sont peu intéressantes.

L'idée est de partir d'une transformation $\mathbf{M}$, puis d'itérer une sélection d'un sous ensemble $\mathbf{B'}$ de $\mathbf{B}$ et l'analyse procustéenne classique entre $\mathbf{A}$ et $\mathbf{B'}$, jusqu'à convergence.

La méthode *Iterative Closest Point* (ICP) [85, 86, 87] résout cette problématique en sélectionnant, dans $\mathbf{B}$, les points les plus proches de $\mathbf{MA}$ à chaque itération.

Cette version itérative n'est pas globale et risque de n'atteindre qu'un minimum local, dépendant du choix de la transformation initiale. Pour augmenter les chances d'obtenir le minimum global, la méthode est répétée à partir d'un ensemble de transformations initiales dispersées pour couvrir l'espace de recherche de la transformation optimale.

Le schéma général de cette méthode est alors le suivant :

1. initialisation : on se donne une transformation initiale $\mathbf{M}$,
2. on choisit $\mathbf{B'}$ comme sous ensemble de $\mathbf{B}$ des points les plus proches de $\mathbf{MA}$,
3. application de `ProcustePaire` sur $(\mathbf{A}, \mathbf{B'})$ pour estimer $\mathbf{M}$,
4. retour à l'étape 2. jusqu'à convergence.

7.2.2 Discussion sur les différentes variantes des méthodes statistiques et leurs limites

Dans les applications pratiques, les ensembles $\mathbb{A}$ et $\mathbb{B}$ sont rarement observés directement : ils sont plutôt représentés par des échantillons finis, souvent corrompus par du bruit. Par conséquent, il est rarement possible d'obtenir une correspondance parfaite entre deux ensembles de données. Dans ce contexte, le meilleur alignement obtenu à partir de (7.15) est généralement considéré comme une approximation pertinente du problème de recalage sous-jacent (7.1).

Cependant, les méthodes statistiques ou d'optimisation classiques présentent plusieurs inconvénients majeurs :

— **Fausse convergence :** elles produisent invariablement une seule « meilleure » solution, même lorsqu'il n'existe pas de correspondance exacte.
— **Ambiguïté et symétries :** elles convergent généralement vers une solution unique, sans tenir compte des cas où plusieurs minima globaux existent en raison de symétries (voir la figure 7.3).
— **Manque de fiabilité :** N'étant pas garanties, ces méthodes peuvent converger vers des minima locaux, conduisant à des correspondances incorrectes.

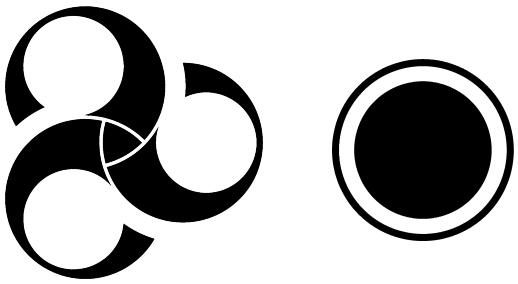


FIGURE 7.3 – Exemples de problèmes de recalage sans solution unique : un triskel (à gauche) présentant une triple symétrie de rotation, donnant lieu à trois solutions distinctes, et une configuration disque/anneau (à droite) où toute rotation constitue une solution valide.

Alors que les méthodes orthogonales et de similarité de Procruste peuvent souffrir de problèmes numériques locaux, la solution analytique basée sur les quaternions proposée par [79] permet de calculer un optimum global. Cependant, même un alignement globalement optimal peut ne pas refléter le véritable recalage lorsque les données fournissent des observations bruitées ou non exhaustives des ensembles sous-jacents.

Comme dans de nombreux cadres basés sur l'optimisation, permettre la mise à l'échelle dans ce contexte peut entraîner une divergence de la méthode vers des solutions dégénérées où $k = 0$. Cela empêche de trouver des solutions significatives (même pour les versions globales telles que Go-ICP [88]), un phénomène documenté dans [89].

En revanche, l'approche proposée dans ce manuscrit s'appuie sur des méthodes ensemblistes et intervalles. Ces paradigmes garantissent une recherche exhaustive et fiable, permettant une caractérisation rigoureuse de toutes les transformations possibles.

7.2.3 Méthodes ensemblistes existantes pour le recalage

Afin de développer un cadre exhaustif capable de saisir toute la multiplicité des solutions, la méthodologie que nous utilisons dans cette thèse s'appuie sur des outils ensemblistes. Ces techniques sont particulièrement efficaces pour caractériser les ensembles de solutions réalisables en éliminant systématiquement les régions de l'espace de recherche qui ne contiennent aucune solution valide. Dans le contexte du recalage, cela permet d'obtenir une approximation rigoureuse et fiable de l'ensemble $\mathbb{P} = \{\mathbf{p} \in \mathbb{S} \mid \mathbf{f_p}(\mathbb{A}) = \mathbb{B}\}$.

Une base formelle pour le recalage dans le paradigme ensembliste est fournie dans [90], qui présente plusieurs opérateurs adaptés à ce problème. Notre travail s'appuie sur deux composantes de ces travaux :

— le formalisme *séparateur* détaillé dans le chapitre 8, qui permet de représenter et manipuler des ensembles,

— un opérateur permettant de répondre au problème : Desrochers propose un séparateur exploitant une fonction paramétrée (ici une fonction de similarité) mettant en relation deux ensembles. Ce séparateur est décrit en section 8.4. Celui-ci pourrait en théorie résoudre le problème de recalage, mais il est inapplicable directement en raison de la grande dimensionnalité de l'espace des paramètres $\mathbb{S}$. Cet opérateur est décrit en section 8.4.

La principale contribution de cet article est l'introduction d'une décomposition de type Procuste : en décomposant le problème initial en une séquence de sous-problèmes de dimension inférieure, nous contournons ce problème de dimensionnalité et permettons l'utilisation efficace d'outils ensemblistes.

Les outils numériques et les algorithmes basés sur les intervalles nécessaires au calcul de l'approche Procuste sont détaillés dans les chapitres 8 et 9.

Chapitre 8

Les séparateurs comme représentation des sous-ensembles réels

Plan du chapitre

Dans cette contribution, les entrées du problème de recalage sont les ensembles bornés $\mathbb{A}$ et $\mathbb{B}$ de $\mathbb{R}^2$, pour lesquels nous cherchons à estimer l'ensemble des transformations de similarité introduit précédemment, $\mathbb{P} = \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h}_\mathbf{p}(\mathbb{A}) = \mathbb{B}\}$. Comme les méthodes ensemblistes éliminent des non-solutions, elles calculent plutôt un sur-ensemble $\mathbb{P}^+$ de $\mathbb{P}$. Cependant, le calcul de $\mathbb{P}^+ \in \mathcal{P}(\mathbb{S})$ avec un algorithme classique de paveur (voir section 2.2.4.2, page 22) est coûteux, en raison de la complexité exponentielle de ces méthodes de branchement et des 4 dimensions de $\mathbb{P}^+$. Dans le chapitre 7, nous avons présenté une séquence d'opérations sur les ensembles suivant une méthode de type Procuste qui limite la complexité de cette approximation à un problème de *branch-and-bound* unidimensionnel. Nous avons aussi vu, au chapitre 2, comment représenter et manipuler des ensembles en se basant sur l'analyse intervalle. Inspiré de la contribution précédente, consacrée au SLAM ensembliste, nous avons exploité les pavages épais réguliers afin de manipuler ces ensembles.

Pour la seconde contribution de ce manuscrit (cf. partie II), nous allons utiliser un autre outil : le *séparateur*. Ce chapitre est consacré à cette représentation, en fournissant aussi les outils algorithmiques utilisés pour sa manipulation.

8.1 Séparateurs

Nous avons déjà introduit les *contracteurs* à la section 2.2.4, page 19, qui permettent d'obtenir une surestimation d'un ensemble solution pour une contrainte donnée. Lorsque nous avons besoin de calculer

aussi une approximation intérieure de cet ensemble, nous avons vu qu'il était possible de déterminer un second contracteur, obtenu à partir de la négation de la contrainte. Cette paire d'opérateurs est à la base des pavages intervalles, définis en section 3.2.3, page 44.

Dans des contributions plus récentes, cette paire de contracteurs forme un *séparateur* [91], un opérateur de $\mathbb{IR}^n \to \mathbb{IR}^n \times \mathbb{IR}^n$ qui *sépare* une boîte en deux boîtes, représentant respectivement les approximations intérieures et extérieures d'un ensemble. La figure 8.1 illustre cet opérateur.

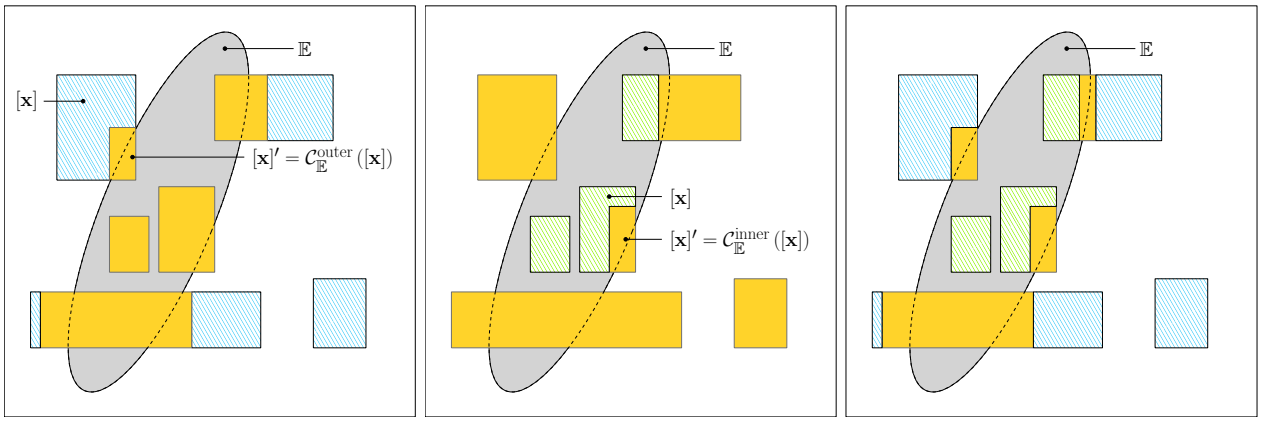


FIGURE 8.1 – Principe d'application d'un séparateur $\mathcal{S}_{\mathbb{E}} = \left\{\mathcal{C}_{\mathbb{E}}^{\text{inner}}, \mathcal{C}_{\mathbb{E}}^{\text{outer}}\right\}$ associé à un ensemble $\mathbb{E}$. *À gauche* : contractions extérieures avec $\mathcal{C}_{\mathbb{E}}^{\text{outer}}$. *Au centre* : contractions intérieures avec $\mathcal{C}_{\mathbb{E}}^{\text{inner}}$. *À droite* : boîtes séparées par des contractions intérieures/extérieures combinées.

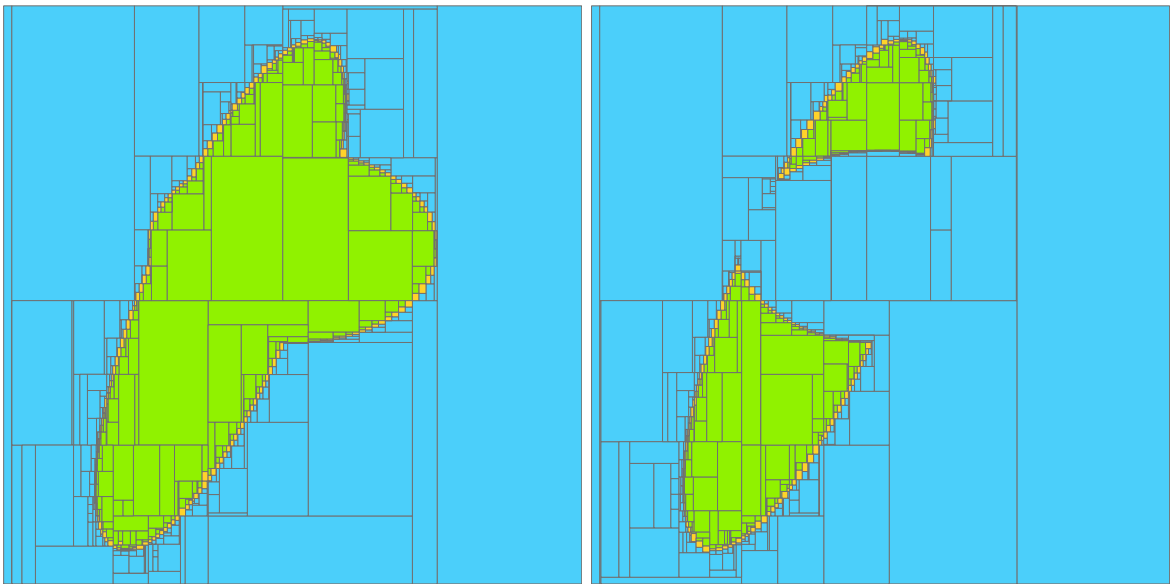
Remarque 32. *Une autre manière d'appréhender un séparateur est de le considérer comme l'extension d'une fonction implicite définissant un ensemble. Une fonction implicite est aussi une représentation indirecte qui permet de classer chaque point de l'espace comme étant intérieur, frontière ou extérieur de l'ensemble à représenter. Le séparateur permet d'étendre ces classificateurs à des boîtes. De manière plus intéressante, un séparateur permet aussi de séparer une boîte, c'est-à-dire d'identifier des parties complètement intérieures ou complètement extérieures par filtrage. Embarqué dans un arbre de recherche comme un paveur (voir section 2.2.4.2 en page 22), il permet de paver l'ensemble par un nombre fini de boîtes.*

8.2 Programmation par séparateurs

Depuis [8], un contracteur peut être considéré comme un ensemble. Notons $\text{set}(\mathcal{L})$ l'ensemble associé à une contrainte $\mathcal{L}$: $\text{set}(\mathcal{L}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathcal{L}(\mathbf{x}) \text{ est vrai}\}$. De la même manière, nous pouvons définir l'ensemble associé à un contracteur : $\text{set}(\mathcal{C}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathcal{C}(\{\mathbf{x}\}) = \{\mathbf{x}\}\}$, où $\{\mathbf{x}\}$ est un singleton, ou boîte dégénérée, qui n'est pas supprimé par l'opérateur. Nous rappelons que, si $\mathcal{C}$ est un contracteur pour la contrainte $\mathcal{L}$, alors $\text{set}(\mathcal{C}) \supset \text{set}(\mathcal{L})$. En particulier, si $\mathcal{C}$ est un contracteur minimal (optimal), alors $\text{set}(\mathcal{L}) = \text{set}(\mathcal{C})$. Ces relations impliquent qu'un contracteur peut être considéré comme un ensemble *équivalent* à la contrainte associée : $\text{set}(\mathcal{L}) \sim \mathcal{C}$. Par conséquent, il devient possible d'utiliser des opérateurs d'ensembles classiques sur les contracteurs afin de mettre en œuvre des opérations logiques. Considérons par exemple la conjonction $\mathcal{L}_3 = \mathcal{L}_1 \wedge \mathcal{L}_2$, alors : $\text{set}(\mathcal{L}_3) = \text{set}(\mathcal{L}_1 \wedge \mathcal{L}_2) = \text{set}(\mathcal{L}_1) \cap \text{set}(\mathcal{L}_2) \sim \mathcal{C}_1 \cap \mathcal{C}_2$.

Cette nouvelle approche de modélisation a permis de manipuler les contracteurs comme des ensembles,

c'est-à-dire avec les opérations habituelles sur les ensembles : intersection, union, projection, produit cartésien, etc. Cette approche s'applique également aux séparateurs [91]. Nous fournissons dans la figure 8.2 une illustration de la combinaison de séparateurs pour représenter des ensembles complexes avec des approximations intérieures et extérieures. La bibliothèque Codac [17] rassemble un catalogue de contracteurs et de séparateurs pour approximer les contraintes.



(a) Approximation par pavage d'une contrainte $\mathcal{E}_1 \vee \mathcal{E}_2$ représentée par le séparateur $(\mathcal{S}_{\mathbb{E}_1} \cup \mathcal{S}_{\mathbb{E}_2})$.

(b) Approximation par pavage de la contrainte $\mathcal{E}_1 \wedge \neg\mathcal{E}_2$ représentée par le séparateur $\left(\mathcal{S}_{\mathbb{E}_1} \cap \overline{\mathcal{S}_{\mathbb{E}_2}}\right)$.

FIGURE 8.2 – Exemple de combinaison de deux séparateurs $\mathcal{S}_{\mathbb{E}_1}$ et $\mathcal{S}_{\mathbb{E}_2}$ associés à deux contraintes ellipsoïdales $\mathcal{E}_1$ et $\mathcal{E}_2$, chacune étant une expression mathématique de la forme $\mathbf{h}(\mathbf{x}) < 0$. Un pavage muni des séparateurs résultants permet de révéler les ensembles associés.

Remarque 33. *L'ensemble solution d'une contrainte correspond à la définition en compréhension d'un ensemble, c'est-à-dire de type* $\{\mathbf{x} \mid P(\mathbf{x})\}$ *où le prédicat* P *est alors vu comme une contrainte. Ainsi, lorsque ce prédicat est traduisible en une version intervalle, il devient possible de définir un séparateur associé.*

8.3 Traduction du problème en programmation par séparateurs

Grâce à ce nouveau paradigme, nous pouvons donner un système d'équations entre séparateurs, vus comme des ensembles (à droite), qui permet de décrire les mêmes solutions que le système (7.13), rappelé à gauche.

$$\left\{\begin{array}{l} 1.\ \mathbf{c}^{\mathbb{A}} = \texttt{MinCircle}(\mathbb{A}), \\ 2.\ \mathbf{c}^{\mathbb{B}} = \texttt{MinCircle}(\mathbb{B}), \\ 3.\ \mathbb{A}_N = \left(\mathbb{A} - \mathbf{c}^{\mathbb{A}}_{\mathbf{12}}\right)/c^{\mathbb{A}}_3, \\ 4.\ \mathbb{B}_N = \left(\mathbb{B} - \mathbf{c}^{\mathbb{B}}_{\mathbf{12}}\right)/c^{\mathbb{B}}_3, \\ 5.\ \mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N, \\ 6.\ k = c^{\mathbb{B}}_3/c^{\mathbb{A}}_3, \\ 7.\ \mathbf{t} = \mathbf{c}^{\mathbb{B}}_{\mathbf{12}} - k\mathbf{R}_\theta \cdot \mathbf{c}^{\mathbb{A}}_{\mathbf{12}}. \end{array}\right. \sim \left\{\begin{array}{l} 1.\ \mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}} = \texttt{SepMinCircle}(\mathcal{S}_{\mathbb{A}}), \\ 2.\ \mathcal{S}_{\{\mathbf{c}^{\mathbb{B}}\}} = \texttt{SepMinCircle}(\mathcal{S}_{\mathbb{B}}), \\ 3.\ \mathcal{S}_{\mathbb{A}_N} = \left(\mathcal{S}_{\mathbb{A}} - \mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}_{12}}\right)/\mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}_3}, \\ 4.\ \mathcal{S}_{\mathbb{B}_N} = \left(\mathcal{S}_{\mathbb{B}} - \mathcal{S}_{\{\mathbf{c}^{\mathbb{B}}\}_{12}}\right)/\mathcal{S}_{\{\mathbf{c}^{\mathbb{B}}\}_3}, \\ 5.\ \mathcal{S}_{\mathbb{B}_N} = \mathbf{R}_{\mathcal{S}_\Theta} \cdot \mathcal{S}_{\mathbb{A}_N}, \\ 6.\ \mathcal{S}_{\mathbb{K}} = \mathcal{S}_{\{\mathbf{c}^{\mathbb{B}}\}_3}/\mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}_3}, \\ 7.\ \mathcal{S}_{\mathbb{T}} = \mathcal{S}_{\{\mathbf{c}^{\mathbb{B}}\}_{12}} - k\mathbf{R}_{\mathcal{S}_\Theta} \cdot \mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}_{12}}. \end{array}\right. \tag{8.1}$$

Remarque 34 (**Notations**). *Dans ce manuscrit, les objets utilisant le symbole calligraphique $\mathcal{S}$ désignent des séparateurs, c'est-à-dire des opérateurs agissant sur des boîtes. En revanche, les noms écrits en police à chasse fixe préfixées par* `Sep`*, comme* `SepMinCircle`*, désignent des procédures qui construisent et retournent de tels séparateurs.*

Le chapitre 9 explicite chacune de ces nouvelles équations afin d'en déduire des séparateurs pour chacune des variables du problème. En particulier, il est nécessaire de traduire la fonction `MinCircle` en une fonction `SepMinCircle` sur séparateurs. Celle-ci sera explicitée dans la section 9.1.

8.4 Séparateurs pour fonction paramétrée

Soient une fonction $\mathbf{h} : \mathbb{R}^l \times \mathbb{R}^m \mapsto \mathbb{R}^n$ et deux ensembles $\mathbb{X} \subset \mathbb{R}^l$ et $\mathbb{Z} \subset \mathbb{R}^n$. Nous présentons deux séparateurs différents issus de la section 4.2.2 de [90] pour déterminer l'ensemble des paramètres mettant en correspondance deux ensembles. Un séparateur « pour tout » pour l'inclusion, et un séparateur « il existe » pour l'intersection.

8.4.1 Séparateur « pour tout »

Nous cherchons les ensembles de paramètres qui permettent d'inclure un ensemble dans un autre, c'est-à-dire l'ensemble des paramètres $\mathbf{y}$ tels que l'inclusion $\mathbf{h}(\mathbb{X}, \mathbf{y}) \subset \mathbb{Z}$ est respectée. Nous considérons un ensemble $\mathbb{Y}$ défini de la manière suivante :

$$\mathbb{Y} = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{h}(\mathbb{X}, \mathbf{y}) \subset \mathbb{Z}\}. \tag{8.2}$$

À partir de cette définition, [90] montre qu'il est possible d'en déduire un séparateur :

$$\begin{aligned}
\mathbb{Y} &= \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{h}(\mathbb{X}, \mathbf{y}) \subset \mathbb{Z}\} \\
&= \{\mathbf{y} \in \mathbb{R}^m \mid \forall \mathbf{x} \in \mathbb{X},\, \mathbf{h}(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}\} \\
&= \overline{\left\{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{X},\, \mathbf{h}(\mathbf{x}, \mathbf{y}) \in \overline{\mathbb{Z}}\right\}} \\
&= \overline{\left\{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^l,\, (\mathbf{x}, \mathbf{y}) \in (\mathbb{X} \times \mathbb{R}^m) \cap \mathbf{h}^{-1}(\overline{\mathbb{Z}})\right\}} \\
&= \overline{\mathrm{proj}_{[l+1,l+m]}\left\{(\mathbb{X} \times \mathbb{R}^m) \cap \mathbf{h}^{-1}(\overline{\mathbb{Z}})\right\}} \\
&= \overline{\mathrm{proj}_{[l+1,l+m]}\left\{(\mathbb{X} \times \mathbb{R}^m) \cap \mathbf{h}^{-1}(\overline{\mathbb{Z}})\right\}},
\end{aligned} \tag{8.3}$$

où $\mathrm{proj}_{[l+1,l+m]}(\mathbb{Z})$ désigne la projection de $\mathbb{Z}$ sur ses composantes indexées par $i \in \{l+1, \ldots, l+m\}$, comme décrit dans l'équation (2.10), page 14.

De par l'algèbre des séparateurs [91], un séparateur $\mathcal{S}_{\mathbb{Y}}$ peut être obtenu à partir des séparateurs $\mathcal{S}_{\mathbb{X}}, \mathcal{S}_{\mathbb{Z}}$ pour $\mathbb{X}, \mathbb{Z}$:

$$\mathcal{S}_{\mathbb{Y}} = \overline{\mathrm{proj}_{[l+1,l+m]}\left\{(\mathcal{S}_{\mathbb{X}} \times \mathcal{S}_{\mathbb{R}^m}) \cap \mathbf{h}^{-1}(\overline{\mathcal{S}_{\mathbb{Z}}})\right\}}. \tag{8.4}$$

Nous verrons qu'un tel séparateur sera utilisé pour calculer les cercles minimaux $\mathbf{c}^{\mathbb{A}}$, $\mathbf{c}^{\mathbb{B}}$, les ensembles standardisés $\mathbb{A}_N$, $\mathbb{B}_N$ et le recalage par rotation.

Remarque 35. *Ce séparateur est très intéressant, puisqu'il permet de donner directement un séparateur pour $\mathbb{P}$, comme utilisé dans [90]. Nous rappelons que $\mathbb{P}$ est défini ainsi :*

$$\mathbb{P} = \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) = \mathbb{B}\} \tag{8.5}$$

Nous en déduisons alors :

$$\begin{aligned}
\mathbb{P} &= \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) = \mathbb{B}\} \\
&= \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) \subset \mathbb{B}\} \cap \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) \supset \mathbb{B}\} \\
&= \{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}(\mathbb{A}) \subset \mathbb{B}\} \cap \left\{\mathbf{p} \in \mathbb{S} \mid \mathbf{h_p}^{-1}(\mathbb{B}) \subset \mathbb{A}\right\} \\
&= \left\{\mathbf{p} = (\theta, \mathbf{t}, k)^\intercal \mid \mathbf{h}(\mathbb{A}, \mathbf{p}) \subset \mathbb{B}\right\} \cap \left\{\mathbf{p} = (\theta, \mathbf{t}, k)^\intercal \mid \mathbf{h}(\mathbb{B}, (-\theta, -\mathbf{t}, \mathbf{k^{-1}})^\intercal) \subset \mathbb{A}\right\}.
\end{aligned}$$

En pratique, appliquer ce séparateur est bien trop coûteux en raison de la dimension de $\mathbb{P}$. En utilisant une approche de type Procuste, nous montrons comment calculer le même ensemble en restant à petite dimension.

8.4.2 Séparateur « il existe »

Nous cherchons maintenant les ensembles de paramètres cohérents avec l'intersection non vide de deux ensembles $\mathbf{h}(\mathbb{X}, \mathbf{y}) \cap \mathbb{Z} \neq \emptyset$ est respectée. Nous considérons alors un ensemble $\mathbb{Y}'$ ayant pour définition :

$$\mathbb{Y}' = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{h}(\mathbb{X}, \mathbf{y}) \cap \mathbb{Z} \neq \emptyset\} \tag{8.6}$$

$$= \overline{\left\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{h}(\mathbb{X}, \mathbf{y}) \subset \overline{\mathbb{Z}}\right\}}. \tag{8.7}$$

Nous déduisons ainsi un séparateur $\mathcal{S}_{\mathbb{Y}'}$ de (8.4) et de (8.7) :

$$\mathcal{S}_{\mathbb{Y}'} = \mathrm{proj}_{[l+1,l+m]}\left\{(\mathcal{S}_{\mathbb{X}} \times \mathcal{S}_{\mathbb{R}^m}) \cap \mathbf{h}^{-1}(\mathcal{S}_{\mathbb{Z}})\right\}. \tag{8.8}$$

Nous verrons qu'un tel séparateur est aussi utilisé pour calculer les ensembles standardisés $\mathbb{A}_N$, $\mathbb{B}_N$.

8.5 Séparateur pour changement de coordonnées cartésiennes et polaires

Le passage des coordonnées cartésiennes (x, y) aux coordonnées polaires (ρ, θ) et réciproquement constitue une application pertinente de l'arithmétique intervalle et des séparateurs, en raison de la non-monotonicité des fonctions impliquées. En effet, ce changement de coordonnées est défini par :

$$\begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = \text{atan2}(y, x) \\ x = \rho \cos(\theta) \\ y = \rho \sin(\theta) \end{cases} \tag{8.9}$$

Dans la littérature, on retrouve plusieurs types de contracteurs et séparateurs basés sur ces équations :

- $\mathcal{C}_{\text{Polar}}$ [62], un contracteur minimal qui permet de garantir la cohérence entre les coordonnées polaires et cartésiennes d'un point de $\mathbb{R}^2$;
- $\mathcal{S}_{\text{Polar}}$ [90], la version séparateur du contracteur précédent ;
- $\mathcal{S}_{\text{PolarXY}}$, un séparateur de coordonnées cartésiennes paramétré par une boîte polaire $[\rho] \times [\theta]$. Embarqué dans un paveur, $\mathcal{S}_{\text{PolarXY}}$ permet de paver la boîte polaire $[\rho] \times [\theta]$ par des boîtes cartésiennes.

Nous proposons dans cette thèse un nouveau séparateur de coordonnées polaires, appelé $\mathcal{S}_{\text{CartPolar},\mathcal{S}_0}$, paramétré par un séparateur $\mathcal{S}_0$ associé à un ensemble en coordonnées cartésiennes. $\mathcal{S}_{\text{CartPolar},\mathcal{S}_0}$ est alors un séparateur de coordonnées polaires cohérent avec $\mathcal{S}_0$. Il nous permet de paver un ensemble quelconque $\mathcal{S}_0$ complet par des boîtes polaires.

Ce nouvel opérateur, implémenté dans l'algorithme 3, permettra donc d'enrichir le catalogue implémentant l'algèbre des séparateurs. Concrètement, afin de séparer une boîte polaire $\mathbf{u^P}$ à partir d'un séparateur en coordonnées cartésiennes $\mathcal{S}_0$, l'algorithme suit le schéma suivant :

1. passage de la boîte polaire en coordonnées cartésiennes grâce à $\mathcal{C}_{\text{Polar}}$,
2. séparation de la boîte selon $\mathcal{S}_0$,
3. passage des boîtes contractées en coordonnées polaires grâce à $\mathcal{C}_{\text{Polar}}$.

L'opérateur `CartHull` (resp. `PolarHull`) donne la boîte cartésienne (resp. polaire) enveloppe de la boîte polaire (resp. cartésienne) donnée en entrée. Ces opérateurs sont construits à l'aide du contracteur $\mathcal{C}_{\text{Polar}}$.

Algorithme 3 : Séparateur $\mathcal{S}_{\text{CartPolar},\mathcal{S}_0}$

Données : Séparateur cartésien $\mathcal{S}_0$
Entrée : Boîte polaire $[\mathbf{u^P}]$
Sortie : Boîtes polaires $[\mathbf{u_i^P}]$ et $[\mathbf{u_o^P}]$ contenant avec respectivement l'intérieur et l'extérieur

1 $[\mathbf{u^C}] \leftarrow$ `CartHull`$([\mathbf{u^P}])$
2 $([\mathbf{u_i^C}], [\mathbf{u_o^C}]) \leftarrow \mathcal{S}_0([\mathbf{u^C}])$
3 $[\mathbf{u_i^P}] \leftarrow [\mathbf{u^P}] \cap$ `PolarHull`$([\mathbf{u_i^C}])$
4 $[\mathbf{u_o^P}] \leftarrow [\mathbf{u^P}] \cap$ `PolarHull`$([\mathbf{u_o^C}])$
5 **retourner** $[\mathbf{u_i^P}], [\mathbf{u_o^P}]$

Chapitre 9

Résolution du problème

Plan du chapitre

Puisque les solutions du système (7.13) forme un ensemble, l'algèbre des ensembles et des séparateurs peut être mobilisée afin de le calculer.

Cette section détaille tous les séparateurs nécessaires à l'application de la méthode de résolution.

9.1 Séparateur pour le cercle minimum

Nous montrons d'abord comment définir un séparateur pour les cercles minimaux à l'étape 1 de notre approche illustrée dans la figure 7.1, p.98. Les cercles minimaux sont définis par les deux équations suivantes :

$$\begin{cases} 1.\ \mathbf{c}^{\mathbb{A}} = \texttt{MinCircle}(\mathbb{A}), \\ 2.\ \mathbf{c}^{\mathbb{B}} = \texttt{MinCircle}(\mathbb{B}). \end{cases} \tag{9.1}$$

Soit $\mathbb{C}^{\mathbb{A}}$ l'ensemble de tous les cercles englobants de $\mathbb{A}$. Le sous-ensemble $\mathbb{C}^{\mathbb{A}^\star} \subset \mathbb{C}^{\mathbb{A}}$ tel que le rayon est minimisé est défini comme suit :

$$\mathbb{C}^{\mathbb{A}^\star} = \left\{ \mathbf{c} \in \mathbb{C}^{\mathbb{A}} \mid c_r = \min_{\mathbf{c}' \in \mathbb{C}^{\mathbb{A}}} c'_r \right\}. \tag{9.2}$$

Cet ensemble définit le cercle minimal de $\mathbb{A}$ et, en raison de son unicité, il s'agit d'un singleton :

$$\mathbb{C}^{\mathbb{A}^\star} = \left\{ \mathbf{c}^{\mathbb{A}} \right\}. \tag{9.3}$$

Pour calculer $\mathbb{C}^{\mathbb{A}}$, nous commençons par utiliser la fonction de distance :

$$\begin{aligned} h_0 : \mathbb{R}^2 \times \mathbb{R}^3 &\to \mathbb{R} \\ (\mathbf{u}, \mathbf{c}) &\mapsto (u_1 - c_1)^2 + (u_2 - c_2)^2 - c_3^2 \end{aligned} \tag{9.4}$$

où $\mathbf{u}$ et $\mathbf{c}$ sont respectivement les coordonnées cartésiennes d'un point de $\mathbb{R}^2$, et les paramètres d'un cercle à savoir son centre (c_1, c_2) et son rayon c_3. En particulier, lorsque $h_0(\mathbf{u}, \mathbf{c}) \leq 0$, alors $\mathbf{u}$ se situe à l'intérieur du cercle $\mathbf{c}$ ou sur sa frontière.

Pour exprimer l'ensemble des cercles de rayon inférieur ou égal à c_3, nous avons alors :

$$\mathbb{C}^{\mathbb{A}} \triangleq \left\{ \mathbf{c} \in \mathbb{R}^3 \mid h_0(\mathbb{A}, \mathbf{c}) \subset [-\infty, 0] \right\} \tag{9.5}$$

Cette expression est de la forme (8.2), ce qui nous donne donc directement un séparateur pour $\mathbb{C}^{\mathbb{A}}$.

En projetant $\mathbb{C}^{\mathbb{A}}$ sur sa composante de rayon, on obtient l'intervalle $[c_3^{\mathbb{A}}, +\infty]$, où $\mathbf{c}^{\mathbb{A}}$ est le cercle englobant minimal de $\mathbb{A}$, puisque tout cercle concentrique à $\mathbf{c}^{\mathbb{A}}$ et de plus grand rayon est un cercle englobant. Pour être cohérent avec les intervalles fermés, sa négation donne l'intervalle $[-\infty, c_3^{\mathbb{A}}]$, donc :

$$c_3^{\mathbb{A}} = \mathrm{proj}_3(\mathbb{C}^{\mathbb{A}}) \cap \overline{\mathrm{proj}_3(\mathbb{C}^{\mathbb{A}})}. \tag{9.6}$$

L'algorithme 5 montre une implémentation possible d'un séparateur pour le cercle minimal d'un ensemble.

9.2 Séparateur pour les ensembles standardisés

Les ensembles standardisés, utilisés à l'étape 2 de notre méthode illustrée dans la figure 7.1, p.98, peuvent être définis à partir des ensembles initiaux et des cercles minimaux, comme le montrent les équations décrites dans le système (7.13) :

$$\begin{cases} 3.\ \mathbb{A}_N = \left(\mathbb{A} - \mathbf{c}_{\mathbf{12}}^{\mathbb{A}}\right) / c_3^{\mathbb{A}}, \\ 4.\ \mathbb{B}_N = \left(\mathbb{B} - \mathbf{c}_{\mathbf{12}}^{\mathbb{B}}\right) / c_3^{\mathbb{B}}, \end{cases} \tag{9.7}$$

Considérons la fonction suivante :

$$\mathbf{h_1}(\mathbf{c}, \mathbf{a_n}) = \mathbf{a_n} * c_3 + \mathbf{c_{12}}. \tag{9.8}$$

Nous pouvons réécrire la définition de $\mathbb{A}_N$ en utilisant cette fonction :

$$\mathbb{A}_N = \left\{ \mathbf{a_n} \in \mathbb{R}^2 \mid \mathbf{h_1}(\mathbf{c}^{\mathbb{A}}, \mathbf{a_n}) \in \mathbb{A} \right\}. \tag{9.9}$$

Comme $\left\{\mathbf{c}^{\mathbb{A}}\right\}$ est un singleton et la fonction $\mathbf{a_n} \mapsto \mathbf{h_1}(\mathbf{c}^{\mathbb{A}}, \mathbf{a_n})$ est une bijection, alors :

$$\mathbb{A}_N = \left\{\mathbf{a_n} \in \mathbb{R}^2 \mid \mathbf{h_1}\left(\left\{\mathbf{c}^{\mathbb{A}}\right\}, \mathbf{a_n}\right) \subset \mathbb{A}\right\} \tag{9.10}$$

$$= \left\{\mathbf{a_n} \in \mathbb{R}^2 \mid \mathbf{h_1}\left(\left\{\mathbf{c}^{\mathbb{A}}\right\}, \mathbf{a_n}\right) \cap \mathbb{A} \neq \emptyset\right\} \tag{9.11}$$

Ces équations sont respectivement analogues aux équations (8.2) et (8.6), ce qui nous permet de construire deux séparateurs pour $\mathbb{A}_N$:

$$\mathcal{S}^i_{\mathbb{A}_N} = \overline{\text{proj}_{\mathbf{23}}\left\{(\mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}} \times \mathcal{S}_{\mathbb{R}^2}) \cap \mathbf{h_1^{-1}}(\overline{\mathcal{S}_{\mathbb{A}}})\right\}}, \tag{9.12}$$

$$\mathcal{S}^o_{\mathbb{A}_N} = \text{proj}_{\mathbf{23}}\left\{(\mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}} \times \mathcal{S}_{\mathbb{R}^2}) \cap \mathbf{h_1^{-1}}(\mathcal{S}_{\mathbb{A}})\right\}. \tag{9.13}$$

Puisque $\{\mathbf{c}^{\mathbb{A}}\}$ est un singleton, seul le contracteur extérieur de $\mathcal{S}_{\{\mathbf{c}^{\mathbb{A}}\}}$ peut filtrer. Ainsi, le séparateur $\mathcal{S}^o_{\mathbb{A}_N}$ (9.13) ne peut contracter que l'extérieur de $\mathbb{A}_N$. Nous obtenons alors $\mathcal{S}^o_{\mathbb{A}_N} = (\mathcal{C}_{\mathbb{R}^2}, \mathcal{C}^o_{\mathbb{A}_N})$.

De manière similaire, $\mathcal{S}^i_{\mathbb{A}_N}$ (9.12) ne peut contracter que l'extérieur de $\overline{\mathbb{A}_N}$, soit l'intérieur de $\mathbb{A}_N$. Nous obtenons alors $\mathcal{S}^i_{\mathbb{A}_N} = (\mathcal{C}^i_{\mathbb{A}_N}, \mathcal{C}_{\mathbb{R}^2})$.

Le séparateur $\mathcal{S}_{\mathbb{A}_N}$ choisi dans cette contribution se définit ainsi comme la paire de contracteurs suivante :

$$\mathcal{S}_{\mathbb{A}_N} = (\mathcal{C}^i_{\mathbb{A}_N}, \mathcal{C}^o_{\mathbb{A}_N}). \tag{9.14}$$

L'algorithme qui permet de construire un tel séparateur est appelé `SepPair`$(\mathcal{S}^i_{\mathbb{A}_N}, \mathcal{S}^o_{\mathbb{A}_N})$.

9.3 Séparateur pour la mise en correspondance par rotation

La mise en correspondance par rotation, c'est-à-dire l'étape 3 de notre approche (cf. figure 7.1) est couverte par la relation 5. du système (7.13) :

$$5.\ \mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N. \tag{9.15}$$

Rappelons l'ensemble Θ de toutes les rotations satisfaisant cette relation :

$$\Theta := \{\theta \mid \mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N\}. \tag{7.5}$$

Afin de décrire Θ, nous exprimons sa définition (7.5) en coordonnées polaires. L'avantage d'utiliser les coordonnées polaires est que la rotation est une somme, comme nous l'avons introduit dans la première contribution de ce manuscrit en section 4.2.2 (page 61) :

$$\mathbf{R}_\theta \cdot \begin{pmatrix}\rho\\ \phi\end{pmatrix} = \begin{pmatrix}\rho\\ \phi + \theta\end{pmatrix}, \tag{9.16}$$

où $\begin{pmatrix}\rho\\ \phi\end{pmatrix}$ est une coordonnée polaire module-argument arbitraire.

Le séparateur $\mathcal{S}_{\text{CartPolar}}$, décrit dans l'algorithme 3, est utilisé pour faire le passage de coordonnées cartésiennes en coordonnées polaires.

En utilisant $\mathbb{A}_N^P$ et $\mathbb{B}_N^P$ comme version polaire de $\mathbb{A}_N$ et $\mathbb{B}_N$, nous avons :

$$(\mathbb{B}_N = \mathbf{R}_\theta \cdot \mathbb{A}_N) \iff (\mathbb{A}_N^P + (0,\theta)^\intercal = \mathbb{B}_N^P). \tag{9.17}$$

Considérons maintenant les deux fonctions suivantes :

$$\mathbf{h_2^+}(\mathbf{u},\theta) = \mathbf{u} + (0,\theta)^\intercal, \tag{9.18}$$
$$\mathbf{h_2^-}(\mathbf{u},\theta) = \mathbf{u} - (0,\theta)^\intercal. \tag{9.19}$$

Cela nous permet de déduire :

$$\Theta := \left\{\theta \in \mathbb{R}/(2\pi\mathbb{Z}) \mid \mathbf{h_2^+}(\mathbb{A}_N^P,\theta) = \mathbb{B}_N^P\right\}. \tag{9.20}$$

En décomposant l'égalité en une double inclusion, nous retrouvons une forme familière :

$$\Theta = \left\{\theta \in \mathbb{R}/(2\pi\mathbb{Z}) \mid \left(\mathbf{h_2^+}(\mathbb{A}_N^P,\theta) \subset \mathbb{B}_N^P\right) \wedge \left(\mathbf{h_2^-}(\mathbb{B}_N^P,\theta) \subset \mathbb{A}_N^P\right)\right\}. \tag{9.21}$$

En effet, on retrouve deux fois une formulation de la forme (8.2). On en déduit alors un séparateur pour Θ :

$$\mathcal{S}_\Theta = \overline{\text{proj}_3\left\{(\mathcal{S}_{\mathbb{A}_N^P} \times \mathcal{S}_\mathbb{R}) \cap \mathbf{h_2^+}^{-1}(\overline{\mathcal{S}_{\mathbb{B}_N^P}})\right\}} \cap \overline{\text{proj}_3\left\{(\mathcal{S}_{\mathbb{B}_N^P} \times \mathcal{S}_\mathbb{R}) \cap \mathbf{h_2^-}^{-1}(\overline{\mathcal{S}_{\mathbb{A}_N^P}})\right\}} \tag{9.22}$$

9.4 Séparateurs pour contraintes classiques

Les équations 6. et 7. du système (7.13) sont de simples évaluations de fonctions qui impliquent des opérations classiques, y compris cos et sin pour la rotation, qui sont toutes implémentées dans la bibliothèque intervalles IBEX [10].

$$\begin{cases} 6.\ k = c_3^\mathbb{B}/c_3^\mathbb{A}, \\ 7.\ \mathbf{t} = \mathbf{c_{12}^\mathbb{B}} - k\mathbf{R}_\theta \cdot \mathbf{c_{12}^\mathbb{A}}. \end{cases} \tag{9.23}$$

9.5 Algorithmes de résolution

L'algorithme 4 montre les étapes de l'approche Procuste présentée dans la figure 7.1 et est directement basé sur le système (7.13). L'algorithme 5 décrit le calcul de tous les séparateurs définis dans ce chapitre. La section 9.5.2 précise comment les séparateurs et les algorithmes de pavage sont entrelacés.

9.5.1 Approche Procuste intervalle implémentée par des séparateurs

Dans l'algorithme 4, nous pouvons remarquer qu'aux lignes 6 et 7, seuls les rayons et les centres sont utilisés. Ceux-ci sont obtenus à partir des cercles minimaux avec des projections non détaillées ici.

L'algorithme 5 décrit le calcul de tous les séparateurs définis dans ce chapitre. Comme précisé dans la section 9.2, le séparateur retourné par `SepPair`$(\mathcal{S}_1, \mathcal{S}_2)$ présent à la ligne 11 prend le premier séparateur pour effectuer une contraction interne, et le second pour une contraction externe. Les séparateurs $\mathcal{S}_{\text{CartPolar},\mathcal{S}_{\mathbb{A}_N}}$ et $\mathcal{S}_{\text{CartPolar},\mathcal{S}_{\mathbb{B}_N}}$ sont décrits dans l'algorithme 3, et permettent de transformer un séparateur d'un ensemble exprimé en coordonnées cartésiennes en un séparateur décrivant le même ensemble en coordonnées polaires.

Algorithme 4 : Recalage par cercles minimums

Données : Separateurs $\mathcal{S}_{\mathbb{A}}$ et $\mathcal{S}_{\mathbb{B}}$ représentant les ensembles en entrée.
Résultat : Séparateur $\mathcal{S}_{\mathbb{P}}$ pour les paramètres de transformation $(\theta, \mathbf{t}, k)$.

`// 1. Cercles minimums de` $\mathbb{A}$ `et` $\mathbb{B}$
1 $\mathcal{S}_{\mathbf{c}^{\mathbb{A}}} \leftarrow$ `SepMinCircle`$(\mathcal{S}_{\mathbb{A}})$
2 $\mathcal{S}_{\mathbf{c}^{\mathbb{B}}} \leftarrow$ `SepMinCircle`$(\mathcal{S}_{\mathbb{B}})$

`// 2. Normalisation`
3 $\mathcal{S}_{\mathbb{A}_N} \leftarrow$ `SepNormalized`$(\mathcal{S}_{\mathbb{A}}, \mathcal{S}_{\mathbf{c}^{\mathbb{A}}})$
4 $\mathcal{S}_{\mathbb{B}_N} \leftarrow$ `SepNormalized`$(\mathcal{S}_{\mathbb{B}}, \mathcal{S}_{\mathbf{c}^{\mathbb{B}}})$

`// 3. Identification des paramètres`
5 $\mathcal{S}_{\Theta} \leftarrow$ `SepRotation`$(\mathcal{S}_{\mathbb{A}_N}, \mathcal{S}_{\mathbb{B}_N})$
6 $\mathcal{S}_{\mathbb{K}} \leftarrow$ `SepScaling`$(\mathcal{S}_{c_3^{\mathbb{A}}}, \mathcal{S}_{c_3^{\mathbb{B}}})$

`// 4. Composition du séparateur des paramètres`
7 $\mathcal{S}_{\mathbb{T}} \leftarrow$ `SepTranslation`$(\mathcal{S}_{\mathbf{c}_{\mathbf{12}}^{\mathbb{A}}}, \mathcal{S}_{\mathbf{c}_{\mathbf{12}}^{\mathbb{B}}}, \mathcal{S}_{\mathbb{K}}, \mathcal{S}_{\Theta})$
8 $\mathcal{S}_{\mathbb{P}} \leftarrow \mathcal{S}_{\Theta} \times \mathcal{S}_{\mathbb{T}} \times \mathcal{S}_{\mathbb{K}}$
9 **retourner** $\mathcal{S}_{\mathbb{P}}$

9.5.2 Intégration de $\mathcal{S}_{\mathbb{P}}$ dans un algorithme de pavage

Le séparateur $\mathcal{S}_{\mathbb{P}}$ obtenu dans les algorithmes 4 et 5 fournit la capacité de contraction nécessaire pour caractériser les quatre paramètres de transformation $(\theta, \mathbf{t}, k)$. Cependant, l'intégration de ce séparateur dans un algorithme naïf de pavage « brancher et séparer » sur l'espace de recherche complet à quatre dimensions $\mathbb{S}$ entraîne une charge de calcul importante en raison de la dimension élevée. Pour surmonter ce problème, une première amélioration a consisté à proposer une stratégie de branchement spécifique afin d'améliorer les performances.

Au lieu d'effectuer des bissections sur le domaine des paramètres, dans les expériences présentées ci-après nous utilisons une stratégie combinatoire plus efficace qui reflète la décomposition de type Procuste du problème. Cette approche décompose la recherche en 4 dimensions en une séquence coordonnée de sous-problèmes de dimension inférieure :

1. **Caractérisation des descripteurs :** Un processus de pavage dédié effectue d'abord une approximation des descripteurs des ensembles d'entrée (rayons et centres des cercles minimums) en

Algorithme 5 : Définition des séparateurs

Fonction `SepMinCircle`$(\mathcal{S}_{\mathbb{X}})$
  $h_0 \leftarrow (\mathbf{x}, \mathbf{c}) \mapsto (x_1 - c_1)^2 + (x_2 - c_2)^2 - c_3^2$
  $\mathcal{S}_1 \leftarrow \neg\text{proj}_{\mathbf{345}} \left\{ (\mathcal{S}_{\mathbb{X}} \times \mathcal{S}_{\mathbb{R}^3}) \cap \mathbf{h_0^{-1}}([0, \infty]) \right\}$
  $\mathcal{S}_2 \leftarrow \text{proj}_3 (\mathcal{S}_1)$
  $\mathcal{S}_3 \leftarrow \mathcal{S}_2 \cap \neg\mathcal{S}_2$
  retourner $\mathcal{S}_1 \cap (\mathcal{S}_{\mathbb{R}^2} \times \mathcal{S}_3)$
Fonction `SepNormalized`$(\mathcal{S}_{\mathbb{X}}, \mathcal{S}_{\mathbf{c}^{\mathbb{X}}})$
  $\mathbf{h_1} \leftarrow (\mathbf{c}, \mathbf{x_n}) \mapsto \mathbf{x_n} \cdot c_3 + \mathbf{c_{12}}$
  $\mathcal{S}_1 \leftarrow \neg\text{proj}_{\mathbf{45}} \left\{ (\mathcal{S}_{\mathbf{c}^{\mathbb{X}}} \times \mathcal{S}_{\mathbb{R}^2}) \cap \mathbf{h_1^{-1}}(\neg\mathcal{S}_{\mathbb{X}}) \right\}$
  $\mathcal{S}_2 \leftarrow \text{proj}_{\mathbf{45}} \left\{ (\mathcal{S}_{\mathbf{c}^{\mathbb{X}}} \times \mathcal{S}_{\mathbb{R}^2}) \cap \mathbf{h_1^{-1}}(\mathcal{S}_{\mathbb{X}}) \right\}$
  retourner `SepPair`$(\mathcal{S}_1, \mathcal{S}_2)$
Fonction `SepRotation`$(\mathcal{S}_{\mathbb{A}_N}, \mathcal{S}_{\mathbb{B}_N})$
  $\mathbf{h_2^+} \leftarrow (\mathbf{y}, \theta) \mapsto \mathbf{y} + (0, \theta)^\intercal$
  $\mathbf{h_2^-} \leftarrow (\mathbf{y}, \theta) \mapsto \mathbf{y} - (0, \theta)^\intercal$
  $\mathcal{S}_1 \leftarrow \neg\text{proj}_3 \left\{ (\mathcal{S}_{\text{CartPolar}, \mathcal{S}_{\mathbb{A}_N}} \times \mathcal{S}_{\mathbb{R}}) \cap \mathbf{h_2^{+-1}}(\neg\mathcal{S}_{\text{CartPolar}, \mathcal{S}_{\mathbb{B}_N}}) \right\}$
  $\mathcal{S}_2 \leftarrow \neg\text{proj}_3 \left\{ (\mathcal{S}_{\text{CartPolar}, \mathcal{S}_{\mathbb{B}_N}} \times \mathcal{S}_{\mathbb{R}}) \cap \mathbf{h_2^{--1}}(\neg\mathcal{S}_{\text{CartPolar}, \mathcal{S}_{\mathbb{A}_N}}) \right\}$
  retourner $\mathcal{S}_1 \cap \mathcal{S}_2$
Fonction `SepScaling`$(\mathcal{S}_{c_3^{\mathbb{A}}}, \mathcal{S}_{c_3^{\mathbb{B}}})$
  $h_3 \leftarrow (c_3^{\mathbb{A}}, c_3^{\mathbb{B}}) \mapsto c_3^{\mathbb{B}} / c_3^{\mathbb{A}}$
  retourner $h_3(\mathcal{S}_{c_3^{\mathbb{A}}}, \mathcal{S}_{c_3^{\mathbb{B}}})$
Fonction `SepTranslation`$(\mathcal{S}_{\mathbf{c_{12}^{\mathbb{A}}}}, \mathcal{S}_{\mathbf{c_{12}^{\mathbb{B}}}}, \mathcal{S}_{\mathbb{K}}, \mathcal{S}_{\Theta})$
  $\mathbf{h_4} \leftarrow (\mathbf{c_{12}^{\mathbb{A}}}, \mathbf{c_{12}^{\mathbb{B}}}, k, \theta) \mapsto \mathbf{c_{12}^{\mathbb{B}}} - k \cdot \mathbf{R}_\theta \, \mathbf{c_{12}^{\mathbb{A}}}$
  retourner $\mathbf{h_4}(\mathcal{S}_{\mathbf{c_{12}^{\mathbb{A}}}}, \mathcal{S}_{\mathbf{c_{12}^{\mathbb{B}}}}, \mathcal{S}_{\mathbb{K}}, \mathcal{S}_{\Theta})$

invoquant `SepMinCircle`.

2. **Recherche angulaire :** Le paramètre de rotation θ est ensuite résolu via un pavage unidimensionnel. Cette étape utilise de manière récursive les séparateurs de normalisation $\mathcal{S}_{\mathbb{A}_N}$ et $\mathcal{S}_{\mathbb{B}_N}$ (où les ensembles normalisés peuvent être considérés comme des variables intermédiaires bidimensionnelles) afin de maintenir la cohérence avec les descripteurs précédemment bornés.
3. **Extension analytique :** Enfin, la mise à l'échelle k et la translation $\mathbf{t}$ sont calculées par des évaluations directes d'intervalles, en tirant parti des encadrements obtenus lors des étapes précédentes.

Cette décomposition hiérarchique réduit considérablement le nombre de bissections requises, assurant ainsi que le séparateur global $\mathcal{S}_{\mathbb{P}}$ reste calculable tout en garantissant de capturer toutes les solutions cohérentes.

Chapitre 10

Expérimentations et conclusions

Plan du chapitre

Dans ce chapitre, nous expérimentons notre approche sur différents exemples. Nous étudions notamment comment elle se comporte sur des problèmes avec différents niveaux de symétries, ainsi que sur des cas dégénérés.

10.1 Benchmark

Les séparateurs pour les ensembles d'entrée $\mathbb{A}$ et $\mathbb{B}$ sont obtenus à partir de deux images différentes, à une résolution de 4000×4000 pixels. Le benchmark retenu est obtenu à partir des images présentées dans la figure 10.1. Il contient les jeux de données suivants.

— Le **Domaine d'O** est un cas typique, sans symétrie, obtenu en utilisant une version nettoyée de la vue satellite du Domaine d'O à Montpellier, en France. Cet exemple particulier est un peu délicat car sa forme générale allongée rend l'estimation du centre du cercle minimum moins précise.

— La **croix** est une forme géométrique avec quatre solutions différentes.

— La **croix étirée** est une version étirée horizontalement de la croix précédente, qui ne donne que deux solutions, mais où il peut s'avérer difficile d'écarter les deux non-solutions.

— Le **triskel** est une forme non-convexe et non-connexe avec trois solutions.

— Les **anneaux concentriques** sont un cas limite présentant une infinité de solution de par leur forme de révolution. Toute rotation peut être étendue en une solution.

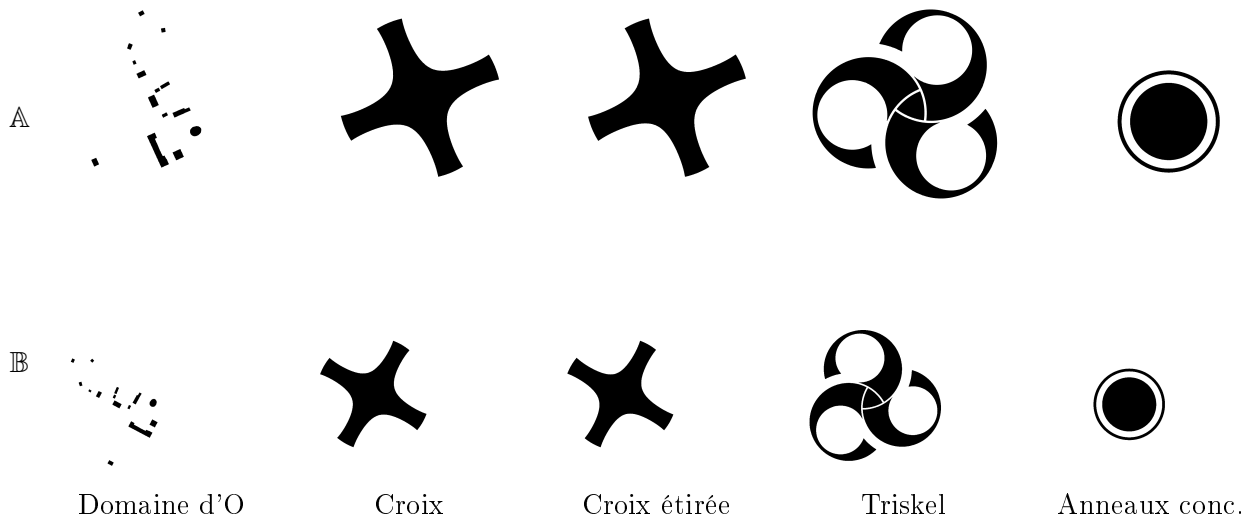


FIGURE 10.1 – Paires d'images sources utilisées comme référence. La première ligne représente les ensembles $\mathbb{A}$, et la deuxième ligne montre les ensembles $\mathbb{B}$.

10.2 Expérimentations

Le benchmark a été traité à l'aide de la séquence hiérarchique d'algorithmes de pavage décrite dans la section 9.5.2, en implémentant les séparateurs des algorithmes 4 et 5.

Un paramètre de précision global $\varepsilon = 10^{-3}$ a été défini comme critère d'arrêt pour tous les processus de sous-pavage. Ce seuil est appliqué à la fois aux principales boucles de branchement et aux mécanismes internes des séparateurs de projection.

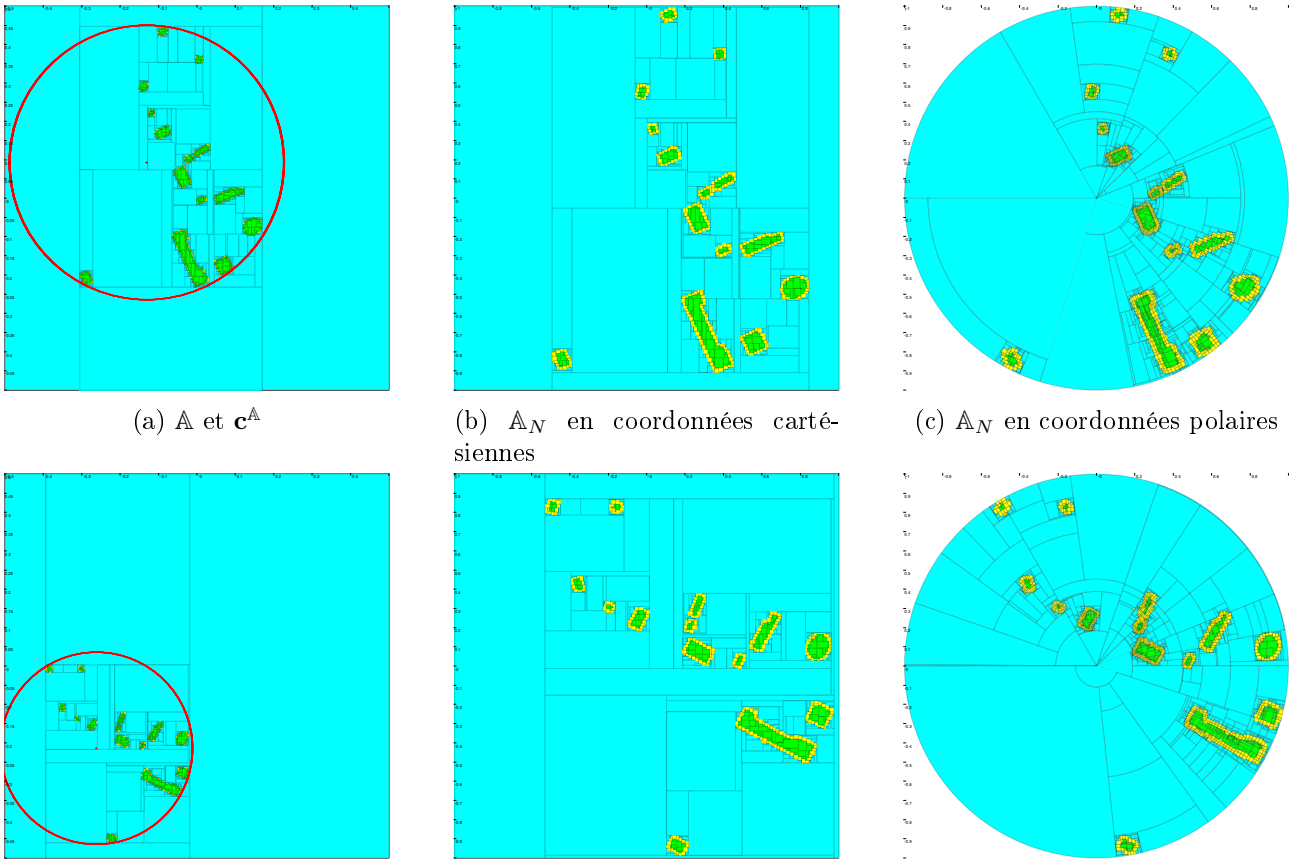
10.2.1 Analyse des résultats

La figure 10.2 illustre les approximations des ensembles intermédiaires clés pour le cas *Domaine d'O* : $\mathbb{A}$, $\mathbb{B}$, $\mathbb{A}_N$ et $\mathbb{B}_N$.

Les résultats de la recherche angulaire sont présentés dans la figure 10.3. Chaque région bleue représente un ensemble d'angles dont il a été rigoureusement prouvé qu'ils ne contiennent aucune rotation valide pour le problème de recalage.

Notons que dans les cas *croix* et *triskel*, la figure révèle respectivement quatre et trois boîtes de solution distinctes, capturant parfaitement les symétries de rotation discrètes de ces formes. Ces résultats soulignent la nature exhaustive de l'approche : alors que les méthodes de recherche locale peuvent se retrouver piégées dans l'un des multiples minima de ces cas, notre pavage basé sur des séparateurs récupère l'ensemble complet des rotations cohérentes.

Pour la *croix étirée*, l'algorithme identifie avec succès exactement deux boîtes de solution isolées. Malgré la proximité géométrique avec la *croix* standard et sa quadruple symétrie, l'approche basée sur les séparateurs rejette rigoureusement les deux orientations « parasites » prouvant qu'elles sont incompatibles, à la précision prescrite ε.



(a) $\mathbb{A}$ et $\mathbf{c}^{\mathbb{A}}$ (b) $\mathbb{A}_N$ en coordonnées cartésiennes (c) $\mathbb{A}_N$ en coordonnées polaires

(d) $\mathbb{B}$ et $\mathbf{c}^{\mathbb{B}}$ (e) $\mathbb{B}_N$ en coordonnées cartésiennes (f) $\mathbb{B}_N$ en coordonnées polaires

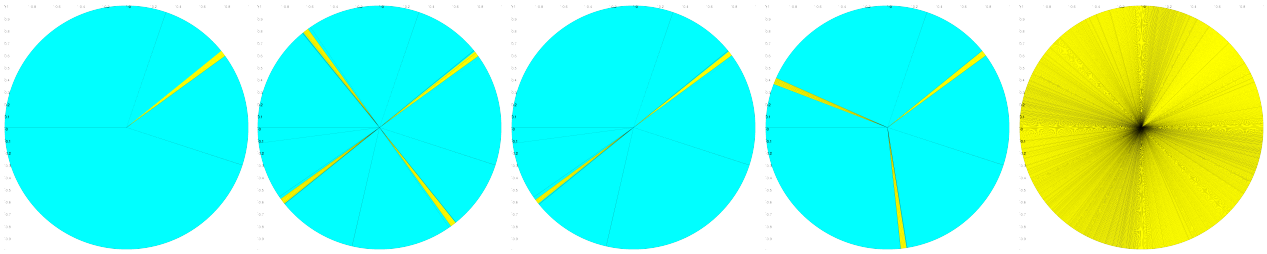
FIGURE 10.2 – Visualisation des ensembles clés de la résolution pour le Domaine d'O, obtenus à partir de leurs séparateurs respectifs dans un paveur.

Le cas des *anneaux concentriques* donne lieu à une bande continue d'espace, reflétant correctement le nombre infini de solutions inhérentes aux formes de révolution.

Le tableau 10.1 résume les performances numériques de l'algorithme pour une précision fixe $\varepsilon = 10^{-3}$. Il détaille le temps de calcul total et les incertitudes résultantes pour chaque paramètre estimé : les descripteurs des cercles minimums (rayons et centres), l'angle de rotation θ et le vecteur de translation $\mathbf{t}$.

Les résultats démontrent que la stratégie hiérarchique maintient un encadrement précis de la solution, même pour des formes complexes. Pour les cas symétriques tels que la *croix* ou le *triskel*, chaque ligne représente une région cohérente isolée dans l'espace des paramètres, confirmant la capacité de l'algorithme à distinguer tous les minima globaux. Dans le cas des anneaux concentriques, les incertitudes de rotation et de translation sont marquées par « – », car l'ensemble des solutions est une variété continue plutôt qu'un ensemble discret de points.

Les résultats permettent d'obtenir plusieurs observations intéressantes.



(a) Domaine d'O (b) Croix (c) Croix étirée (d) Triskel (e) Anneaux conc.

FIGURE 10.3 – Solutions de rotation(s), isolées dans des approximations externes représentées par des parts de cercle jaunes.

Banc d'essai	$c_3^{\mathbb{A}}$	$\mathbf{c_{12}^{\mathbb{A}}}$	$c_3^{\mathbb{B}}$	$\mathbf{c_{12}^{\mathbb{B}}}$	θ	**t**	**Temps (s)**
Domaine d'O	0.905	5.54	0.968	4.65	7.39	11.5	1.30
Croix	0.994	2.81	1.00	2.98	5.50 5.30 5.56 6.57	7.54 7.78 7.57 8.24	13.6
Croix étirée	0.995	2.58	0.997	2.90	5.63 5.60	7.66 7.65	7.20
Triskel	0.998	3.46	1.00	3.11	6.59 6.42 6.61	8.58 8.59 8.95	13.1
Anneaux concentriques	1.00	2.00	0.974	2.07	–	–	21.7

TABLE 10.1 – **Précision et performances de calcul.** Incertitudes mesurées sur les paramètres de transformation pour $\varepsilon = 10^{-3}$. Les incertitudes des rayons, centres et translation sont exprimées en millièmes de la taille de l'image ; l'incertitude angulaire θ est exprimée en millième de la révolution complète. Les lignes multiples par benchmark indiquent des groupes de solutions isolés distincts trouvés par l'algorithme de pavage.

Incertitudes croissantes Rappelons que le calcul des paramètres (par séparation) se font dans l'ordre suivant : rayons – et donc mise à l'échelle – puis centres des cercles minimaux, rotation et enfin translation. Chacune de ces estimations est basée sur les précédentes, ce qui a pour effet d'accumuler les incertitudes au fur et à mesure, comme on peut le constater dans le tableau 10.1.

Estimation du centre Le Domaine d'O illustre bien comment le centre du cercle minimal est estimé par les séparateurs : cela revient à utiliser les points de l'ensemble situés sur le cercle minimal et le rayon préalablement obtenu afin de trianguler la position du centre. Ainsi, dans des cas dégénérés où l'ensemble est allongé, et ne comporte que deux points diamétralement opposés sur le cercle, cela revient à estimer l'intersection entre deux cercles tangents, moyennant les incertitudes sur leur rayon et centre. Dans le cas du Domaine d'O, même si trois points sont bel et bien sur le cercle, ils ne sont pas très bien répartis sur le cercle, ce qui est illustré dans la figure 10.2. Au contraire, pour les anneaux concentriques chaque point de la frontière extérieure est aussi un point du cercle minimal. Ainsi chaque point du cercle minimal est disponible pour trianguler la position de son centre.

Estimation des paramètres de la transformation La figure 10.3 donne une représentation plus visuelle de l'ensemble des rotations réalisables. En ce qui concerne la translation, le séparateur `SepTranslation` (voir algorithme 5) utilisé permet de coupler une solution possible de translation avec une rotation compatible, ce qui permet d'avoir une information plus riche que des domaines indépendants induits par le produit cartésien $\Theta \times \mathbb{T}$.

Infinité de solutions Les *anneaux concentriques* soulignent le cas limite, avec une infinité de solutions possibles. Dans ce cas, les séparateurs ne permettent pas d'écarter des non solutions de rotation, ce qui reste coûteux en temps.

10.3 Conclusions

En conclusion, la méthode proposée pour le recalage d'ensembles se distingue des approches probabilistes classiques par son caractère garanti et exhaustif, offrant une approche robuste capable d'éliminer des configurations incohérentes sans omettre aucune solution.

Cette méthode tire parti de la puissance expressive des séparateurs. Le paradigme des séparateurs est élégant et puissant car un séparateur peut être construit (presque) directement sur l'expression analytique des ensembles impliqués dans les contraintes définissant le problème.

Ainsi, notre approche applique une version ensembliste des étapes de standardisation des ensembles proposées par la méthode de Procuste et ce, de manière garantie. Ceci lui permet à la fois :

— d'atteindre les objectifs de garantie et d'exhaustivité attendues d'une méthode ensembliste,
— tout en bénéficiant des avantages de diriger la résolution dans un certain ordre et
— de se ramener à une série de problèmes en plus petite dimension bien moins coûteux en calcul.

10.4 Perspectives

Notre méthode est pour le moment limitée à des ensembles bornés en dimension 2. L'extension à la 3ème dimension étendrait grandement son domaine d'application. Le seul verrou est le calcul de la rotation. En effet, même en coordonnées sphériques, la rotation devient plus complexe car :

— elle ne se réduit plus à une simple somme,
— elle devient tridimensionnelle (en nombre de variables) au lieu d'unidimensionnelle sur le plan.

Par ailleurs, il serait aussi intéressant de traiter ce problème comme un CSP, c'est-à-dire avec des variables ensemblistes ayant pour domaine des ensembles épais, voir section 2.5 page 30. Dans notre première contribution décrite au chapitre 5, nous avons vu comment implémenter ces ensembles épais avec des pavages épais réguliers. Pour étendre naturellement les travaux de cette seconde contribution, une idée intéressante serait de représenter ces domaines avec une paire de séparateurs, un pour chaque borne de l'ensemble épais. Une telle paire de séparateurs forme ce que l'on appelle un **séparateur épais** dans [92]. Un tel opérateur permettrait donc de filtrer à la fois les paramètres de la transformation et les ensembles, et donc de se présenter comme un opérateur de filtrage pour une contrainte « hybride »

faisant intervenir des ensembles et des réels :

$$\begin{cases} \textbf{Variables :} \ \mathbb{A},\ \mathbb{B},\ k,\ \theta,\ \mathbf{t} \\ \textbf{Domaines :} \ [\mathbb{A}],\ [\mathbb{B}],\ [k],\ \Theta,\ \mathbb{T} \\ \textbf{Contrainte :} \\ \quad 1. \quad \mathbb{B} = k\mathrm{R}_\theta\mathbb{A} + \mathbf{t} \end{cases} \tag{10.1}$$

Conclusion

Dans cette thèse, nous avons présenté des approches ensemblistes afin de traiter le problème de recalage en dimension 2. Nous avons vu que ce problème intervient dans de nombreux domaines, en particulier en robotique d'exploration puisqu'il permet de déduire un déplacement à partir de deux prises de vues. Contrairement au formalisme probabiliste, les résolutions ensemblistes permettent de garantir le résultat retourné.

En première partie, nous avons exploré d'une part le cadre CSP et plus précisément intervalle afin d'offrir une alternative au formalisme probabiliste. Nous avons introduit l'analyse intervalle, en soulignant notamment l'extension de l'arithmétique réelle aux intervalles de réels, les tubes, et l'extension de l'algèbre des parties d'un ensemble aux ensembles épais. Avec ce formalisme puissant, certains problèmes peuvent être exprimés comme un CSP, puis résolus de manière ensembliste. Nous avons détaillé notamment une approche de résolution ensembliste d'un problème de SLAM, le *DigSLAM*, qui utilise un capteur de distance et une formulation CSP.

La première contribution de cette thèse s'est intéressée à un problème de SLAM difficile dans un environnement non structuré, et à sa résolution ensembliste. En particulier, nous avons considéré des capteurs distance et angle, qui ne permettent d'obtenir qu'une information partielle de l'environnement du robot. À partir de ces informations et comme pour le *DigSLAM*, nous avons formalisé ce problème comme un CSP faisant intervenir des variables réelles, trajectoires et ensemblistes. Après une décomposition en contraintes plus simples à traiter, nous avons développé un formalisme intervalle afin de gérer les domaines ensemblistes : le pavage épais régulier. Les pavages intervalles épais de la littérature permettent de paver l'espace en boîtes et de les catégoriser en trois types : les boîtes intérieures, extérieures et incertaines. Les boîtes à cheval sur plusieurs régions ne peuvent pas être classifiées dans l'une de ces trois catégories et ne sont pas caractérisées finement. Les pavages épais que nous proposons reposent sur des étiquettes intervalle constituées d'un triplet d'intervalles de booléens, ce qui permet de bien caractériser toutes les boîtes du pavage. Avec ce cadre plus complet, nous avons pu développer une algèbre sur les pavages épais réguliers permettant de garantir les opérations ensemblistes classiques, ainsi qu'un algorithme de pavage dédié.

Avec ces nouveaux outils, nous avons pu résoudre ce problème de SLAM difficile de manière ensembliste. Nous avons validé l'approche sur quelques exemples simulés obtenus à partir d'une carte connue sous forme de pixels et d'une trajectoire analytique qui boucle dans cet environnement. A ce modèle "exact", on ajoute les modèles des capteurs (IMU et Lidar) qui comprennent une incertitude bornée et paramétrable. Une dizaine d'instants d'observation permettent d'aligner les cartes locales en ces points de passage pour mieux approximer la carte globale et la trajectoire. A chaque instant, un millier de mesures (Lidar) fournissent des cônes d'observation buttant chacun sur un point d'obstacle. L'algorithme *offline* développé permet de calculer en quelques minutes un très grand pourcentage du volume de la

carte qu'il est possible de recalculer compte-tenu des points de mesure retenus. On pourrait produire aisément une version *online* de cet algorithme.

La seconde contribution de cette thèse s'est intéressée au problème de recalage sous transformation de similarité. Après avoir étudié les méthodes classiques de résolution, nous nous sommes intéressé à l'approche que nous avons qualifiée de Procuste : il s'agit de recentrer et mettre à l'échelle les ensembles avant de résoudre le problème d'orientation (ou orthogonal) résultant. Cependant, nous avons aussi remarqué que le problème général peut avoir plusieurs solutions en raison de symétries que les méthodes d'optimisation numériques ne peuvent pas calculer, ce qui a motivé notre approche ensembliste. Nous avons utilisé des paramètres de centrage et normalisation différents de l'approche numérique classique, mais calculable par des séparateurs. Nous avons prouvé la correction de ce choix. En profitant de l'expressivité des opérations de séparation, nous avons pu concevoir facilement des séparateurs qui calculent l'ensemble des variables du problème de Procuste. Les résultats expérimentaux ont montré la capacité de la méthode à gérer la multiplicité des solutions.

La piste d'amélioration principale de cette thèse sera d'étendre les contributions à la dimension 3. Le seul verrou à cette extension est la rotation en dimension 3, qui n'a pas été étudiée ici pour deux raisons : d'une part, la combinatoire supplémentaire entre le groupe de rotation en 2D ($SO(2)$ de dimension 1) et celui en 3D ($SO(3)$ de dimension 3) ; d'autre part, la nécessité de développer une représentation des rotations plus appropriée comme les quaternions, afin de limiter le *wrapping effect* inévitable avec les matrices de rotation en dimension 3.

Bibliographie

[1] Lisa Gottesfeld Brown. A survey of image registration techniques. ACM Computing Surveys, 24(4) :325–376, December 1992.

[2] John C. Gower and Garmt B. Dijksterhuis. Procrustes Problems. OUP Oxford, January 2004.

[3] Ugo Montanari. Networks of constraints : Fundamental properties and applications to picture processing. Inf. Sci., 7 :95–132, 1974.

[4] L. Jaulin. Range-only SLAM with occupancy maps ; A set-membership approach. IEEE Transaction on Robotics, 27(5) :1004–1010, 2011.

[5] Ramon E Moore. Interval analysis, volume 4. Prentice-Hall Englewood Cliffs, 1966.

[6] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Éric Walter. Applied Interval Analysis. Springer, London, 2001.

[7] Ramon E Moore. Methods and applications of interval analysis. SIAM, 1979.

[8] Gilles Chabert and Luc Jaulin. Contractor programming. Artificial Intelligence, 173(11) :1079–1100, July 2009.

[9] Frédéric Benhamou, Frédéric Goualard, Laurent Granvilliers, and Jean-Francois Puget. Revising hull and box consistency. In Danny De Schreye, editor, Logic Programming : The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, pages 230–244. MIT Press, 1999.

[10] Gilles Chabert. Ibex documentation, 2012.

[11] Aymeric Bethencourt and Luc Jaulin. Solving Non-Linear Constraint Satisfaction Problems Involving Time-Dependant Functions. Mathematics in Computer Science, 8(3) :503–523, September 2014.

[12] Mario Milanese, John Norton, Hélène Piet-Lahanier, and Éric Walter. Bounding approaches to system identification. Springer Science & Business Media, 2013.

[13] Simon Rohou, Luc Jaulin, Lyudmila Mihaylova, Fabrice Le Bars, and Sandor M. Veres. Guaranteed computation of robot trajectories. Robotics and Autonomous Systems, 93 :76–84, July 2017.

[14] Simon Rohou, Abderahmane Bedouhene, Gilles Chabert, Alexandre Goldsztejn, Luc Jaulin, Bertrand Neveu, Victor Reyes, and Gilles Trombettoni. Towards a generic interval solver for differential-algebraic CSP. In Helmut Simonis, editor, Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings, volume 12333 of Lecture Notes in Computer Science, pages 548–565. Springer, 2020.

[15] Simon Rohou, Luc Jaulin, Lyudmila Mihaylova, Fabrice Le Bars, and Sandor M. Veres. Reliable non-linear state estimation involving time uncertainties. Autom., 93 :379–388, 2018.

[16] Raphael Voges. Bounded-error visual-LiDAR odometry on mobile robots under consideration of spatiotemporal uncertainties. PhD thesis, University of Hanover, Germany, 2020.

[17] Simon Rohou, Benoit Desrochers, and Fabrice Le Bars. The Codac library. Acta Cybernetica, Mar. 2024.

[18] Simon Rohou, Benoît Desrochers, and Luc Jaulin. Set-membership state estimation by solving data association. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 4393–4399, May 2020. ISSN : 2577-087X.

[19] Luc Jaulin. Solving set-valued constraint satisfaction problems. Computing, 94(2) :297–311, 2012.

[20] Benoît Desrochers and Luc Jaulin. Thick set inversion. Artificial Intelligence, 249 :1–18, 2017.

[21] Lotfi A Zadeh. Fuzzy sets. Information and Control, 1965.

[22] George Klir and Bo Yuan. Fuzzy sets and fuzzy logic, volume 4. Prentice hall New Jersey, 1995.

[23] Didier Dubois and Henri Prade. Fundamentals of fuzzy sets, volume 7. Springer Science & Business Media, 2012.

[24] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2) :91–110, November 2004.

[25] H. Bay, Tinne Tuytelaars, Luc Van Gool, A. Leonardis, H. Bischof, and A. Pinz. SURF : Speeded up robust features. In Lecture Notes in Computer Science, volume 3951. Springer, January 2006. ISSN : 3-540-33832-2, 9783540338321.

[26] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB : An efficient alternative to SIFT or SURF. In 2011 International Conference on Computer Vision, pages 2564–2571, November 2011. ISSN : 2380-7504.

[27] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2 : An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics, 33(5) :1255–1262, October 2017. Conference Name : IEEE Transactions on Robotics.

[28] Johann Borenstein, H. R. Everett, and Liqiang Feng. Where am I ? Sensors and methods for mobile robot positioning. University of Michigan, 119(120), 1996. Publisher : Citeseer.

[29] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM) : Part II. IEEE robotics & automation magazine, 13(3) :108–117, 2006. Publisher : IEEE.

[30] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. Autonomous robots, 4 :333–349, 1997.

[31] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375), pages 318–325, November 1999.

[32] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), volume 1, pages 206–211 vol.1, October 2003.

[33] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping : part I. IEEE Robotics & Automation Magazine, 13(2) :99–110, June 2006.

[34] A. Elfes. Using occupancy grids for mobile robot perception and navigation. Computer, 22(6) :46–57, 1989.

[35] Sebastian Thrun. Learning Occupancy Grid Maps with Forward Sensor Models. Autonomous Robots, 15(2) :111–127, September 2003.

[36] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. Pattern recognition, 26(9) :1277–1294, 1993.

[37] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In Proceedings. 1985 IEEE International Conference on Robotics and Automation, volume 2, pages 116–121, 1985.

[38] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. Intelligent robotics and autonomous agents. MIT Press, 2005.

[39] Hanan Samet. The quadtree and related hierarchical data structures. ACM Computing Surveys (CSUR), 16(2) :187–260, 1984.

[40] Donald Meagher. Geometric modeling using octree encoding. Computer Graphics and Image Processing, 19(2) :129–147, June 1982.

[41] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9) :509–517, September 1975.

[42] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. Estimation with Applications to Tracking and Navigation : Theory Algorithms and Software. John Wiley & Sons, June 2001. Google-Books-ID : NtvWDwAAQBAJ.

[43] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on Robotics and Automation, 17(3) :229–241, June 2001. Conference Name : IEEE Transactions on Robotics and Automation.

[44] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. Journal of the Royal Statistical Society : Series B (Methodological), 39(1) :1–22, 1977. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1977.tb01600.x.

[45] Wolfram Burgard, Dieter Fox, Hauke Jans, Christian Matenar, and Sebastian Thrun. Sonar-based mapping with mobile robots using em. In MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, pages 67–76. Citeseer, 1999.

[46] Sebastian Thrun. A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots. The International Journal of Robotics Research, 20(5) :335–363, May 2001. Publisher : SAGE Publications Ltd STM.

[47] Kevin Murphy and Stuart Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, Sequential Monte Carlo Methods in Practice, pages 499–515. Springer, New York, NY, 2001.

[48] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem. Aaai/iaai, 593598, 2002.

[49] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 2432–2437, April 2005. ISSN : 1050-4729.

[50] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing, 50(2) :174–188, 2002.

[51] Arnaud Doucet. On sequential simulation-based methods for Bayesian filtering. Department of Engineering, University of Cambridge, 1998.

[52] Sebastian Thrun and Michael Montemerlo. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. The International Journal of Robotics Research, 25(5-6) :403–429, May 2006.

[53] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino. Set membership localization and mapping for autonomous navigation. International Journal of Robust and Nonlinear Control, 11(7) :709–734, 2001. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/rnc.619.

[54] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino. A Set Theoretic Approach to Dynamic Robot Localization and Mapping. Autonomous Robots, 16(1) :23–47, January 2004.

[55] J.M. Porta. CuikSLAM : A Kinematics-based Approach to SLAM. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 2425–2431, April 2005. ISSN : 1050-4729.

[56] L. Jaulin. A Nonlinear Set Membership Approach for the Localization and Map Building of Underwater Robots. IEEE Transactions on Robotics, 25(1) :88–98, February 2009.

[57] Luc Jaulin and Eric Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. Automatica, 29(4) :1053–1064, 1993.

[58] Luc Jaulin and Alain Godon. Motion planning using interval analysis. In Proceedings of MISC'99 Workshop on Application of Interval Analysis to System and Control, volume 24, pages 335–346, 1999.

[59] Hermann Minkowski. Volumen und oberfläche. In Ausgewählte Arbeiten zur Zahlentheorie und zur Geometrie : Mit D. Hilberts Gedächtnisrede auf H. Minkowski, Göttingen 1909, pages 146–192. Springer, 1989.

[60] Hugo Hadwiger. Minkowskische addition und subtraktion beliebiger punktmengen und die theoreme von erhard schmidt. Mathematische Zeitschrift, 53(3) :210–218, 1950.

[61] Pijush K Ghosh. A unified computational framework for minkowski operations. Computers & Graphics, 17(4) :357–378, 1993.

[62] Benoît Desrochers and Luc Jaulin. A minimal contractor for the polar equation : Application to robot localization. Eng. Appl. Artif. Intell., 55 :83–92, 2016.

[63] Olivier Lhomme. Consistency techniques for numeric csps. In Ruzena Bajcsy, editor, Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993, pages 232–238. Morgan Kaufmann, 1993.

[64] Bertrand Neveu, Gilles Trombettoni, and Ignacio Araya. Adaptive constructive interval disjunction : algorithms and experiments. Constraints An Int. J., 20(4) :452–467, 2015.

[65] Michal Irani and Shmuel Peleg. Improving resolution by image registration. CVGIP : Graphical models and image processing, 53(3) :231–239, 1991.

[66] Roger P Woods, John C Mazziotta, Simon R Cherry, et al. MRI-PET Registration with Automated Algorithm. Journal of Computer Assisted Tomography, 17(4) :536–546, 1993.

[67] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In IJCAI'81 : 7th international joint conference on Artificial intelligence, volume 2, pages 674–679, 1981.

[68] Mani Golparvar Fard and Feniosky Peña-Mora. Application of Visualization Techniques for Construction Progress Monitoring. In Computing in Civil Engineering (2007), pages 216–223, Pittsburgh, Pennsylvania, United States, July 2007. American Society of Civil Engineers.

[69] Padmanabhan Anandan. A computational framework and an algorithm for the measurement of visual motion. International Journal of Computer Vision, 2(3) :283–310, 1989.

[70] Gary J Sullivan and Richard L Baker. Motion compensation for video compression using control grid interpolation. In Acoustics, Speech, and Signal Processing, IEEE International Conference on, pages 2713–2714. IEEE Computer Society, 1991.

[71] Tommi Tykkälä, Cédric Audras, and Andrew I Comport. Direct Iterative Closest Point for real-time visual odometry. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pages 2050–2056. IEEE, 2011.

[72] James Joseph Sylvester. A question in the geometry of situation. Quarterly Journal of Pure and Applied Mathematics, 1(1) :79–80, 1857.

[73] Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. SIAM journal on computing, 12(4) :759–776, 1983.

[74] Bert F. Green. The orthogonal approximation of an oblique structure in factor analysis. Psychometrika, 17(4) :429–440, December 1952.

[75] Norman Cliff. Orthogonal rotation to congruence. Psychometrika, 31(1) :33–42, March 1966.

[76] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations : a comparison of four major algorithms. Machine vision and applications, 9(5) :272–290, 1997.

[77] Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. Psychometrika, 31(1) :1–10, March 1966.

[78] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. Journal of the Optical Society of America A, 5(7) :1127–1135, 1988.

[79] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. JOSA A, 4(4) :629–642, April 1987.

[80] Peter H. Schönemann and Robert M. Carroll. Fitting one matrix to another under choice of a central dilation and a rigid motion. Psychometrika, 35(2) :245–255, June 1970.

[81] J. C. Gower. Generalized procrustes analysis. Psychometrika, 40(1) :33–51, March 1975.

[82] Jos MF Ten Berge. Orthogonal procrustes rotation for two or more matrices. Psychometrika, 42(2) :267–276, 1977.

[83] F James Rohlf and Dennis Slice. Extensions of the procrustes method for the optimal superimposition of landmarks. Systematic zoology, 39(1) :40–59, 1990.

[84] Adrien Bartoli, Daniel Pizarro, and Marco Loog. Stratified Generalized Procrustes Analysis. International Journal of Computer Vision, 101(2) :227–253, January 2013.

[85] Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In Sensor Fusion IV : Control Paradigms and Data Structures, volume 1611, pages 586–606. SPIE, April 1992.

[86] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. Image and Vision Computing, 10(3) :145–155, April 1992.

[87] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. International Journal of Computer Vision, 13(2) :119–152, October 1994.

[88] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP : A Globally Optimal Solution to 3D ICP Point-Set Registration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(11) :2241–2254, November 2016.

[89] Nicolas Bonneel and David Coeurjolly. SPOT : sliced partial optimal transport. ACM Transactions on Graphics, 38(4) :89 :1–89 :13, July 2019.

[90] Benoît Desrochers. Simultaneous localization and mapping in unstructured environments : a set-membership approach. phdthesis, ENSTA Bretagne - École nationale supérieure de techniques avancées Bretagne, May 2018.

[91] Luc Jaulin and Benoît Desrochers. Introduction to the Algebra of Separators with Application to Path Planning. ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE, 33, 2014.

[92] Luc Jaulin and Benoît Desrochers. Thick separators. In Martine Ceberio and Vladik Kreinovich, editors, Decision Making under Constraints, volume 276, pages 125–131. Springer, 2020.